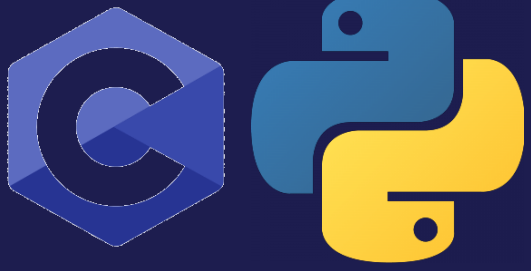
TRƯỜNG ĐHSP HUẾ
KHOA TIN HỌC

NGUYỄN THẾ DŨNG, LÊ THỊ ANH ĐỨC
NGUYỄN QUANG THUẬN, PHẠM HỒ NHƯ NGUYỆT
NGUYỄN VIẾT PHÙNG, HOÀNG VĂN DIỆU

MỘT SỐ CHUYÊN ĐỀ LẬP TRÌNH

(minh họa với C và Python)

[Ver 1]



I Love HUEdu
HUE UNIVERSITY OF EDUCATION
WHERE KNOWLEDGE BECOMES VALUE

FOLLOW YOUR DREAMS

HUẾ, 2022

Lời nói đầu

Tập tài liệu này khởi đầu là tư liệu sưu tầm tham khảo từ các nguồn để dạy bồi dưỡng học sinh giỏi của Cô giáo Lê Thị Anh Đức (giáo viên trường THPT Tây Hiếu, tỉnh Nghệ An), một cựu sinh viên Khoa Tin học – ĐHSP Huế. Tư liệu có hướng dẫn giải và có các chương trình minh họa bằng ngôn ngữ C khá chu đáo. Với xu thế chuyển đổi ngôn ngữ lập trình trong dạy học ở phổ thông hiện nay là ngôn ngữ Python. Tập tài liệu này hy vọng là một sự đóng góp nhỏ của thầy trò Khoa Tin học – ĐHSP Huế, nhằm giúp học sinh và các Thầy cô giáo có thêm nguồn tham khảo trong việc dạy và học lập trình với ngôn ngữ Python.

Tài liệu gồm các chuyên đề sau:

Nội dung
Chuyên đề 1: Kiến thức cơ bản về NNLT C và Python;
Chuyên đề 2: Số học;
Chuyên đề 3: Xử lý dãy số;
Chuyên đề 4: Xử lý xâu ký tự;
Chuyên đề 5: Đệ quy, quay lui;
Chuyên đề 6: Quy hoạch động;
Phân tích các đề thi, nhận dạng các dạng toán và giải quyết bài toán.

Tác giả Nguyễn Thế Dũng chỉ là người chủ xướng, biên tập, hiệu đính, và mời gọi các anh chị em cựu sinh viên, sinh viên của Khoa Tin học – ĐHSP Huế chuyển soạn tài liệu trên với minh họa bằng ngôn ngữ Python.

Trân trọng cảm ơn sự chung tay của các thầy cô, anh chị em: Nguyễn Quang Thuận (THPT chuyên Quốc Học - Huế), Phạm Hồ Như Nguyệt (THPT Phan Đăng Lưu – Huế), Nguyễn Viết Phùng (THPT Nguyễn Hữu Cảnh – Đồng Nai), Hoàng Văn Diệu (THPT chuyên Lê Quý Đôn – Quảng Trị) và nhiều anh chị em khác… (sắp xếp họ tên ở trên theo thứ tự chuyên mục tài liệu đã chuyển soạn).

Trong phiên bản đầu tiên này, các phần hướng dẫn giải và minh họa bởi C, chúng tôi giữ nguyên tác của tư liệu gốc và bổ sung các kiến thức cơ bản về Python cũng như bổ sung các chương trình minh họa bởi Python. Một số mã nguồn Python của các bài tập trong tài liệu này, sẽ được đưa lên đường link để người đọc có thể tham khảo tiện lợi.

Rất mong nhận được sự góp ý, đóng góp thêm cho tư liệu này của thầy cô, anh chị em gần xa để tài liệu được ngày một phong phú hơn, tạo nên một tài liệu hữu ích cho việc dạy học lập trình.

Huế, mùa thu 2022.

Đối với mỗi nội dung sẽ triển khai theo các hoạt động:

- Hướng dẫn học sinh tìm hiểu các kiến thức lý thuyết, các khái niệm, tính ứng dụng và cả kinh nghiệm về chủ đề.

- Giáo viên cùng học sinh giải quyết một vài ví dụ về bài toán cơ bản, tiêu biểu về chủ đề. Sau đó phát triển các toán này với các yêu cầu cao hơn, mở rộng hơn.

- Luyện tập giải một số bài tập theo chuyên đề; Giải một số bài tập mở rộng và nâng cao có vận dụng kiến thức kết hợp nhiều chủ đề đã được học ở trước.

1) Kiểu dữ liệu đơn giản và các phép toán cơ bản.

Một số kiểu dữ liệu thường dùng:

+ Kiểu logic: bool

+ Kiểu ký tự: char

+ Số nguyên: int, long long, unsigned long long

+ Số nguyên: float, double

Ta thấy rằng phạm vi kiểu dữ liệu càng lớn thì càng cần nhiều bộ nhớ và các đề thi học sinh giỏi ngày nay ngoài giới hạn về thời gian thực hiện chương trình còn giới hạn cả về dung lượng bộ nhớ vì vậy cần sử dụng các kiểu dữ liệu một cách hợp lý.

Các phép toán trên các loại dữ liệu đã được giới thiệu trong chương trình chính khóa nên tôi không nhắc lại ở đây.

Lưu ý trong một số trường hợp tính toán cụ thể hoặc cần biểu diễn giá trị dưới những định dạng khác nhau, chúng ta cần thực hiện **ép kiểu** để chuyển đổi qua lại giữa những kiểu dữ liệu có khả năng lưu trữ giá trị giống nhau.

Bài toán cơ bản

Viết chương trình nhập vào 2 số nguyên a, b ($a,b<=10^9$). Tính tổng (a+b), hiệu (a-b), tích (a*b), du (a%b), thương nguyên (a/b), thương thực a chia b hiển thị 2 chữ số sau dấu phẩy.

ere pheptinh.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int a,b, tong, hieu, du,thuongnguyen ;
long long tich;
float thuongthuc;
int main()
{
      cin>>a>>b;
      tong=a+b;
      hieu=a-b;
      tich=a*b;
      thuongnguyen=a/b;
      du=a%b;
      thuongthuc=(float)a/b;
     cout<<tong<<"   "<<hieu<<"    "<<tich<<endl;
     cout<<setprecision(2); cout <<fixed;
     cout<<du<<"   "<<thuongnguyen<<"   "<<thuongthuc<<"    ";
    return 0;
 }
```

Lưu ý học sinh:

- Khai báo kiểu dữ liệu phù hợp cho từng biến.

- Hai biến a,b là số nguyên nên khi thực hiện phép chia a/b ta sẽ thu được kết quả là nguyên. Khi muốn lấy kết quả là số thực ta cần ém kiểu (float). Ta có thể viết (float)a/b hoặc a/(float)b.

- Câu lệnh *cout<<setprecision(2);* để lấy 2 chữ số thập phân sau dấu phẩy.

- Câu lệnh *cout <<fixed;* để luôn hiển thị 2 chữ số thập phân. Ví dụ phép chia (float)4/2 nếu không có câu lệnh này thì kết quả hiện ra là 2. Khi có câu lệnh này thì kết quả hiển thị là 2.00.

2) Cấu trúc điều khiển:

- *Cấu trúc tuần tự.*
- *Cấu trúc rẽ nhánh: if…, if …. else…*
- *Lặp với số lần biết trước câu lệnh: for…*
- *Lặp với số lần chưa biết trước: while …*

Bài toán cơ bản:

Bài 1: Viết chương trình nhập vào số nguyên N ($1 < n <= 10^9$). Hãy đưa ra màn hình các số chẵn thuộc đoạn từ 1 đến N.

Ý tưởng:

Cách 1: Sử dụng vòng lặp với giá trị đầu là 1, bước nhảy 1 để duyệt tất cả các giá trị i thuộc đoạn từ 1 đến N. Dùng lệnh rẽ nhánh kiểm tra giá trị i nếu là số chẵn thì xuất ra màn hình.

Cách 2: Sử dụng vòng lặp với giá trị đầu là 2, bước nhảy 2 để duyệt tất cả các giá trị i là số chẵn để xuất ra màn hình.

Sau đây chương trình sử dụng câu lệnh for và chương trình sử dụng câu while để giải bài toán.

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
int main()
{
    cin>>n;
    for(int i=1; i<=n; i++)
        if(i%2==0)
        cout<< i<<" ";
   return 0;
}
```

Cách 1: Dùng câu lệnh for

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
int main()
{
    cin>>n;
    for(int i=2; i<=n; i+=2)
        cout<< i<<" ";
   return 0;
}
```

Cách 2: Dùng câu lệnh for

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
int main()
{
    cin>>n;
    int i=1;
    while (i<=n)
    {if (i%2==0)
        cout<< i<<" ";
    i++;
    }
    return 0;
}
```

Cách 1: Dùng câu lệnh while

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
int main()
{
    cin>>n;
    int i=2;
    while (i<=n)
    {   cout<< i<<" ";
        i+=2;
    }
   return 0;
}
```

Cách 2: Dùng câu lệnh while

Đánh giá thuật toán:

Trong 2 ý tưởng tưởng trên ta thấy cách 1 độ phức tạp thuật toán là O(n). cách 2 độ phức tạp là O(n/2). Như vậy ta nên chọn cách 2 để giải quyết bài toán.

Lưu ý: *Câu lệnh lặp có độ phức tạp lớn khi số lần lặp lớn. Vì vậy ta cần nghiên cứu tìm cách làm giảm số lần lặp xuống hết mức có thể.*

3) Kiểu dữ liệu có cầu trúc.

Ngôn ngữ lập trình C++ cung cấp rất nhiều kiểu dữ liệu có cấu trúc. Hai kiểu cơ bản thường dùng nhất đó là kiểu mảng một chiều và xâu dữ liệu. Ngoài ra, trong quá

trình giảng dạy để thuận lợi hơn trong việc lưu trữ và xử lý chúng ta có thể giới thiệu thêm cho học sinh các kiểu khác như: *pair, vecto*…

***Kiểu dữ liệu mảng một chiều:**

- Khái niệm về mảng một chiều
- Khai báo mảng
- Truy nhập phần tử mảng
- Nhập/xuất mảng

Khi nhập, xuất hay duyệt các phần tử của mảng ta cần dùng lệnh for để truy cập đến các phần tử của mảng.

***Kiểu dữ liệu Xâu ký tự:**

- Khai báo xâu.

- Nhập xâu bằng lệnh: cin, getline
- Xuất xâu: cout
- Một số thao tác trên xâu:

+ Phép ghép nối xâu: +

+ Phép so sánh: >, >=, <, <=, ==, !=

+ Hàm lấy độ dài xâu s: s.length() hoặc s.size();

+ Hàm sao chép trong xâu s từ chỉ số VT lấy n ký tự: s.substr(VT, n);

+ Hàm chèn thêm vào xâu x vào xâu s tại vị trí VT: s.insert(VT, x);

+ Hàm xóa trong xâu s từ chỉ số VT xóa đi n ký tự: s.erase(VT, n);

+ Hàm tìm kiếm: tìm vị trí đầu tiên xuất hiện xâu x trong xâu s: s.find(x). Nếu xâu x không có trong xâu S thì kết quả = -1

+ Hàm tìm kiếm ngược: tìm vị trí cuối cùng xuất hiện xâu x trong xâu s: s.rfind(x). Nếu xâu x không có trong xâu S thì kết quả = -1

+ Hàm đổi ký tự thành ký tự hoa: toupper(ch);

+ Hàm đổi ký tự thành ký tự thường: tolower(ch);

***Bài toán cơ bản:**

Bài 1: Cho dãy số A gồm N số nguyên A1, A2, ..., AN. Hãy tìm giá trị lớn nhất của dãy số A và chỉ số các phần tử có giá trị là lớn nhất.

Dữ liệu vào: Gồm 2 dòng:

- Dòng đầu tiên chứa số N ($N<=10^6$).
- Dòng thứ hai chứa dãy số A, các số ghi cách nhau ít nhất là một ký tự trống ($1<=a[i]<=10^9$).

Dữ liệu ra: Gồm 2 dòng:

- Dòng đầu tiên chứa một số nguyên là giá trị lớn nhất của dãy A.
- Dòng thứ hai chứa dãy số là chỉ số các phần tử có giá trị là lớn nhất, các số ghi cách nhau ít nhất là một ký tự trống.

Ví dụ:

Input	Output
10 1 2 5 5 8 3 8 8 2 2	8 5 7 8

Ý tưởng giải quyết bài toán:

Cách 1: - Đọc các giá trị vào mảng A và tìm giá trị lớn nhất Max của dãy.

- Duyệt tất cả các phần tử của dãy A để đưa ra chỉ số các phần tử Max

Cách 2: Đọc các giá trị của dãy, vừa đọc vừa tìm các giá trị lớn nhất và lưu các chỉ số của các phần tử lớn nhất vào mảng B. (Khi duyệt đến phần tử i thì mảng B sẽ lưu chỉ số của các phần tử lớn nhất trong các phần tử từ 1 đến i).

Chương trình diễn đạt các ý tưởng như sau:

here | *Mang.cpp

```
#include <bits/stdc++.h>
using namespace std;
int n,Max, a[1000002];

int main()
{
      cin>>n;
      Max=INT_MIN;
      for(int i=1;i<=n;i++)
         {cin>>a[i];
         if (a[i]>Max) Max=a[i];
         }
     cout<<Max<<'\n';
     for(int i=1;i<=n;i++)
         if (a[i]==Max) cout<<i<<" ";
return 0;
 }
```

Cách 1

Mangc2.cpp

```
#include <bits/stdc++.h>
using namespace std;
int n,k,X,Max,B[1000002];
int main()
{      cin>>n;
      Max=INT_MIN; k=0;
      for(int i=1;i<=n;i++)
         {cin>>X;
         if (X==Max)
             {k++; B[k]=i;}
         else
         if (X>Max)
             {Max=X; k=1; B[k]=i;}
         }
     cout<<Max<<'\n';
     for(int i=1;i<=k;i++)
         cout<<B[i]<<" ";
return 0;
 }
```

Cách 2

Phân tích thuật toán:

- Cả 2 cách đều phải dùng đến 2 vòng for để duyệt, nhưng cách 2 sẽ thực hiện nhanh hơn cách 1 vì ở vòng for thứ hai cách 1 lặp n lần, cách 2 lặp k lần mà k luôn nhỏ hơn hoặc bằng n.

- Thông qua chương trình ở cách 2, giáo viên nhấn mạnh cho học sinh cách tạo mảng lưu kết quả trong khi duyệt.

Bài 2: Để chuẩn bị cho kỳ thi nghề THPT sắp tới, cô đã cho các nhóm đăng ký dự thi và gửi danh sách đăng ký cho bạn An lớp trưởng tổng hợp thành danh sách đăng ký của lớp. Do các nhóm làm việc cẩu thả nên danh sách gửi cho An còn rất nhiều lỗi chính tả: Giữa các từ còn gõ nhiều dấu cách, họ tên học sinh chưa viết hoa đầu từ.

Em hãy viết chương trình: nhập vào xâu s là họ tên của một học sinh, xuất ra xâu đã xóa các dấu cách thừa và viết hoa đầu từ.

Ý tưởng giải quyết bài toán:

- Các công việc chính: Khai báo, nhập xâu, xóa dấu cách thừa, viết hoa đầu từ, xuất kết quả.

- Xóa dấu cách thừa:

+ Cách 1: Dùng câu lệnh for duyệt lùi, cứ gặp 2 dấu cách liền nhau thì dùng lệnh s.erase() xóa đi 1. (Lưu ý với hs nếu dùng for tiến sẽ k xóa hết được).

+ Cách 2: Dùng lệnh while duyệt tiến để xóa.

+ Cách 3: Dùng lệnh while kết hợp hàm s.find để xóa.

+ Cách 4: Dùng thêm biến phụ để lưu kết quả khi duyệt for tiến.

- Chuyển ký tự đầu từ thành chữ hoa: Duyệt xâu, cứ gặp ký tự khác cách mà ký tự trước đó là dấu cách thì sẽ chuyển ký tự này thành ký tự hoa.

Lưu ý: Cần xử lý tình huống đặc biệt đầu dãy.

Chương trình diễn đạt các ý tưởng như sau:

Cách 1:

```
#include <bits/stdc++.h>
using namespace std;
string s;
long l, i;
int main()
{
    getline(cin,s);
    l = s.length();
    /************************/
    for (int i=l-1; i>=0; i--)
        if ((s[i]==' ')&&(s[i-1]==' '))
           s.erase(i,1);
    if (s[0]==' ') s.erase(0,1);
    /************************/
        s[0]=toupper(s[0]);
    for (int i=0; i<s.length(); i++)
        if((s[i] != ' ') && (s[i-1]==' '))
              s[i] = toupper(s[i]);
    /************************/
    cout<<s;
    return 0;
}
```

Cách 2::

```
#include <bits/stdc++.h>
using namespace std;
string s;
long l, i;
int main()
{
    getline(cin,s);
    l = s.length();
    /************************/
    i=0;
    while (i<s.length())
        if ((s[i]==' ')&&(s[i+1]==' '))
           s.erase(i,1); else  i++;
    if (s[0]==' ') s.erase(0,1);
    /************************/
        s[0]=toupper(s[0]);
    for (int i=0; i<s.length(); i++)
        if((s[i] != ' ') && (s[i-1]==' '))
              s[i] = toupper(s[i]);
    /************************/
    cout<<s;
    return 0;
}
```

Cách 3:

```
#include <bits/stdc++.h>
using namespace std;
string s;
long  i;
int main()
{
    getline(cin,s);
    /************************/
     while (s.find("  ")!=-1)
        {   i=s.find("  ");
            s.erase(i,1);
         }
    if (s[0]==' ') s.erase(0,1);
    /************************/
        s[0]=toupper(s[0]);
    for (int i=0; i<s.length(); i++)
        if((s[i] != ' ') && (s[i-1]==' '))
              s[i] = toupper(s[i]);
    /************************/
    cout<<s;
    return 0;
}
```

Cách 4:

```
#include <bits/stdc++.h>
using namespace std;
string s, sl;
long l, i;
int main()
{
    getline(cin,s);
    l = s.length();
    /************************/
    sl = "";
    for (int i=0; i<l; i++)
    if((s[i]!=' ')||((s[i]==' ')&&(s[i+1]!=' ')
        sl = sl + s[i];
    if (sl[0]==' ') sl.erase(0,1);
    /************************/
        sl[0]=toupper(sl[0]);
    for (int i=0; i<sl.length(); i++)
        if((sl[i] != ' ') && (sl[i-1]==' '))
              sl[i] = toupper(sl[i]);
    /************************/
    cout<<sl;
    return 0;
}
```

Thông qua các chương trình nhấn mạnh cho học sinh các thao tác đã được vận dụng trong bài toán này, như hàm s.length(); s.find(); s.erase(); toupper(); +…

4) Kiểu dữ liệu tệp:

Trong các kỳ thi học sinh giỏi, thường chỉ yêu cầu xử lý với một tệp Input và một tệp Output. Như vậy để làm việc với tệp trước tiên ta cần mở tệp bằng hai câu lệnh sau:

freopen(<tên tệp input>,"r", stdin); freopen(<tên tệp output>,"w", stdout);

- Đọc tệp khi số phần tử đã xác định, sau khi mở tệp ta thực hiện bình thường như đọc vào từ bàn phím.

- Khi đọc tệp với số phần tử chưa xác định ta cần kết hợp câu lệnh while. Ví dụ đọc tệp vào mảng A với số phần tử chưa xác định:

int i = 1;

while (cin >> a[i])

{ i++; }

Đọc các dòng của tệp vào xâu S với số phần tử chưa xác định:

while (getline(cin,S))

- Câu lệnh đưa con trỏ xuống dòng trong tệp Input: *cin.ignore() ;*

Bài tập cơ bản:

Bài 1: Cho dãy số A gồm N số nguyên A1, A2, ..., AN (n<=10^6), giá trị các số không vượt quá 10^9. Hãy cho biết trong tệp có bao nhiêu số và đưa ra các số lẻ trong dãy.

Trường hợp 1: Dữ liệu vào từ Tệp TEPSO.INP

- Dòng đầu chứa số nguyên n.
- Dòng thứ 2 chứa n số nguyên ghi cách nhau ít nhất là một ký tự trống.

Trường hợp 2: Dữ liệu vào Tệp TEPSO.INP

Chứa n số nguyên ghi cách nhau ít nhất là một ký tự trống.

Kết quả ra: Ghi vào tệp TEPSO.OUT gồm 2 dòng:

- Dòng đầu tiên chứa một số nguyên là số lượng số có mặt trong tệp.
- Dòng thứ hai chứa dãy số lẻ có trong dãy theo tuần tự từ trước ra sau, các số ghi cách nhau ít nhất là một ký tự trống.

Ý tưởng giải quyết:

Trường hợp 1:

+ Đọc n từ tệp Input ghi vào tệp Output.

+ Ta chỉ cần duyệt 1 lần bằng lệnh for để vừa đọc vừa kiểm tra tìm các số lẻ.

Trường hợp 2: Ta cần duyệt 2 lần trên các số:

+ Lần 1 dùng câu lệnh while kết hợp lệnh cin để đọc các số vào mảng và đếm số lượng số.

+ Duyệt các số trên mảng để tìm các số lẻ.

Chương trình giải:

Tepso.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n,a[1000002];
int main()
{
 freopen("Tepso.inp","r",stdin);
 freopen("Tepso.out","w",stdout);
    cin>>n;
    cout<<n;
    for(int i=1; i<=n; i++)
        {cin >> a[i];
        if (a[i]%2!=0)
            cout<<a[i]<<" ";
        }
    return 0;
}
```

Trường hợp 1

Tepso.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n,a[1000002];
int main()
{
 freopen("Tepso.inp","r",stdin);
 freopen("Tepso.out","w",stdout);
    int  i = 1;
    while (cin >> a[i])
        {   i++; }
        n=i-1;
    cout<<n<<endl;
    for(int i=1;i<=n;i++)
        if (a[i]%2!=0)
            cout<<a[i]<<" ";
    return 0;
}
```

Trường hợp 2

Bài 2: Cho tệp văn bản TEPXAU.INP ghi họ tên học sinh trong một lớp học. Hãy cho biết trong tệp có bao nhiêu học sinh và đưa ra họ tên của học sinh dài nhất.

Trường hợp 1: Dữ liệu vào: Tệp TEPXAU.INP

Gồm nhiều dòng, mỗi dòng là họ tên của 1 học sinh.

Trường hợp 2: Dữ liệu vào: Tệp TEPXAU.INP.

- Dòng đầu tiên chứa số nguyên n là số học sinh trong lớp.
- n dòng tiếp theo mỗi dòng là họ tên của 1 học sinh.

Kết quả ra: Ghi vào tệp TEPSO.OUT gồm 2 dòng:

- Dòng đầu tiên chứa số lượng học sinh trong lớp.
- Dòng thứ hai ghi họ tên của học sinh dài nhất.

Ý tưởng giải quyết:

Trường hợp 1: Ta dùng câu lệnh while kết hợp lệnh getline(cin,s) để đọc và đếm số học sinh.

Trường hợp 2:

- Ta dùng lệnh cin để đọc số lượng học sinh
- Dùng lệnh *cin.ignore();* để đưa con trỏ xuống dòng.
- Dùng câu lệnh for kết hợp lệnh getline(cin,s) để đọc từng dòng.

Lưu ý: Ta nên vừa đọc vừa xử lý để không phải lưu lại danh sách trong mảng và tránh việc duyệt danh sánh nhiều lần.

Chương trình:

*Tepxau.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n=1,l,lm=0;
string s,sm="";
int main()
{
 freopen("Tepxau.inp","r",stdin);
 freopen("Tepxau.out","w",stdout);
    while (getline(cin,s))
        {   l=s.size();
            if(l>lm)
                {sm=s;
                 lm=l;
                }
            n++;
        }
    cout<<n-1<<endl;
    cout<<sm;
    return 0;
}
```

Trường hợp 1

*Tepxau.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n=1,l,lm=0;
string s,sm="";
int main()
{
 freopen("Tepxau.inp","r",stdin);
 freopen("Tepxau.out","w",stdout);
   cin>>n;
   cin.ignore();
   for(int i=1; i<=n;i++)
       {getline(cin,s);
        l=s.size();
        if(l>lm)
           {sm=s;
            lm=l;
           }
       }
    cout<<n<<endl;
    cout<<sm;
    return 0;
}
```

Trường hợp 2

5. *Chương trình con và lập trình có cấu trúc.*

- Chương trình con đóng vai trò quan trọng trong lập trình, đặc biệt trong lập trình có cấu trúc.

- Có 2 loại hàm đó là hàm có giá trị trả về và hàm không có giá trị trả về.
- Biến toàn cục, biến cục bộ.
- Tham số hình thức, tham số thực sự;
- Tham trị, tham biến.

Bài toán cơ bản:

Cho dãy số A gồm N số nguyên A1, A2, ..., AN. Hãy cho biết có bao nhiêu số trong dãy số A có tổng các chữ số là một số chính phương.

Dữ liệu vào: Tệp CTC.INP gồm 2 dòng:

- Dòng đầu tiên chứa số N ($N<=10^4$).
- Dòng thứ hai chứa dãy số A, các số ghi cách nhau ít nhất là một ký tự trống ($1<=a[i]<=10^9$).

Dữ liệu ra: Tệp CTC.OUT chứa 1 số nguyên duy nhất là kết quả của bài toán.

Ý tưởng: Để giải bài toán ta cần thực hiện các công việc sau:

- Xây dựng hàm **void doc()** để đọc dữ liệu từ tệp vào mảng.
- Xây dựng hàm **int tongcs(int n)** đưa vào số n, lấy ra tổng chữ số của n.
- Xây dựng hàm **bool cphuong(int n)** đưa vào số nguyên n kiểm tra n có phải là chính phương hay không.
- Hàm int main()

+ Gọi hàm doc();

+ Sau đó duyệt từng phần tử của mảng gọi hàm tính tổng các chữ số và kiểm tra chính phương để đếm số lượng số thỏa mãn.

Chương trình giải quyết bài toán:

CTC.cpp

```
#include <bits/stdc++.h>
using namespace std;
int n,k,dem=0, a[100002];
/*************************/
void doc()
{    cin>>n;
     for(int i=1; i<=n; i++)
         cin >> a[i];
}
/*************************/
int tong(int x)
{     int t=0;
     while (x!=0)
     {    t = t + x%10;
          x = x/10; }
     return t;
}
/*************************/
bool cphuong(int x)
{     int t = sqrt(x);
     if (t*t==x) return true;
     else return false;
}
/*************************/
int main()
{
 freopen("CTC.inp","r",stdin);
 freopen("CTC.out","w",stdout);
     doc();
     for(int i=1; i<=n; i++)
          {k=tong(a[i]);
           if(cphuong(k)==true)
              dem++;
          }
   cout<< dem;
   return 0;
}
```

Trong chương trình trên ta đã sử dụng một hàm có sẵn xây dựng 3 hàm:

- Hàm có sẵn **sqrt(n);**

- Hàm không có giá trị trả về **void doc(),**
- Hàm giá trị trả về là kiểu số nguyên **int tong(int x)**
- Hàm trả về giá trị logic **bool cphuong(int x)**

Lưu ý:

Khi giải quyết mỗi bài toán ta nên dành thời gian để phân tích, xây dựng bố cục cấu trúc chương trình. Công việc này sẽ đơn giản và dễ dàng hơn khi ta sử dụng bí quyết TTNV (Tách, Tìm, Nhìn, Viết) để giải quyết vấn đề.

+ Tách (Tách lớn ra nhỏ): Tách công việc lớn thành nhiều công việc nhỏ hơn, dễ giải quyết hơn. Có thể tách lớn ra nhỏ nhiều lần cho đến khi công việc đủ đơn giản dễ thực hiện.

+ Tìm (Tìm điểm chung): Tìm điểm chung giữa các công việc nhỏ với nhau.

+ Nhìn (Nhìn tổng quát): Nhìn tổng quát để tìm ra điều cốt lõi bằng cách lược bỏ hoặc đơn giản hóa các chi tiết.

KIẾN THỨC CƠ BẢN NGÔN NGỮ LẬP TRÌNH PYTHON

1) Kiểu dữ liệu đơn giản và các phép toán cơ bản

a. Biến và các kiểu dữ liệu cơ bản

• **Biến (variable)** đại diện cho vùng nhớ lưu trữ dữ liệu (trên RAM) của chương trình. Biến chứa giá trị nhập vào, giá trị của một biểu thức, giá trị tính toán hoặc xử lý trong chương trình. Biến được nhận dạng thông qua tên biến và kiểu dữ liệu.

• Python cũng giống như một số các ngôn ngữ bậc cao khác, **khi ta khai báo biến thì kiểu dữ liệu của nó sẽ tự động được detect**. Vì vậy nên chúng ta cũng không phải quá vất vả khi khai báo 1 biến.

```
#string
name = "Vũ Thanh Tài"
#integer
age = 22
#float
point = 8.9
#lists
option = [1,2,3,4,5]
#Tuple
tuple = ('Vũ Thanh Tài', 22 , True)
#Dictionary
  dictionary = {"name": "Vu Thanh Tai", "age": 22,
  "male": True}
```

• Trong python, để kiểm tra kiểu dữ liệu của một biến thì chúng ta có thể sử dụng hàm type với cú pháp như sau: **type(data)**

Ví dụ:

```
1   age = 22
2   print(type(age))
```

Màn hình thực thi hiển thị <class 'int'>

- Trong một trường hợp nào đó mà bạn muốn chuyển đổi kiểu dữ liệu của một biến, thì Python cũng hỗ trợ bạn qua các hàm cơ bản sau:
  - **float(data)** chuyển đổi sang kiểu số thực.
  - **int(data,base)** chuyển đổi sang kiểu số, trong đó base là kiểu hệ số mà các bạn muốn chuyển đổi sang (tham số này có thể bỏ trống).
  - **str(data)** chuyển đổi sang dạng chuỗi.
  - **complex(data)** chuyển đổi sang kiểu phức hợp.
  - **tuple(data)** chuyển đổi sang kiểu Tuple.
  - **dict(data)** chuyển đổi sang kiểu Dictionary.
  - **hex(data)** chuyển đổi sang hệ 16.
  - **oct(data)** chuyển đổi sang hệ 8.
  - **chr(data)** chuyển đổi sang dạng ký tự.

b. Các toán tử trong Python

Ở phần kiểu dữ liệu number trong Python mình cũng nói qua về một số toán tử trong Python rồi, nhưng đó mới chỉ là một phần nhỏ thôi, trên thực tế thì Python sẽ có các toán tử cơ bản như sau:

- Toán tử số học - Arithmetic Operators
- Toán tử quan hệ - Comparison (Relational) Operators
- Toán tử gán - Assignment Operators.
- Toán tử logic - Logical Operators.
- Toán tử Biwter - Bitwise Operators.
- Toán tử khai thác - Membership Operators.
- Toán tử xác thực - Indentity Operators.

1, Toán tử số học - Arithmetic Operators.

Toán tử số học trong python được thể hiện dưới 7 dạng cơ bản sau: (trong các ví dụ dưới đây thì ta coi a có giá trị là 5 và b có giá trị là 7).

Toán tử	Mô Tả	Ví Dụ
+	Toán tử cộng các giá trị lại với nhau	a + b = 12
-	Toán tử trừ các giá trị lại với nhau	a - b = -2
*	Toán tử nhân các giá trị lại với nhau	a * b = 42
/	Toán tử chia các giá trị cho nhau	a / b = 0.7142857142857143
%	Toán tử chia lấy phần dư	a % b = 5
**	Toán tử mũ. a**b = a^b	a ** b = 78125
//	Toán tử chia làm tròn xuống. **VD**: 0,57 => 0	a // b = 0

	0.9 => 0 -07 => -1 -0.1 => -1	

2. Toán tử Quan hệ

Dạng toán tử này dùng để so sánh các giá trị với nhau kết quả của nó sẽ trả về là True nếu đúng và False nếu sai. Và nó thường được dùng trong các câu lệnh điều kiện.

Trong Python thì nó cũng tồn tại 6 dạng toán tử quan hệ cơ bản như sau:

(trong các ví dụ dưới đây thì ta coi a có giá trị là 5 và b có giá trị là 7).

Toán tử	Chú Thích	Ví Dụ
==	So sánh giá trị của các đối số xem có bằng nhau hay không. Nếu bằng nhau thì kết quả trả về sẽ là True và ngược lại sẽ là False.	a == b // False
!=	So sánh giá trị của các đối số xem có khác nhau hay không. Nếu khác nhau thì kết quả trả về sẽ là True và ngược lại sẽ là False.	a != b //True
<	Dấu < đại diện cho phép toán nhỏ hơn, nếu đối số 1 nhỏ hơn đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a < b //True
>	Dấu > đại diện cho phép toán lớn hơn, nếu đối số 1 lớn hơn đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a > b //False
<=	Dấu > đại diện cho phép toán nhỏ hơn hoặc bằng, nếu đối số 1 nhỏ hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a <= b //True
>=	Dấu > đại diện cho phép toán lớn hơn hoặc bằng, nếu đối số 1 lớn hơn hoặc bằng đối số 2 thì kết quả sẽ trả về là True và ngược lại sẽ là False.	a>= b //False

3. Toán tử gán

Toán tử gán là toán tử dùng để gán giá trị của một đối tượng cho một đối tượng khác. Và trong Python thì nó cũng được thể hiện giống như các ngôn ngữ khác. Và dưới đây là 8 toán tử nằm trong dạng này mà Python hỗ trợ.

Toán Tử	Chú Thích	Ví Dụ
=	Toán tử này dùng để gán giá trị của một đối tượng cho một giá trị	c = a (lúc này c sẽ có giá trị = 5)
+=	Toán tử này cộng rồi gắn giá trị cho đối tượng	c += a (tương đương với c = c + a)
-=	Toán tử này trừ rồi gắn giá trị cho đối tượng	c -= a (tương đương với c = c - a)
*=	Toán tử này trừ rồi gắn giá trị cho đối tượng	c *= a (tương đương với c = c * a)
/=	Toán tử này chia rồi gắn giá trị cho đối tượng	c /= a (tương đương với c = c / a)
%	Toán tử này chia hết rồi gắn giá trị cho đối tượng	c %= a (tương đương với c = c % a)
**=	Toán tử này lũy thừa rồi gắn giá trị cho đối tượng	c **= a (tương đương với c = c ** a)
//=	Toán tử này chia làm tròn rồi gắn giá trị cho đối tượng	c //= a (tương đương với c = c // a)

4. Toán tử logic

Toán tử logic trong Python hoàn toàn giống như các ngôn ngữ khác. Nó gồm có 3 kiểu cơ bản như sau:

Toán Tử	Chú Thích
and	Nếu 2 vế của toán tử này đều là True thì kết quả sẽ là True và ngược lại nếu 1 trong 2 vế là False thì kết quả trả về sẽ là False.

or	Nếu 1 trong 2 vế là True thì kết quả trả về sẽ là True và ngược lại nếu cả 2 vế là False thì kết quả trả về sẽ là False.
not	Đây là dạng phủ định, nếu biểu thức là True thì nó sẽ trả về là False và ngược lại.

5. Toán tử biwter

Toán tử này thực hiện trên các bit của các giá trị. Hãy tưởng tượng mình có 2 biến a = 12 và b = 15 nhưng nếu chúng ta convert chúng sang hệ nhị phân thì 2 biến này sẽ có giá trị như sau: a = 00001100 và b = 00001111. Về phần này thì rất ít khi sử dụng và cũng khó translate sang tiếng việt nên mình xin được phép viết toán hạng và ví dụ thôi.

Toán Tử	Ví Dụ
&	(a & b) = 12 (00001100)
|	(a | b) = 14 (00001111)
^	(a ^ b) = 2 (00000010)
~	(-a) = -13 (00001101)
<<	a<<a = 49152
>>	a>>a = 0

6. Toán Tử khai thác

Toán tử này thường được dùng để kiểm tra xem 1 đối số có nằm trong 1 tập hợp đối số hay không (list). Trong Python hỗ trợ chúng ta 2 dạng toán tử như sau:

Giả sử: a = 4, b = [1,5,7,6,9]

Toán Tử	Chú Thích	Ví Dụ
in	Nếu 1 đối số thuộc một tập đối số nó sẽ trả về True và ngược lại/	a in b //False
not in	Nếu 1 đối số **không** thuộc một tập đối số nó sẽ trả về True và ngược lại/	a not in b //True

7. Toán tử xác thực

Dạng Toán tử này dùng để xác thực hai giá trị xem nó có bằng nhau hay không. Và trong Python hỗ trợ chúng ta 2 dạng sau:

Giả sử: a = 4, b =5

Toán Tử	Chú Thích	Ví Dụ
is	Toán tử này sẽ trả về True nếu a == b và ngược lại	a is b //False
not is	Toán tử này sẽ trả về True nếu a != b và ngược lại	a is not b //True

Bài toán cơ bản

Viết chương trình nhập vào 2 số nguyên a, b (a,b<=10^9). Tính

- tổng (a+b),
- hiệu (a-b),
- tích (a*b),
- du (a%b),
- thương nguyên (a/b),
- thương thực a chia b hiển thị 2 chữ số sau dấu phẩy.

demo.py

F: > python > demo.py > ...

```
1  a=int(input("Nhập số thứ 1:"))
2  b=int(input("Nhập số thứ 2:"))
3  print(a+b)
4  print(a-b)
5  print(a*b)
6  print(a%b)
7  print(int(a/b))
8  print(format(a/b, '.2f'))
```

5) Cấu trúc điều khiển:

- Cấu trúc tuần tự.

- Cấu trúc rẽ nhánh: if..., if .... else...

- Lặp với số lần biết trước câu lệnh: for...

- Lặp với số lần chưa biết trước: while ...

Bài toán cơ bản:

Bài 1: Viết chương trình nhập vào số nguyên N ($1 < n <= 10^9$). Hãy đưa ra màn hình các số chẵn thuộc đoạn từ 1 đến N.

Sử dụng câu lệnh lặp for và range	Sử dụng câu lệnh while

```
n=int(input())
for i in range(1,n+1):
    if (i%2==0):
        print(i,end=' ')
```

```
n=int(input())
i=1
while i<=n:
    if (i%2==0):
        print(i,end=' ')
    i=i+1
```

3. KIỂU DỮ LIỆU LIST

3.1 List là gì? và khai báo list trong Python.

List trong Python là một dạng dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu khác nhau trong nó, và chũng ta có thể truy xuất đến các phần tử bên trong nó thông qua vị trí của phần tử đó trong list. Ở đây, nếu như bạn nào đã tìm hiểu qua một ngôn ngữ nào đó thì có thể coi list trong Python như một mảng tuần tự trong các ngôn ngữ khác.

Để khai báo một list trong Python thì chúng ta sử dụng cặp dấu [] và bên trong là các giá trị của list.

```
[value1, value2,..., valueN]
```

Trong đó: value1, value2,..., valueN là các giá trị của list.

VD: Mình sẽ khai báo 1 list chứa tên các học sinh.

```
name = ['VU Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
```

3.2. Truy cập đến các giá trị trong list.

Để truy cập đến các phần tử trong list thì các bạn làm tương tự như đối với chuỗi.

Các phần tử trong một list được đánh dấu bắt đầu từ 0 theo chiều từ trái sang phải và từ -1 theo chiều từ phải qua trái.

VD: Mình sẽ truy xuất đến từng phần tử một của list trong VD trên.

```
name = ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
print(name[0]) # Vu Thanh Tai
print(name[1]) # Nguyen Van A
print(name[2]) # Nguyen Thi E
# hoặc
print(name[-3]) # Vu Thanh Tai
```

```
print(name[-2]) # Nguyen Van A
print(name[-1]) # Nguyen Thi E
```

Trong trường hợp bạn muốn in ra một phần của list thì bạn sử dụng cú pháp sau:

```
list[start:end]
```

Trong đó:

- list là tên của biến chứa list.
- start là ví trí bắt đầu lấy ra list con. Nếu để trống thì nó sẽ lấy từ đầy list.
- end là vị trí kết thúc. Nếu để trống thì nó sẽ lấy đến phần tử cuối cùng của list.

VD: Lấy ra 2 phần tử đầu của list trên.

```
name = ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
print(name[0:2])
# ['Vu Thanh Tai', 'Nguyen Van A']
# hoặc
print(name[-3:-1])
# ['Vu Thanh Tai', 'Nguyen Van A']
```

3.3 Sửa đổi và xóa bỏ giá trị phần tử trong list.

Sau khi bạn đã truy cập được đến các phần tử trong list rồi thì bạn có thể xử lý nó như nào tùy thích theo ý của bạn (sửa - xóa).

Update

Để sửa giá trị của các phần tử trong list thì các bạn chỉ cần truy cập đến phần tử mà mình cần sửa đổi và tiến hành gán giá trị mới cho nó.

VD: Sửa name thứ 2 trong list ở ví dụ trên thành 1996.

```
name = ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
print(name)
# ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
name[1] = 1996
print(name)
# ['Vu Thanh Tai', 1996, 'Nguyen Thi E']
```

Delete

Để xóa một hoặc nhiều phần tử trong list thì các bạn cần truy cập đến phần tử cần xóa và dùng hàm del để xóa. Và sau khi chúng ta xóa phần tử trong list thì index của list sẽ được cập nhật lại.

VD: Xóa phần tử thứ 3 trong list trên.

```
name = ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
print(name)
# ['Vu Thanh Tai', 'Nguyen Van A', 'Nguyen Thi E']
del name[2]
print(name)
# ['Vu Thanh Tai', 'Nguyen Van A']
```

3.4. List lồng nhau.

Do list có thể chứa nhiều kiểu dữ liệu khác nhau lên chúng ta hoàn toàn có thể khai báo một list chứa một hoặc nhiều list khác nhau.

VD:

```
option = [12,5,1996]
myList = ['Vu Thanh Tai', option]
print(myList)
# ['Vu Thanh Tai', [12, 5, 1996]]
```

Và cứ như thế chúng ta có thể lồng N list khác vào trong list.

Đối với list lồng nhau như này thì chúng ta chũng truy xuất đến các phần tử như bình thường, theo cấp từ ngoài vào trong.

VD: Mình sẽ truy cập vào phần tử dầu tiên trong list option.

```
option = [12,5,1996]
myList = ['Vu Thanh Tai', option]
print(myList)
# ['Vu Thanh Tai', [12, 5, 1996]]
subList = myList[1] # [12, 5, 1996]
subList[0] # 12
# hoặc có thể viết ngắn gọn như sau
myList[1][0] # 12
```

4. Các hàm xử lý list trong Python

Phần trước đã giới thiệu với mọi người các hàm xử lý chuỗi hay dùng trong Python, phần này chúng ta tiếp tục chuyển sang tìm hiểu về một số hàm xử lý list hay được sử dụng trong Python

4.1, list().

Hàm này có tác dụng chuyển đổi kiểu dữ liệu của một biến sang dạng list.

Cú pháp: `list(data)`

Trong đó, data là biến chứa tuple bạn cần chuyển đổi.

VD:

```
string = "Vu Thanh Tai"

print(list(string))

# Ket Qua: ['V', 'u', ' ', 'T', 'h', 'a', 'n', 'h', ' ',
'T', 'a', 'i']

tup = ('A', 'B', 'C')

print(list(tup))

# Ket Qua: ['A', 'B', 'C']
```

4.2, len().

Hàm này trả về số lượng phần tử có trong list.

Cú pháp: `len(list)`

Trong đó, list là list mà các bạn cần đếm.

VD:

```
list = ['A', 'B', 'C']

print(len(list))

#Kết quả: 3
```

4.3, max().

Hàm này sẽ trả về phần tử có giá trị lớn nhất trong list. Nếu là chuỗi thì nó sẽ trả về phần tử có độ dài chuỗi dài nhất, nếu là số thì nó sẽ trả về phần tử có số lớn nhất.

Cú pháp: `max(list)`

Trong đó, list là list mà các bạn cần kiểm tra.

VD:

```
list = ['A', 'B', 'C']
print(max(list))
#Kết quả: C
list = ['1', '3', '2']
print(max(list))
#Kết quả: 3
```

4.4, min().

Hàm này sẽ trả về phần tử có giá trị nhỏ nhất trong list. Nếu là chuỗi thì nó sẽ trả về phần tử có độ dài chuỗi ngắn nhất, nếu là số thì nó sẽ trả về phần tử có số nhỏ nhất.

Cú pháp: min(list)

Trong đó, list là list mà các bạn cần kiểm tra.

VD:

```
list = ['A', 'B', 'C']
print(min(list))
#Kết quả: A
list = ['1', '3', '2']
print(max(list))
#Kết quả: 1
```

4.5, append().

Phương thức này có tác dụng thêm phần vào cuối của một list.

Cú pháp: mylist.append(obj)

Trong đó:

mylist là list mà các bạn cần thêm phần tử.

obj là phần tử mà bạn muốn thêm vào mylist.

VD:

```
list = ['A', 'B', 'C']
list.append('D')
print(list)
```

```
# Kết quả: ['A', 'B', 'C', 'D']
list.append(('E', 'F'))
print(list)
# Kết quả: ['A', 'B', 'C', 'D', ('E', 'F')]
```

4.6, extend().

Hàm này có tác dụng kế thừa lại các phần tử của list2 và thêm vào trong list1.

Cú pháp:

list1.extend(list2)

Trong đó:

list1 là list mà bạn muốn kế thừa từ một list khác (ở đây là list2).

list2 là list được sử dụng để cho list khác kết thừa (ở đây là list1).

VD:

```
list = ['A', 'B', 'C']
list.extend('D')
print(list)
# Kết quả: ['A', 'B', 'C', 'D']
list.extend(('E', 'F'))
print(list)
# Kết quả: ['A', 'B', 'C', 'D', 'E', 'F']
```

Lưu ý: Ở đây mình đã cố tình để ví dụ của phương thức append() và extend() là giống nhau để cho các bạn thấy được sự khác biệt giữa 2 phương thức này).

4.7, count().

Phương thức này có tác dụng đếm số lần xuất hiện của một thành phần trong list!

Cú pháp:

```
mylist.count(val)
```

Trong đó:

mylist là list mà các bạn cần kiểm tra.

val là phần tử mà bạn muốn tìm và đếm trong list mylist.

VD:

```
list = ['A', 'B', 'C']
print(list.count('A'))
# Kết quả: 1
```

4.8, index().

Phương thức này có tác dụng trả về index xuất hiện đầu tiên của phần tử mà bạn muốn tìm và nếu như không tìm thấy thì nó sẽ gọi exception.

Cú Pháp: `mylist.index(val)`

Trong đó:

mylist là list mà các bạn cần kiểm tra.

val là phần tử mà bạn muốn tìm trong list mylist.

VD:

```
list = ['A', 'B', 'C']
print(list.index('B'))
# Kết quả: 1
print(list.index('D'))
# Kết quả: ValueError: 'D' is not in list
```

4.9, insert().

Phương thức có tác dụng thêm phần tử vào vị trí index của list, và các phần tử sau index đó sẽ được đẩy về phía sau.

Cú pháp: `mylist.insert(index, val)`

Trong đó:

mylist là list mà các bạn cần thêm.

index là vị trí mà bạn muốn thêm phần tử val vào.

val là phần tử mà bạn muốn thêm vào trong list mylist.

VD:

```
list = ['A', 'B', 'C']
list.insert(0, 'Z')
print(list)
# Kết quả: ['Z', 'A', 'B', 'C']
list.insert(2, 'D')
```

```
print(list)
# Kết quả: ['Z', 'A', 'D', 'B', 'C']
```

4.10, reverse().

Phương thức này có tác dụng đảo ngược vị trí của các phần tử trong list.

Cú pháp:

mylist.reverse()

Trong đó, mylist là list mà các bạn muốn đảo ngược.

VD:

```
list = ['A', 'B', 'C']
list.reverse()
print(list)
# Kết quả: ['C', 'B', 'A']
```

4.11, remove().

Phương thức này có tác dụng xóa phần tử khỏi list.

Cú Pháp: `mylist.remove(val)`

Trong đó:

mylist là list mà các bạn cần xóa phần tử.

val là phần tử mà bạn muốn muốn xóa trong list mylist.

VD:

```
list = ['A', 'B', 'C']
list.remove('C')
print(list)
# Kết quả: ['A', 'B']
```

4.12, pop().

Phương thức này có tác dụng xóa bỏ phần tử trong list dựa trên index của nó.

Cú pháp: `mylist.pop(index)`

Trong đó:

mylist là list mà các bạn cần xóa phần tử.

index là index của phần tử mà bạn muốn muốn xóa trong list mylist. Mặc định thì index = mylist[-1] (phần tử cuối cùng trong list).

VD:

```
list = ['A', 'B', 'C', 'D', 'E']

list.pop()

print(list)

# Kết quả: ['A', 'B', 'C', 'D']

list.pop(2)

print(list)

# Kết quả: ['A', 'B', 'D']
```

4.13, sort().

Phương thức này có tác dụng sắp xếp lại các phần tử trong list theo một thứ tự xác định.

Cú pháp: `mylist.sort(reverse, key)`

Trong đó:

mylist là list mà các bạn muốn sắp xếp.

reverse là một boolean cấu hình kiểu sắp xếp. Nếu reverse = True thì list sẽ được sắp xếp từ lớn đến bé, nếu reverse = False thì list sẽ được sắp xếp theo thứ tự từ bé đến lớn. Mặc định thì reverse = False.

key là callback def để xử lý list hoặc là một lamda function (thường được dùng để sắp xếp các list tuple hoặc dictionary).

VD:

```
list = ['A', 'C', 'B', 'E', 'D']

list.sort()

print(list)

# Kết quả: ['A', 'B', 'C', 'D', 'E']

list.sort(reverse=True)

print(list)

# Kết quả: ['E', 'D', 'C', 'B', 'A']

def custom_sort(elem):

    return elem[1]
```

```
list = [(1, 2), (5, 7), (7, 100), (4, 4)]
list.sort(key=custom_sort)
print(list)
# Kết quả: [(1, 2), (4, 4), (5, 7), (7, 100)]
```

4.14, clear().

Phương thức này có tác dụng xóa bỏ hết tất cả các phần tử trong list

Cú pháp: `mylist.clear()`

Trong đó, mylist là list mà bạn muốn xóa bỏ hết phần tử.

VD:

```
list = ['A', 'C', 'B', 'E', 'D']
list.clear()
print(list)
# Kết quả: []
```

5. Chuỗi (String) trong Python

Kiểu dữ liệu chuỗi (String) trong Python là một trong các kiểu phổ biến nhất trong Python. Chuỗi ký tự trong python được bao quanh bởi dấu ngoặc kép đơn hoặc dấu ngoặc kép. Python coi các lệnh trích dẫn đơn và kép là như nhau. Ví dụ: 'Hello' tương đương với "Hello".

Bạn có thể hiển thị một chuỗi trong Python bằng print(). Ví dụ:

```
print("Hello")
print('Hello')
```

Gán chuỗi cho một biến

Việc gán một chuỗi cho một biến được thực hiện với tên biến theo sau là dấu bằng và chuỗi, Ví dụ:

```
str1 = "Hello World!"
print(str1)
```

Chuỗi đa dòng

Bạn có thể gán một chuỗi nhiều dòng cho một biến bằng cách sử dụng 3 dấu ngoặc kép hoặc 3 dấu nháy đơn:

Ví dụ nhập chuỗi đa dòng với 3 dấu ngoặc kép:

```
str1 = """Vi du nhap chuoi nhieu dong trong Python
day la dong thu 2
day la dong thu 3
day la dong thu 4"""
print(str1)
```

Ví dụ nhập chuỗi đa dòng với 3 dấu nháy đơn:

```
str1 = '''Vi du nhap chuoi nhieu dong trong Python
day la dong thu 2
day la dong thu 3
day la dong thu 4'''
print(str1)
```

Lưu ý: các ngắt dòng phải giống nhau (thụt dầu dòng giống nhau). Đó là quy tắc cơ bản trong Python.

Chuỗi là một mảng

Các chuỗi trong Python là mảng các byte đại diện cho các ký tự unicode.

Tuy nhiên, Python không có kiểu dữ liệu ký tự, một ký tự đơn giản chỉ là một chuỗi có độ dài bằng 1.

Dấu ngoặc vuông [] có thể được sử dụng để truy cập các phần tử của chuỗi. Ký tự đầu tiên có chỉ số là 0.

```
str1 = "Hello World!"
print(str1[0])
```

Kết quả:

H

Truy cập các giá trị trong String

Dấu ngoặc vuông [] có thể được sử dụng để truy cập các phần tử của chuỗi. Ký tự đầu tiên có chỉ số là 0.

Ví dụ 1:

```
str1 = "HELLO"
str1 = "HELLO"
print(str1[0])
print(str1[1])
print(str1[2])
print(str1[3])
print(str1[4])
```

Kết quả: trả về một chuỗi con từ vị trí 6 đến 8 của chuỗi đã cho:

H

E

L

L

O

Chỉ định chỉ mục bắt đầu và chỉ mục kết thúc, được phân tách bằng dấu hai chấm, để trả về một phần của chuỗi.

Ví dụ 2:

```
str1 = "HELLO"
print(str1[:])
print(str1[0:])
print(str1[:5])
print(str1[:3])
print(str1[0:2])
print(str1[1:4])
```

Kết quả:

HELLO

HELLO

HELLO

HEL

HE

ELL

Truy cập chuỗi bằng chỉ mục âm

Sử dụng các chỉ mục âm để lấy ra chuỗi con bắt đầu từ cuối chuỗi: Ví dụ:

```
str1 = "Hello World!"
print(str1[-5:-2])
```

Kết quả: trả về một chuỗi con từ vị trí 3 đến 5 từ từ cuối chuỗi của chuỗi đã cho:

orl

6. Các hàm xử lý chuỗi trong Python

Ở phần đầu của series mình đã giới thiệu với mọi người về chuỗi trong Python rồi, nhưng Python là một ngôn ngữ khá là linh động và mềm dẻo nên nó cũng đã cung cấp cho chúng ta rất nhiều hàm có sẵn dùng để xử lý chuỗi. Bài viết này mình sẽ liệt kê một số hàm hay dùng và ví dụ kèm theo cho mọi người cùng tham khảo.

1, Count().

Hàm này có tác dụng đếm xem trong chuỗi có bao nhiêu ký tự cần tìm.

Cú Pháp:

string.count(sub, start, end)

Trong đó:

sub là chuỗi các bạn cần tìm kiếm và đếm.

start là index bắt của chuỗi cần tìm. Mặc định thì strart = 0.

end là index kết thúc của chuỗi cần tìm. Mặc định thì end = len() của chuỗi.

VD:

```
string = "toidicode.com"
print(string.count('i'));
# Kết quả: 2
print(string.count('i', 3));
# Kết quả: 1
```

2, find().

Hàm này có tác dụng tìm kiếm một chuỗi trong một chuỗi hoặc khoảng chuỗi. Nó sẽ trả về là vị trí bắt đầu của chuỗi tìm được trong chuỗi nếu tìm thấy và nếu không tìm thấy nó sẽ trả về -1.

Cú pháp:

string.find(str, start, end)

Trong đó:

str là chuỗi các bạn cần xác thực xem có phải chuỗi kết thúc không.

strart là index bắt đầu chuỗi cần so sánh. Mặc định thì start = 0.

end là index kết thúc chuỗi cần so sánh. Mặc định thì end = len().

VD:

string = 'toidicode.com'

print(string.find('di'));

Kết quả: 3

3, isalnum().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các ký tự chữ hoặc chuỗi hay không? Nó sẽ trả về True nếu chuỗi chỉ chứa các ký tự chữ hoặc số. Và ngược lại nó sẽ trả về False nếu chuỗi chứa ký tự khác chuỗi và số.

VD:

string = 'toidicode'

print(string.isalnum());

Kết quả: True

string = 'toidicode.com'

print(string.isalnum());

Kết quả: False

4, isalpha().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các ký tự chữ hay không? Nó sẽ trả về True nếu chuỗi này chỉ chứa duy các ký tự chữ trong bảng chữ cái, và sẽ trả về False nếu nó chứa số hoặc ký tự đặc biệt khác.

VD:

string = 'toidicode96'

print(string.isalpha());

Kết quả: False

string = 'toidicodecom'

print(string.isalpha());

Kết quả: True

5, isdigit().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải là chứa duy nhất các chữ số hay không? Nó sẽ trả về True nếu đúng và False nếu sai.

VD:

string = 'toidicode96'

print(string.isdigit());

Kết quả: False

string = '12051996'

print(string.isdigit());

Kết quả: True

6, islower().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải là in thường hay không? Nó sẽ trả về True nếu đúng và False nếu sai.

VD:

string = 'toidicode.com'

print(string.islower());

Kết quả: True

string = '12051996'

print(string.islower());

Kết quả: False

string = '9toidicode.com6'

print(string.islower());

Kết quả: True

string = '9Toidicode.com6'

print(string.islower());

Kết quả: False

7, isupper().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải là in Hoa hay không? Nó sẽ trả về True nếu đúng và False nếu sai.

VD:

string = 'TOIDICODE.COM'

print(string.isupper());

Kết quả: True

string = '12051996'

print(string.isupper());

Kết quả: False

string = '9TOIDICODE.COM6'

print(string.isupper());

Kết quả: True

string = '9Toidicode.com6'

print(string.isupper());

Kết quả: False

8, isspace().

Hàm này có tác dụng kiểm tra xem một chuỗi có phải chỉ chứa duy nhất các ký tự khoảng trắng không? Nó sẽ trả về True nếu đúng và False nếu sai.

VD:

string = ' '

print (string.isspace());

Kết quả: True

string = 'Vu Thanh Tai'

print (string.isspace());

Kết quả: False

9, len().

Hàm này có tác dụng trả về độ dài của chuỗi.

VD:

string = "Vu Thanh Tai"

print(len(string))

Kết quả: 12

10, ljust().

Hàm này có tác dụng trả về một chuỗi với độ dài length được xác định, nếu chuỗi được chọn nhỏ hơn width thì nó sẽ sử dụng char để bù chỗ thiếu đó về phía bên phải của chuỗi.

string.ljust(length, char)

Trong đó:

length là độ dài của chuỗi mới cần in ra.

char là ký tự sẽ bù vào chuỗi mới nếu chuỗi cũ không đủ length. Mặc định thì char = khoảng trắng.

VD:

string = "Vu Thanh Tai"

print(string.ljust(17, "-"))

Kết quả: Vu Thanh Tai-----

11, rjust().

Tương tự hàm ljust() nhưng chỉ có điều là nó sẽ bù về phía bên trái của chuỗi.

VD:

string = "Vu Thanh Tai"

print(string.rjust(17, "-"))

Kết quả: -----Vu Thanh Tai

12, lower().

Hàm này có tác dụng chuyển đổi chuỗi về dạng in thường.

VD:

string = "Vu Thanh Tai"

print(string.lower())

Kết quả: vu thanh tai

13, upper().

Hàm này có tác dụng chuyển đổi chuỗi sang dạng in hoa.

VD:

string = "Vu Thanh Tai"

print(string.upper())

Kết quả: VU THANH TAI

14, lstrip().

Hàm này có tác dụng loại bỏ đi các ký tự char ở phía đầu của chuỗi.

Cú Pháp:

string.lstrip(char)

Trong đó: char là ký tự bạn muốn loại bỏ. Mặc định thì char sẽ bằng khoảng trắng (white space).

VD:

string = " Vu Thanh Tai"

print(string.lstrip())

Kết quả: Vu Thanh Tai

string = "----Vu Thanh Tai"

print(string.lstrip('-'))

Kết quả: Vu Thanh Tai

15, rstrip().

Tương tự như lstrip(), chỉ khác là rstrip nó sẽ loại bỏ char ở phần cuối của chuỗi.

VD:

string = "Vu Thanh Tai   "

print(string.rstrip())

Kết quả: Vu Thanh Tai

string = "Vu Thanh Tai----"

print(string.rstrip('-'))

Kết quả: Vu Thanh Tai

16, strip().

Hàm này là sự kết hợp của lstrip() và rstrip(). Nó sẽ lại bỏ các ký tự char ở cả hai đầu của chuỗi.

VD:

string = "   Vu Thanh Tai   "

print(string.strip())

Kết quả: Vu Thanh Tai

```
string = "----Vu Thanh Tai----"
print(string.strip('-'))
# Kết quả: Vu Thanh Tai
```

17, replace().

Hàm này có tác dụng tìm kiếm và thay thế chuỗi tìm được bằng chuỗi mới.

Cú Pháp:

string.replace(old,new,max)

Trong đó:

old là chuỗi mà bạn cần tìm kiếm trong string.

new là chuỗi mà bạn cần thay thế cho chuỗi old tìm được.

max là số lượng từ có thể thay thế tối đa.

VD:

```
string = "Chao *!"
print(string.replace('*', 'Tai'))
# Kết quả: Chao Tai!
string = "A A A"
print(string.replace('A', 'Tai', 2))
# Kết quả: Tai Tai A
```

18, title().

Hàm này có tác dụng chuyển đổi chuỗi sang kiểu title

Cú Pháp:

string.title()

VD:

```
string = "vu thanh tai"
print(string.title())
# Kết quả: Vu Thanh Tai
```

Bài toán cơ bản:

Bài 1: Cho dãy số A gồm N số nguyên A1, A2, ..., AN. Hãy tìm giá trị lớn nhất của dãy số A và chỉ số các phần tử có giá trị là lớn nhất.

Dữ liệu vào: Gồm 2 dòng:

- Dòng đầu tiên chứa số N ($N<=10^6$).

- Dòng thứ hai chứa dãy số A, các số ghi cách nhau ít nhất là một ký tự trống ($1<=a[i]<=10^9$).

Dữ liệu ra: Gồm 2 dòng:

- Dòng đầu tiên chứa một số nguyên là giá trị lớn nhất của dãy A.

- Dòng thứ hai chứa dãy số là chỉ số các phần tử có giá trị là lớn nhất, các số ghi cách nhau ít nhất là một ký tự trống.

Ví dụ:

Input	Output
10 1 2 5 5 8 3 8 8 2 2	8 5 7 8

Ý tưởng giải quyết bài toán:

```
n=int(input())
a=list(map(int,input().split()))
m = a[0]
vt=0
for i in range(1,n):
    if a[i]> m:
        m=a[i]
        vt=i;
print(m)
for i in range(0,n):
    if a[i]==m :
        print(i+1, end=' ')
```

Bài 2: Để chuẩn bị cho kỳ thi nghề THPT sắp tới, cô đã cho các nhóm đăng ký dự thi và gửi danh sách đăng ký cho bạn An lớp trưởng tổng hợp thành danh sách đăng ký của lớp. Do các nhóm làm việc cẩu thả nên danh sách gửi cho An còn rất nhiều lỗi chính tả: Giữa các từ còn gõ nhiều dấu cách, họ tên học sinh chưa viết hoa đầu từ.

Em hãy viết chương trình: nhập vào xâu s là họ tên của một học sinh, xuất ra xâu đã xóa các dấu cách thừa và viết hoa đầu từ.

Ví dụ:

Input (dấu – đại diện khoảng trắng)	Output

--tran---van----an---	Tran-Van-An

Ý tưởng giải quyết bài toán:

```
# Nhập xâu st
st=input()

# Xóa kí tự trắng đầu và cuối xâu
st=st.strip()

# Thay thế tất cả các chuỗi 2 khoảng trắng thành kí tự rỗng
while st.find("  ") !=-1 :
    st = st.replace("  "," ")

# In hoa kí tự đầu của từ
print(st.title())
```

6) Kiểu dữ liệu tệp:

Ví dụ bài toán tìm max ở trên chúng ta vào ra bằng file như sau:

TIMMAX.INP	TIMMAX.OUT
10 1 2 5 5 8 3 8 8 2 2	8 5 7 8

Chương trình được cập nhật như sau:

```
import os, sys
# Khai báo đọc file trong thư mục đặt file py
f = open(os.path.join(sys.path[0], "TIMMAX.INP"), "r")

# Khai báo ghi file trong thư mục đặt file py
g = open(os.path.join(sys.path[0], "TIMMAX.OUT"), "w")

n=int(f.readline())
a=list(map(int,f.readline().split()))
m = a[0]
vt=0
for i in range(1,n):
    if a[i]> m:
        m=a[i]
        vt=i;
g.write(str(m))
g.write("\n")
for i in range(0,n):
    if a[i]==m :
        g.write(str(i+1))
        g.write(" ")
```

Bài tập cơ bản:

Bài 1: Cho tệp văn bản TEPXAU.INP ghi họ tên học sinh trong một lớp học. Hãy cho biết trong tệp có bao nhiêu học sinh và đưa ra họ tên của học sinh dài nhất.

Trường hợp 1: Dữ liệu vào: Tệp TEPXAU.INP

Gồm nhiều dòng, mỗi dòng là họ tên của 1 học sinh.

Trường hợp 2: Dữ liệu vào: Tệp TEPXAU.INP.

- Dòng đầu tiên chứa số nguyên n là số học sinh trong lớp.
- n dòng tiếp theo mỗi dòng là họ tên của 1 học sinh.

Kết quả ra: Ghi vào tệp TEPXAU.OUT gồm 2 dòng:

- Dòng đầu tiên ghi độ dài tên học sinh dài nhất
- Dòng thứ hai ghi họ tên của học sinh dài nhất.

Ví dụ

TEPXAU.INP	TEPXAU.OUT
3 Nguyen Van An Tran Thi B Le Van C	13 Nguyen Van An

Ý tưởng giải quyết:

```
1   import os, sys
2   f = open(os.path.join(sys.path[0], "TEPXAU.INP"), "r")
3   g = open(os.path.join(sys.path[0], "TEPXAU.OUT"), "w")
4
5   n = int(f.readline())
6   lmax = 0
7   for i in range(0,n):
8       name = f.readline()
9
10      if len(name)>lmax:
11          lmax = len(name)
12          lname = name
13  g.write(str(lmax-1) +"\n" + lname)
```

7) Chương trình con và lập trình có cấu trúc.

- Chương trình con đóng vai trò quan trọng trong lập trình, đặc biệt trong lập trình có cấu trúc.

- Có 2 loại hàm đó là hàm có giá trị trả về và hàm không có giá trị trả về.
- Biến toàn cục, biến cục bộ.
- Tham số hình thức, tham số thực sự;
- Tham trị, tham biến.

Bài toán cơ bản:

Cho dãy số A gồm N số nguyên A1, A2, ..., AN. Hãy cho biết có bao nhiêu số trong dãy số A có tổng các chữ số là một số chính phương.

Dữ liệu vào: Tệp CTC.INP gồm 2 dòng:

- Dòng đầu tiên chứa số N ($N<=10^4$).
- Dòng thứ hai chứa dãy số A, các số ghi cách nhau ít nhất là một ký tự trống ($1<=a[i]<=10^9$).

Dữ liệu ra: Tệp CTC.OUT chứa 1 số nguyên duy nhất là kết quả của bài toán.

```
from math import sqrt
import os, sys
def tongcs(a):
    t=0
    while a!=0:
        t=t+a%10
        a=a/10
    return t
def cphuong(a):
    t=sqrt(a)
    if t*t==a:
        return True
    else:
        return False
#chuong trinh chinh
# Khai báo đọc file trong thư mục đặt file py
f = open(os.path.join(sys.path[0], "FUNCTION.INP"), "r")
# Khai báo ghi file trong thư mục đặt file py
g = open(os.path.join(sys.path[0], "FUNCTION.OUT"), "w")
n=int(f.readline())
a=list(map(int,f.readline().split()))
dem=0
for i in range(0,n):
    if cphuong(tongcs(a[i]))==True : dem=dem+1
g.write(str(dem))
```

Nguồn tham khảo: https://toidicode.com/

CHUYÊN ĐỀ 2: SỐ HỌC

1) Phép toán chia lấy dư và nguyên lý đồng dư

Trong các bài toán tin học các yêu cầu xử lý cần dùng phép toán chia lấy dư (%) rất phổ biến. Rất nhiều bài toán cần đến xử lí số lớn, nhưng khi ý đồ của tác giả chỉ là tìm ra thuật toán đúng để giải quyết bài toán chứ không coi trọng việc cài đặt số lớn, lúc đó thường thì đề ra sẽ yêu cầu ghi ra số dư của kết quả khi chia cho 1 cơ số bất kì nào đó. Vì vậy mà ta cần tìm hiểu các tính chất đồng dư để có thể đảm bảo kết quả mình đưa ra là chính xác.

Giả sử có 2 số nguyên x và y, với cơ số là n, có những tính chất đồng dư sau đây:

1. (x+y)% n = ((x % n)+(y % n)) % n.

2. (x*y)% n =((x % n)*(y % n))%n.

Bài toán cơ bản. **Tổng lập phương**

Cho số tự nhiên n. Hãy viết chương trình tính S = $1^3 + 2^3 + 3^3 + \ldots + n^3$.

Dữ liệu vào:

Đọc từ tệp TLP.Inp gồm một dòng duy nhất chứa số nguyên n ($1<=n<=10^9$).

Dữ liệu ra:

Ghi kết quả ra tệp TLP.Out một giá trị duy nhất lài số dư của phép chia S cho 10^9+7

Ví dụ:

TLP.inp	TLP.out
3	36
4	100

Ràng buộc: 70% test ứng với giới hạn $1<=n<=10^4$ 30% test ứng với các trường hợp còn lạiÝ tưởng:

Cách 1: Sử dụng vòng lặp duyệt các số i từ 1 đến n để tính tổng lập phương.

```
S=0;
for (int i=1; i<=n; i++) S = s+ i*i*i;
S= S%(10^9+7);
```

Với cách làm này độ phức tạp là O(n), nếu sử dụng kiểu dữ liệu phù hợp thì ta cũng chỉ mới ăn được các test nhỏ chưa giải quyết được với mọi giá trị. Vì vậy cần tiếp tục cải tiến.

Cách 2: Sử dụng công thức toán học:

Ta có thể chuyển độ phức tạp xuống O(1), nhưng lưu ý rằng giá trị của S là rất lớn, không có kiểu dữ liệu nguyên nào có thể lưu lại giá trị kết quả. Vì vậy ta cần vận dụng tính chất đồng dư để giải quyết bài toán.

Khi đó ta có câu lệnh tính:

$$1^3+2^3+3^3+\cdots+n^3 = n^2.(n+1)^2/4$$

Khai báo hằng e: e= 10^9 +7.

S = ((n%e)*(n%e))%e)*((n+1)%e)*((n+1)%e)%e)/4;

Chương trình tham khảo:

TLP.cpp

```
#include <bits/stdc++.h>
using namespace std;
const int e = 1000000007;
long long n, S;

int main()
{   freopen("TLP.inp","r", stdin);
    freopen("TLP.out","w", stdout);
     cin>>n;
     S = (((n%e)*(n%e))%e)*(((n+1)%e)*((n+1)%e)%e)/4;
     cout<<S;
    return 0;
 }
```

Trong Python mã tham khảo của chương trình sẽ như sau:

```
fi=open("TLP.INP","r")
fo=open("TLP.OUT","w")
e=1000000007

N=int(fi.read())

S=(((N%e)*(N%e))%e)*(((N+1)%e)*((N+1)%e)%e)/4

fo.write(format(S))

fo.close()
fi.close()
```

2) Số hoàn hảo

Số hoàn hảo (hay còn gọi là số hoàn chỉnh, số hoàn thiện) là một số nguyên dương mà tổng các ước nguyên dương chính thức của nó (số nguyên dương bị nó chia hết ngoại trừ nó) bằng chính nó.

Các số sau là số hoàn hảo:

Số 6 (1 + 2 + 3 = 6)

Số 28 (1 + 2 + 4 + 7 + 14 = 28)

Bài toán cơ bản:

Cho số tự nhiên N. Hãy cho biết N có phải là số hoàn hảo hay không?

Dữ liệu vào: Đọc từ tệp SHH.Inp gồm một dòng duy nhất chứa số nguyên N ($1<=n<=10^9$).

Dữ liệu ra: Ghi kết quả ra tệp SHH.Out số 1 nếu N là số hoàn hảo, ngược lại thì ghi số 0.

Ví dụ

SNT.inp	SNT.out
28	1
100	0

Phân tích bài toán:

Để kiểm tra số nguyên dương N(N>1) có là số hoàn hảo không, ta tính tổng S tổng các số nguyên i (1<=i<=N-1) mà i là ước của N. Sau đó ta kiểm tra S có bằng N hay không. Nếu S=N thì N là số hoàn hảo, ngược lại N không phải là số hoàn hảo.

Xét các số i với $i<=\sqrt{N}$, ta thấy nếu i là ước của N thì thương N/i cũng là ước của N. Do đó, việc tìm ước với k từ 1 đến N- 1 là không cần thiết, mà chỉ cần kiểm tra k từ 1 đến $\sqrt{N}$. Khi đó, ta có chương trình:

art here | SHH.cpp

```
#include <bits/stdc++.h>
using namespace std;
int N;
/*************************/
bool SHH(int N)
{    int s=1;
     int t = sqrt(N);
     for (int i=2; i<=t; i++)
     if (N%i==0)
          s = s + i + N/i;
     if (S==N) return true;
     return false ;
}
/*************************/
int main()
{
 freopen("SNT.inp","r",stdin);
 freopen("SNT.out","w",stdout);
   cin>>N;
   cout<<SHH(N);
   return 0;
 }
```

Code với Python sẽ là:

```
import math
fi=open("SHH.INP","r")
fo=open("SHH.OUT","w")

N=int(fi.read())

def SHHao(N):
    S=1
    t=int(math.sqrt(N))+1
    for i in range(2,t):
        if N%i==0: S=S+i+N//i
    if S==N: return 1
    else:return 0


fo.write(format(SHHao(N)))

fo.close()
fi.close()
```

Lưu ý: Trong hàm kiểm tra tính hoàn hảo nói trên. Nếu N là số chính phương thì việc tính tổng các ước thực sự của N là chưa đúng. Vì trong trường hợp này $i = \sqrt{N}$ là ước thì N/i cũng là $\sqrt{N}$. vì vậy tổng đã cộng 2 lần $\sqrt{N}$. Ở đây ta thấy số chính phương không bao giờ là số hoàn hảo nên tổng không phải trừ $\sqrt{N}$ kết quả vẫn đúng.

3) Số Fibonaxi.

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1 hoặc 1 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó. Công thức truy hồi của dãy Fibonacci là:

$$f(n)=\begin{cases}1, & kh\ in=1\\ 1, & kh\ in=2\\ f(n-1)+f, & kh\ in>2\end{cases}$$

Lưu ý: *Giá trị của số Fibonaxi thứ n tăng rất nhanh khi n tăng. Vì vậy đối với các bài toán có liên quan đến số Fibonaxi ta cần đặc biệt lưu ý về miền dữ liệu đầu vào của bài toán để sử dụng cấu trúc dữ liệu và phương án xử lý bài toán được tốt nhất.*

Bài toán cơ bản về dãy số Fibonaxi:

Dãy số fibonacci được thành lập bởi công thức:

$f_1 = f_2 = 1$;

$f_i = f_{i-1} + f_{i-2}$ ($i \geq 3$).

Yêu cầu: Cho số nguyên dương n (n<=50). Tính giá trị Fn Input: Gồm một dòng duy nhất chứa số nguyên dương n. Output: In ra kết quả bài toán.

Ví dụ:

Fibo1.INP	Fibo1.OUT
10	55

Phân tích bài toán:

Để tính Fn ta phải tính các giá trị đứng trước của dãy F1, f2,…, fN-1. Công thức lặp Fi=Fi-1 + Fi-2 với i lần lượt nhận giá trị từ 3 đến n.

Để tính giá trị tiếp theo của dãy thì giá trị trước giá trị hiện tại lại là giá trị trước thứ 2. Giá trị hiện tại lại là giá trị trước giá trị cần tính. Vì vậy với mỗi giá trị mới của i ta lại phải xác định lại giá trị trước và trước thứ 2 của i mới.

Giá trị của số fibonaxi tăng rất nhanh khi n tăng, nên ta cần phải để ý sử dụng biến với kiểu dữ liệu phù hợp.

Cách 1: Sử dụng câu lệnh lặp để tính các giá trị từ F3 tới Fn.

Cách 2: Sử dụng mảng để lưu các giá trị của dãy.

Cách 3: Dùng đệ quy

Chương trình được thể hiện như sau:

Cách 1

```
#include <bits/stdc++.h>
using namespace std;
int n;
long long f, f1, f2;
int main()
{freopen("fibo1.inp","r",stdin);
 freopen("fibo1.out","w",stdout);
     cin>>n;
     f1=f2=1;
     for(int i=3;i<=n;i++)
    {   f=f1+f2;
        f1=f2;
        f2=f;
     }
    cout<<f;
   return 0;
}
```

Cách 2

```
#include <bits/stdc++.h>
using namespace std;
int n;
long long f, a[52];
int main()
{freopen("fibo1.inp","r",stdin);
 freopen("fibo1.out","w",stdout);
    cin>>n;
    a[1]=a[2]=1;
    for(int i=3; i<=n; i++)
        a[i]=a[i-1]+a[i-2];
    cout<<a[n];
   return 0;
}
```

Cách 3

```
#include <bits/stdc++.h>
using namespace std;
int n;
long long f;
long long Fibo(int n)
{    if (n==1 || n==2)
      return 1;
    else
      return Fibo(n-1)+Fibo(n-2);
}
int main()
{freopen("fibo1.inp","r",stdin);
 freopen("fibo1.out","w",stdout);
    cin>>n;
    cout<<Fibo(n);
   return 0;
}
```

Code với Python:

```
fi=open("Fibo1.INP","r")
fo=open("Fibo1.OUT","w")

N=int(fi.read())

def Fibo(N):
    if (N==1 or N==2):
        return 1
    else: return Fibo(N-1)+Fibo(N-2)

fo.write(format(Fibo(N)))

fo.close()
fi.close()
```

Qua bài toán này, ta thấy mỗi thuật toán cần cách thức tổ chức dữ liệu khác nhau. Mỗi cách giải đều có ưu và nhược điểm riêng, trong miền dữ liệu của bài toán trên thì cả 3 cách đều giải quyết tốt. Trong thực tế dạy học ở chủ đề này thì học sinh chưa được học về mảng nên chỉ mới đủ kiến thức để giải quyết bằng cách

1. Ta có thể đề cập lại bài toán trong phát triển bài toán, mở rộng yêu cầu bài toán thành các bài toán sau.

Một số yêu cầu mở rộng về dãy Fibonaxi:

Dãy số fibonacci được thành lập bởi công thức:

$f_1 = f_2 = 1$;

$f_i = f_{i-1} + f_{i-2}$ ($i \geq 3$).

a/ Tính tổng n số đầu tiên của dãy

b/ Đưa ra danh sách n số của dãy và.

c/ Đưa ra danh sách n số lẻ đầu tiên của dãy.

d/ kiểm tra số n có thuộc dãy fibonacci hay không e/ Tính số Fibonaxi thứ n %(10^9+7) với (n<=10^6).

Yêu cầu học sinh chỉnh sửa bài toán gốc để giải quyết các bài toán con.
Giáo viên quan sát và hỗ trợ khi học sinh gặp vướng mắc.

Lưu ý:

- Số fibonaxi thứ n tăng rất nhanh khi n tăng, nên ta cần lưu ý đến việc chọn kiểu dữ liệu đặc biệt là biến lưu tổng.

Nếu sử dụng kiểu dữ liệu int thì chỉ lưu trữ được đến số Fibonxi thứ 43. Kiểu long long thì lưu được đến số thứ 60.

Kiểu unsigned long long ta có thể lưu trữ được số Fibonaxi thứ 93. Với n lớn ta phải có phương pháp xử lý khác.

- Với câu d của bài toán, ta sử dụng tính chất đồng dư để thu gọn giá trị

trong quá trình tính từng số của dãy:

```
f1=f2=1;
for(int i=3;i<=n;i++)
    {  f=(f1+f2)%1000000007;
       f1=f2;       f2=f;      }
```

4) Số nguyên tố

Một số tự nhiên N (N>1) là số nguyên tố nếu N có đúng hai ước số là 1 và N. Ví dụ các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23, …

Tính chất đơn giản sau của số nguyên tố:

+ Trừ số 2 các số nguyên tố còn lại đều là số lẻ.

+ Trừ số 2, số 3 các số nguyên tố có dạng $6k \pm 1$ vì số có có dạng $6k \pm 2$ thì chia hết cho 2, số có dạng $6k \pm 3$thì chia hiêt cho 3.

Bài toán cơ bản:

Cho số tự nhiên N. Hãy cho biết N có phải là số nguyên tố hay không?

Dữ liệu vào: Đọc từ tệp SNT.Inp gồm một dòng duy nhất chứa số nguyên N ($1<=n<=10^9$).

Dữ liệu ra: Ghi kết quả ra tệp SNT.Out số 1 nếu N là số nguyên tố, ngược lại thì ghi số 0. Ví dụ

SNT.inp	SNT.out
13	1
100	0

Phân tích bài toán:

Để kiểm tra số nguyên dương N(N>1) có là số nguyên tố không, ta kiểm tra xem có tồn tại một số nguyên k (2<=i<=N-1) mà i là ước của N (N chia hết

i) thì N không phải là số nguyên tố, ngược lại N là số nguyên tố.

Nếu N(N>1) không phải là số nguyên tố, ta luôn có thể tách $N = i_1 * i_2$ mà ($2 <= i_1 <= i_2 <= N-1$). Vì $i_1 * i_1 <= i_1 * i_2 = N$ nên $i_1 <= \sqrt{N}$. Do đó, việc kiểm tra với i từ 2 đến N-1 là không cần thiết, mà chỉ cần kiểm tra i từ 2 đến $\sqrt{N}$. Khi đó, ta có chương trình:

*SNT.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int N;
/*************************/
bool SNT(int N)
{     int t = sqrt(N);
     for (int i=2; i<=t; i++)
     if (N%i==0) return false ;
     return true;
}
/*************************/
int main()
{
 freopen("SNT.inp","r",stdin);
 freopen("SNT.out","w",stdout);
   cin>>N;
   cout<<SNT(N);
   return 0;
 }
```

SNT1.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int N;
/*************************/
bool SNT(int N)
{   if (N==2||N==3)  return true;
    if (N==1||N%2==0||N%3==0) return false ;
    int t = sqrt(N);
    for (int i=5; i<=t; i+=6)
    if (N%i==0||N%(i+2)==0) return false ;
    return true;
}
/*************************/
int main()
{
 freopen("SNT.inp","r",stdin);
 freopen("SNT.out","w",stdout);
   cin>>N;
   cout<<SNT(N);
   return 0;
 }
```

Code với Python:

```
import math
fi=open("SNT.INP","r")
fo=open("SNT.OUT","w")

N=int(fi.read())

def SNT(N):
    t=int(math.sqrt(N))+1
    for i in range(2,t):
        if N%i==0:
            return False
    return True

fo.write(format(SNT(N)))

fo.close()
fi.close()
```

Dựa vào tính chất: Trừ số 2, số 3 các số nguyên tố đều có dạng $6k \pm 1$. Vậy ta chỉ cần kiểm tra xem N có chia hết cho số 2, số 3 và các số có dạng $6k \pm 1$ trong đoạn $[5, \sqrt{N}]$. Chương trình thể hiện như sau:

5) Sàng nguyên tố

Cho số tự nhiên N. Hãy đưa ra các số nguyên tố thuộc đoạn [1- N];

Dữ liệu vào: Đọc từ tệp SangNT.Inp gồm một dòng duy nhất chứa số nguyên N ($1<=n<=10^7$).

Dữ liệu ra: Ghi kết quả ra tệp SNT.Out số 1 nếu N là số nguyên tố, ngược lại thì ghi số 0. Ví dụ

SangNT.inp	SangNT.out
20	2, 3, 5, 7, 11, 13, 17, 19

Ý tưởng: *Cách 1*: Duyệt vét cạn.

Dùng lệnh for duyệt lần lượt các số i trong đoạn [1, N], kiểm tra tính nguyên tố của i số bằng hàm kiểm tra nguyên tố. Nếu i là số nguyên tố thì xuất i ra.

Cách này đơn giản nhưng chạy chậm để duyệt đến 10^7 là rất mất nhiều thời gian, để cải tiến có thể sử dụng các tính chất của số nguyên tố để loại bỏ trước những số không phải là số nguyên tố và không cần kiểm tra các số này (sử dụng sàng Eratosthene).

Cách 2: Sàng số nguyên tố:

Dùng mảng A với N phần tử để đánh dấu trạng thái nguyên tố. Ban đầu toàn bộ mảng A=1. Ta thấy 1 không phải là nguyên tố nên a[1]=0;

Dùng vòng for duyệt các số i từ 2 cho đến $\sqrt{N}$. Nếu gặp phần tử a[i] có giá trị bằng 1 thì i là nguyên tố, ta tiến hành xóa tất cả các bội của i bằng cách gán toàn bộ

phần tử chia hết cho i giá trị 0

Duyệt cho đến khi gặp số nguyên tố lớn hơn $\sqrt{N}$ thì dừng lại. Tất cả giá trị i có A[i] = 1 đều là số nguyên tố.

Chương trình sử dụng sàng nguyên tố như sau:

there | *SangNT.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n,a[1000002];
void sangnt(int n)
{
    fill(a+1, a+1+n, 1);
    a[1] =0;
    for(int i=2; i*i<n; i++)
        if (a[i]==1)
        for(int j=i*i; j<=n; j+=i)
            a[j]=0;
}
int main()
{
 freopen("Sangnt.inp","r",stdin);
 freopen("Sangnt.out","w",stdout);
    cin>>n;
    sangnt(n);
    for(int i=2;i<=n;i++)
        if (a[i]==1)
            cout<<i<<" ";
    return 0;
}
```

Code với Python:

```python
import math
fi=open("SANGNT.INP","r")
fo=open("SANGNT.OUT","w")

N=int(fi.read())
A=[0,0]

def SANGNT(N):
    for i in range(2,N+1):
        A.append(1)
    t=int(math.sqrt(N))+1
    for i in range(2,t):
        if(A[i]==1):
            j=i*i
            while j<=N:
                A[j]=0
                j=j+i

SANGNT(N)
for i in range(N+1):
    if A[i]==1:
        fo.write(format(i)+' ')

fo.close()
fi.close()
```

:

6. Phân tích thừa số nguyên tố.

Cho số tự nhiên N. Hãy phân tích n thành thừa số nguyên tố; Ví dụ: Số 18 = 2 x 3^2. Số 100 = 2^2 x 5^2

Dữ liệu vào: Đọc từ tệp TSNT. Inp gồm một dòng duy nhất chứa số nguyên N ($1<=n<=10^7$).

Dữ liệu ra: Ghi kết quả ra tệp TSNT.Out mỗi dòng là gồm 2 số nguyên.

Số đầu là thừa số nguyên tố số tiếp theo là số mũ tương ứng với thừa số đó trong phân tích.

Ví dụ:

TSNT.inp	TSNT.out	TSNT.inp	TSNT.out
18	2 1 3 2	100	2 2 5 2

Ý tưởng:

Ta cần khai báo một mảng a để lưu giá trị mỗi lần n chia hết cho số đó.

Tức a[i] sẽ là số lượng thừa số i trong phân tích.

Lần lượt chia **n** cho các **số nguyên tố** nhỏ hơn **n** theo chiều từ nhỏ đến lớn. Tức là chia n cho số nguyên tố nhỏ nhất (i), nếu phép chia hết, chúng ta ghi nhận thêm giá trị (a[i]++), kết quả của phép chia sẽ gán lại cho **n**, nếu không chia hết, chúng ta tiếp tục chia giá trị **n** cho số nguyên tố lớn hơn.

Lưu ý rằng: Khi duyệt các số i lần lượt từ 2 trở đi, thuật toán sẽ nhảy qua số không phải là số nguyên tố vì số không phải là số nguyên tố sẽ chia hết cho số nguyên tố nhỏ hơn nó, do đó ta không cần kiểm tra tính nguyên tố của các số chia.

Chương trình giải quyết bài toán:

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[1000001],i,N;
int main()
{
  freopen("nt.inp","r",stdin);
  freopen("Sangnt.out","w",stdout);
    cin>>n;
    i=2;
    while(i*i<=n)
    {
       while(n%i==0)
        { a[i]++;
            n/=i;
         }
        i++;
    }
    N=i;
    for(int i=2; i<N; i++)
    if(a[i]>0) cout<<i<<' '<<a[i]<<'\n';
    if(n>1) {a[n]=1;
             cout<<n<<' '<<1;
             }
    return 0;
}
```

Code với Python:

```python
fi=open("TSNT.INP","r")
fo=open("TSNT.OUT","w")

N=int(fi.read())
A=[0,0]

def TSNT(N):
    i=2
    while i<=N:
        while N%i==0:
            if(len(A)<i+1):
                A.append(1)
            else: A[i]=A[i]+1
            N=N//i
        A.append(0)
        i=i+1
    return i

N=TSNT(N)
for i in range(N):
    if A[i]>=1:
        fo.writelines(format(i)+' '+format(A[i])+"\n")

fo.close()
fi.close()
```

Một ứng dụng của việc phân tích N ra thừa số nguyên tố đó là các bài toán liên quan đến việc đếm số lượng các ước và tình tổng các ước của N.

Giả sử N được phân tích thành thừa số nguyên tố như sau:

$$N=a^i * b^j * \ldots * c^k.$$

Khi đó số lượng các ước của N là:

$$(i+1)*(j+1) * \ldots * (k+1).$$

CHUYÊN ĐỀ 3: XỬ LÝ DÃY SỐ

Bài toán có xử lý dãy số là một yêu cầu thường gặp trong dạy học lập trình và cũng là nội dung quan trọng trong việc bồi dưỡng HSG. Vì vậy cần rèn luyện cho học sinh kỹ năng nhận dạng bài toán khi được phát biểu dưới nhiều dạng khác nhau và lựa chọn thuật toán thích hợp để giải quyết.

***Một số thuật toán và kỹ thuật xử lý dãy số thông dụng.**

- Duyệt các phần tử, tìm kiếm đơn giản.
- Thuật toán tìm giá trị nhỏ nhất, lớn nhất của dãy số.
- Thuật toán sắp xếp dãy số: Sắp xếp nổi bọt, sắp xếp nhanh, đếm phân phối
- Thuật toán tìm kiếm nhị phân.
- Kỹ thuật dùng mảng đánh đánh dấu (lùa bò vào chuồng)
- Kỹ thuật dùng mảng nhớ tổng tiền tố, hậu tố.
- Kỹ thuật duyệt đoạn con liên tiếp.

***Một số hàm sẵn có trên dãy số của C++**

+ Hàm sắp xếp tăng:

sort (tên mảng + chỉ số đầu, tên mảng + chỉ số cuối +1);

Ví dụ: Sắp xếp n phần tử tăng từ a[1] đến a[n]: sort (a +1, a+ n + 1); Sắp xếp n phần tử tăng từ a[0] đến a[n - 1]: sort (a , a+ n);

+ Hàm sắp xếp giảm:

sort (tên mảng + chỉ số đầu, tên mảng + chỉ số cuối +1, greater < int >( ));

Ví dụ: Sắp xếp n phần tử giảm từ a[1] đến a[n]:

sort (a +1, a+ n + 1, greater < int >( )); Sắp xếp n phần tử giảm từ a[0] đến a[n - 1]:

sort (a , a+ n, greater < int >( ));

+ Hàm tìm kiếm nhị phân: binary_search(l, r, <giá trị >);

Tìm kiếm xem <giá trị> có xuất hiện trong đoạn [l, r - 1] của đoạn cần tìm không (lưu ý đoạn tìm kiếm phải được sắp xếp theo một trật tự nhất định). Nếu tìm thấy <giá trị> thì trả về true, ngược lại trả về false.

Bài toán áp dụng:

***Bài 1:* Marathon:**

Trong cuộc thi chạy marathon có n thí sinh tham dự. Số thứ tự của mỗi thí sinh chính là số báo danh của thí sinh. Thí sinh thứ ico thời gian về đích là a_i.

Ban tổ chức kỳ thi quyết định chọn ra k thí sinh có thời gian chạy ít nhất để trao giải. Em hãy đưa ra số báo danh của k thí sinh được trao giải theo thời gian tăng dần.

Dữ liệu vào: Dòng đầu chứa 2 số nguyên n và k (n,k<=1000) Dòng 2 là thời gian chạy của thí sinh thứ 1 đến thứ n

Kết quả ra: K số trên 1 dòng là số báo danh của thí sinh được giải

Ví dụ:

Marathon.inp	Marathon.out
5 3 9 1 8 6 3	2 5 4

Ý tưởng:

Sử dụng mảng với kiểu dữ liệu **pair** với chỉ số **first** lưu thời gian chạy của thì sinh, chỉ số **second** lưu số báo danh của thí sinh.

- Xây dựng hàm **bool cmp**() để xác định tiêu chí sắp xếp.
- Sử dụng hàm **sort** với tiêu chí sắp xếp cmp để sắp xếp thí sinh theo thời gian chạy tăng dần.
- Duyệt mảng sau khi sắp xếp để xuaart kết quả.

Chương trình giải:

there | *Pair.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
pair<int, int> a[1001];
int n,k;
/************************/
bool cmp(pair<int,int> x, pair<int,int> y)
{
    return (x.first<y.first);
}
/************************************/
int main()
{
    freopen("Marathon.inp","r",stdin);
    freopen("Marathon.out","w",stdout);
    cin>>n>>k;
    for(int i=1;i<=n;i++)
    {   cin>>a[i].first;
        a[i].second=i;
    }
    sort(a+1,a+1+n,cmp);
    for(int i=1;i<=k;i++) cout<<a[i].second<<' ';
    return 0;
}
```

Code với Python:

```python
# Bai 1 - Marathon
with open('Marathon.inp', 'r') as infile, open('Marathon.out', 'w') as outfile:
  n, k = infile.readline().rstrip('\n').split(' ')
  n = int(n)
  k = int(k)
  a = infile.readline().split(' ')
  pair = zip([int(i) for i in a], [i for i in range(1, n+1)])
  s = [t[1] for t in sorted(pair)]
  for i in range(k):
    outfile.write(f'{s[i]} ')
```

Lưu ý:

Trong các bài toán sắp xếp với nhiều tiêu chí khác nhau, để sử dụng hàm sort() sẵn có ta cần xây dựng hàm bool cmp() để xác định tiêu chí sắp xếp. Khi đó ta sắp xếp với lời gọi: *sort(a+1,a+1+n,cmp);* việc này sẽ giúp ta tiết kiệm thời gian viết lại code hàm sắp xếp.

***Bài 2:* SEQ (đề thi HSG tỉnh Nghệ An 2016-2017)**

Cho dãy số gồm n số nguyên $a_1, a_2, \ldots, a_n$ và 2 số nguyên không âm L, R ($L \le R$).

Yêu cầu:

Đếm số cặp (i, j) thỏa mãn điều kiện: $i \le j$ và $L \le |a_i+\ldots+a_j| \le R$.

Dữ liệu vào:

Từ file văn bản SEQ.INP gồm:

- Dòng đầu tiên chứa 3 số nguyên n, L, R ($n \le 10^5$; $0 \le L \le R \le 10^9$)
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, \ldots, a_n$ ($a_i \le 10^9$)

Kết quả:

Ghi ra file văn bản SEQ.OUT

Gồm một số nguyên duy nhất là số lượng cặp (i, j) đếm được.

Ví dụ:

SEQ.INP	**SEQ.OUT**
3 0 1 1 -1 2	4

Hạn chế: - Có 50% số test ứng với $0 < n \le 10^3$

- Có 50% số test ứng với $10^3 < n \le 10^5$

Ý tưởng:

- Sử dụng kết hợp các thuật toán: *Sử dụng kĩ thuật mảng nhớ tổng tiền tố; Sắp xếp; Tìm kiếm nhị phân.*

- Thực hiện:

+ Đọc giá trị của dãy, kết hợp tạo mảng nhớ tổng a, với a[i] là tổng i phần tử đầu của dãy.

+ Sắp xếp các giá trị trong mảng theo thứ tự tăng dần.

+ Xét các giá trị a[i] với $2 \le i \le n$, với mỗi giá trị a[i] ta cần tìm kiếm nhị phân 2 vị trí:

Vị trí d: là vị trí mà a[d] bé nhất nhưng a[d] ≥a[i]-r Vị trí c: là vị trí mà a[c] lớn nhất nhưng a[c] ≤a[i]-l

Và lúc này số lượng đoạn con thỏa mãn kq sẽ tăng lên kq= kq+(c-d+1); Sau khi thực hiện n-1 lần thì số đoạn con thỏa mãn đề bài chính bằng kq.

Chương trình giải quyết bài toán:

there | seq.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long
ll a[100001],l,r,kq=0,n,x,fd,fc;
/*********************************/
void open()
{
    freopen("seq.inp","r",stdin);
    freopen("seq.out","w",stdout);

    ios_base::sync_with_stdio(NULL);
    cin.tie(0);
    cout.tie(0);
}
/*********************************/
int binary_d(ll x,int d,int c)
{
    int g,kq=-1;
    while(d<=c)
    {   g=(d+c)/2;
        if(a[g]<x) d=g+1;
        else
        {   kq=g;
            c=g-1;
        }
    }
    return kq;
}
/*************************/
int binary_c(ll x ,int d,int c)
{
    int g,kq=n;
    while(d<=c)
    {   g=(d+c)/2;
        if(a[g]>x) c=g-1;
        else
        {   kq=g;
            d=g+1;
        }
    }    return kq;
}
/*************************************/
int main()
{   open();
    cin>>n>>l>>r;
    a[0]=0;
    for(int i=1;i<=n;i++)
    {   cin>>x;
        a[i]=a[i-1]+x;
    }
    sort(a,a+n+1);
    for(int i=0;i<n;i++)
    {  fd=binary_d(a[i]+l,i+1,n);
       if(fd!=-1)
       {    fc=binary_c(a[i]+r,i+1,n);
            kq+=(fc-fd+1);
       }
    }    cout<<kq;
    return 0;
}
```

Code với Python:

```python
# Bai 2 - SEQ
def binary_d(x, d, c):
    kq = -1;
    while d <= c:
        g = (d+c)/2
        g = int(g)
        if a[g] < x:
            d = g+1
        else:
            kq = g
            c = g-1
    return kq

def binary_c(x, d, c):
    kq = n
    while d <= c:
        g = (d+c)/2
        g = int(g)
        if a[g] > x:
            c = g - 1
        else:
            kq = g
            d = g + 1
    return kq


with open('SEQ.inp', 'r') as infile, open('SEQ.out', 'w') as outfile:
    a = [0]
    kq = 0

    n, l, r = infile.readline().rstrip('\n').split(' ')
    n = int(n)
    l = int(l)
    r = int(r)
    arr = infile.readline().split(' ')
    for i in range(1, n):
        x = int(arr[i])
        a.append(a[i-1]+x)
    a = sorted(a)
    for i in range(n):
        fd = binary_d(a[i]+l, i+1, n)
        if fd != -1:
            fc = binary_c(a[i]+r, i+1, n)
            kq += (fc - fd + 1)
    outfile.write(f'{kq}')
```

CHUYÊN ĐỀ 4: XỬ LÝ XÂU KÝ TỰ

Xử lý xâu ký là yêu cầu thường gặp trong dạy học lập trình và cũng là bài toán thường xuyên xuất hiện trong các đề thi HSG. Các dạng bài toán phổ biến như: Duyệt các phần tử, duyệt xâu con trên xâu; biến đổi, xử lý trên xâu; lưu trữ và xử lý số lớn. Để dễ dàng trong việc giải các bài toán về xâu, ta cần nắm rõ các hàm xử lý trên kiểu dữ liệu xâu để vận dụng một cách linh hoạt vào từng bài tập cụ thể.

Một số thao tác mở rộng với xâu trong C++

- Hàm chuyển xâu s thành số n kiểu int: n = atoi(s.c_str());
- Hàm chuyển xâu s thành số n kiểu long: n = atol(s.c_str());
- Hàm chuyển xâu s thành số n kiểu long long: n = atoll(s.c_str());
- Hàm chuyển xâu s thành số n kiểu float: n = atof(s.c_str());
- Hàm chuyển số n thành xâu s:

stringstream convert; covert << n; s = convert.str();

- Hàm s.size(); lấy chiều dài của xâu s.
- s.find(x, vt): Tìm kiếm vị trí đầu tiên xuất hiện xâu x trong xâu s.
- s.rfind(x, vt): Tìm kiếm vị trí cuối cùng xuất hiện xâu x trong xâu s.
- .....

Một số thao tác mở rộng với xâu trong Python

- Hàm chuyển xâu s thành số n kiểu nguyên: n = int(s);
- Hàm chuyển xâu s thành số n kiểu thực: n = float(s);
- Hàm chuyển số n thành xâu s: S= str(s);
- Hàm len(s); lấy chiều dài của xâu s;
- s.find(x): Tìm kiếm vị trí đầu tiên xuất hiện xâu x trong xâu s;
- s.rfind(x): Tìm kiếm vị trí cuối cùng xuất hiện xâu x trong xâu s;
- s.split(dk): cắt chuỗi s thành nhiều phần tử danh sách theo điều kiện dk
- s.replace(old, new, max): Thay thế các chuỗi old tìm được trong chuỗi s bằng chuỗi new với số lượng max có thể thay thế.

***Bài toán áp dụng:* Chuẩn hóa văn bản:**

Một văn bản được gọi là văn bản chuẩn nếu:

- Hai từ liền nhau có duy nhất một dấu cách trống.

- Dấu ngắt câu (dấu chấm, dấu phẩy, dấu chấm phẩy, dấu chấm hỏi, dấu chấm than) được đặt sát vào từ ngay trước nó, sau đó mới đến dấu cách trống.

- Dấu mở ngoặc đặt sát vào phía bên trái của từ bắt đầu mở ngoặc.

- Dấu đóng ngoặc đặt sát bên phải từ cuối cùng được đóng ngoặc. Cho một xâu ký tự. Hãy đưa ra xâu đó ở dạng văn bản chuẩn.

Dữ liệu vào: tệp văn bản CHVB.inp chứa một xâu ký tự.

Kết quả ra: Ghi vào tệp văn bản CHVB.inp gồm 1 dòng là kết quả của bài toán.

Ý tưởng:

- Xóa các dấu cách thừa: dùng lệnh for lùi duyệt để tìm và xóa.

- Xử lý với các dấu câu:

+ Dùng xâu phụ để chứa các dấu câu: s1=".,;?!)";

+ Duyệt từng ký tự i của xâu s:

+/ Nếu s[i] có mặt trong xâu s1 thì xóa dấu cách phía trước nếu có và thêm dấu cách phía sau nếu chưa có.

+/ Nếu s[i] là dấu ngoặc mở thì xóa dấu cách phía sau nếu có và thêm dấu cách phía trước nếu chưa có.

Chương trình như sau:

art here | CHVB.cpp

```
#include <bits/stdc++.h>
using namespace std;
string s,s1=".,;?!)";
int i, l;
int main()
{
    freopen("CHVB.inp","r",stdin);
    freopen("CHVB.out","w",stdout);
    getline(cin,s);
    l=s.length();
for (i=l-1; i>=0; i--)
if((s[i]==' ')&&(s[i-1]==' '))s.erase(i,1);
for (i=0; i<=l; i++)
    { if(s1.find(s[i])!=-1)
        {   if(s[i+1]!=' ') s.insert(i+1," ");
            if(s[i-1]==' ') s.erase(i-1,1);
        }
      if (s[i]=='(')
        { if (s[i+1] == ' ') s.erase(i+1,1);
          if (s[i-1] != ' ') s.insert(i-1," ");
        }
    }
cout <<s<<endl;
   return 0;
}
```

Code với Python:

```
import string

fi=open("CHVB.INP","r")
fo=open("CHVB.OUT","w")

s=fi.read()

daucau=['.', ',' ,';' ,'?', '!', ')','(']

def xoakitutrangdaucuoi(s):
    s=s.strip()
    return s
def xoakitutrangduthua(s):
    while s.find("  ")>0:
        s=s.replace("  "," ")
    return s
def themkhoangtrang(s):
    for i in daucau:
        if i=='(':
            s=s.replace(i,' '+i)
        else:
            s=s.replace(i,i+' ')
    return s
def chuanhoadau(s):
    for i in daucau:
        if i=='(':
            s=s.replace(i+' ',i)
        else:
            s=s.replace(' '+i,i)
    return s

s=xoakitutrangdaucuoi(s)
s=themkhoangtrang(s)
s=xoakitutrangduthua(s)
s=chuanhoadau(s)
fo.write(s)
fi.close()
fo.close()
```

CHUYÊN ĐỀ 5: ĐỆ QUY, QUAY LUI

**Khái niệm về đệ quy.*

Ta nói một khái niệm là đệ qui nếu nó gao gồm chính nó như một bộ phận hoặc nó được định nghĩa dưới dạng của chính nó.

Trong tin học, hàm đệ quy là hàm mà bản thân nó có thể gọi lại chính nó.

**Đặc điểm của hàm đệ quy:*

- Trong hàm đệ quy có lời gọi đến chính hàm đó.
- Mỗi lần gọi lại hàm đó thì kích thước bài toán đã thu nhỏ hơn trước.
- Có một trường hợp đặc biệt: đó là trường hợp suy biến.

**Hiệu lực của đệ quy:*

- Ưu điểm: Sáng sủa, dễ hiểu, thủ tục rất gọn, đơn giản
- Nhược điểm: Đệ quy quay lui, thường tính toán nhiều, thời gian thực hiện rất lâu.

**Thiết kế giải thuật đệ quy*

Ba bước thiết kế đệ quy

- Tham số hóa bài toán: xác định các tham số in và out
- Lời gọi đệ quy: Đưa bài toán về bài toán cùng loại nhỏ hơn, dần tiến tới trường hợp suy biến
- Tìm trường hợp suy biến (điểm dừng)

Bài toán áp dụng đệ quy:

Bài 1: Đệ quy cơ bản.

Cho 2 số tự nhiên a và n. Hãy tính giá trị a^n %(10^9+7)

Dữ liệu vào: Đọc từ tệp DQ1.Inp gồm một dòng duy nhất chứa 2 số nguyên a, n (1<=a, n<=10^9).

Dữ liệu ra: Ghi kết quả ra tệp DQ1.Out một số nguyên là kết quả của bài toán. Ví dụ

DQ1.inp	DQ1.out
2 8	256

Ý tưởng:

Cách 1: Đặt hằng e =1000000007;
Khởi tạo kq =1;

Dùng lệnh for lặp lại n lần. Mỗi lần gán kq=(kq*a)%e.

Việc chia lấy dư cho e ngay lập tức sẽ làm cho giá trị kq luôn nhỏ hơn e và sẽ tránh được việc vượt phạm vi lưu trữ. Cách làm này chưa đảm bảo thời gian chạy 1 giây khi n lớn. Vì vậy ta cần cải tiến như các 2 dưới đây.

Cách 2: Ta có thể dùng đệ quy để tính kết quả theo nguyên tắc sau.:

$$a^n \% e = (a^{\frac{n}{2}} \% e)(a^{\frac{n}{2}} \% e) \quad \text{nếu } n \text{ chẵn}$$

$$a^n \% e = (a^{\frac{n}{2}} \% e)(a^{\frac{n}{2}} \% e) * a \quad \text{nếu } n \text{ lẻ}$$

Khi đó ta có chương trình:

DQ1.cpp

```
#include <bits/stdc++.h>
using namespace std;
const int e =1000000007;
long long n,a;
/*************************/
long long mu(long long a, long long n)
{
    if(n==0) return  1;
    long long   tg = mu(a,n/2);
    tg  =  (tg*tg)%e;
    if(n%2==1)  tg = (tg*a)%e;
    return  tg;
}
int  main()
{
freopen("DQ1.inp","r",stdin);
freopen("DQ1.out","w",stdout);
    cin>>a>>n;
    cout<<mu(a,n);
    return 0;
}
```

Code với Python:

```
import string

fi=open("DQ1.INP","r")
fo=open("DQ1.OUT","w")

e=1000000007
s=fi.read()
s=s.split(" ")
a=int(s[0])
n=int(s[1])
tg=1

def mu(a,n):
    if n==0: return 1
    else:
        tg=mu(a,n//2)
        tg=tg*tg%e
        if n%2==0: return tg
        else: return tg*a%e
kq=mu(a,n)
fo.write(format(kq))
fi.close()
fo.close()
```

Lưu ý: *- Không nên sử dụng hàm pow(a,n) có sẵn, vì sử dụng hàm này có một số trường hợp cho kết quả không đúng mong muốn.*

- Giá trị cuả a và n có thể là rất lớn, nên ta cần cân nhắc trong việc chọn kiểu dữ liệu và lựa chọn thuật toán.

- Trong bài toán này ta đã sử dụng đệ quy cơ bản để làm giảm thời gian thực hiện chương trình.

Bài 2: Đệ quy quay lui:

Cho số tự nhiên N.

Hãy liệt kê ra các hoán vị của tập các số tự nhiên từ 1 đến N.

Dữ liệu vào: Đọc từ tệp Hoanvi.Inp gồm một dòng duy nhất chứa 2 số nguyên a, n (1<=30).

Dữ liệu ra: Ghi kết quả ra tệp Hoanvi.Out mỗi dòng là 1 hoán vị.

Ví dụ

Hoanvi.inp	Hoanvi.out

3	123 132 213 231 312 321

art here | *Hoan vi.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n,x[30],d[30];

void Xuat()
{
     for(int i=1;i<=n;i++) cout<<x[i];
     cout<<'\n';
}
long long Try(long long i)
{
     for(int j=1;j<=n;j++)
          if(d[j]==0)
          {
          x[i]=j;
          d[j]=1;
          if(i==n) Xuat();
          else Try(i+1);
          d[j]=0;
          }
}
int main()
{   freopen("Hoanvi.inp","r",stdin);
    freopen("hoanvi.out","w",stdout);
cin>>n;
Try(1);
}
```

Code với Python:

```
fi=open("HOANVI.INP","r")
fo=open("HOANVI.OUT","w")

N=int(fi.read())

vet=[0]
kq=[0]

def mangvet():
    for i in range(1,N+1):
        vet.append(0)
    for i in range(1,N+1):
        kq.append(0)

def inhoanvi():
    for i in range(1,N+1):
        fo.write(format(kq[i]))
    fo.write('\n')

def hoanvi(i):
    for j in range(1,N+1):
        if vet[j]==0:
            kq[i]=j
            vet[j]=1;
            if i==N:
                inhoanvi()
            else:
                hoanvi(i+1)
            vet[j]=0

mangvet()
hoanvi(1)
#fo.write(format(kq))
fi.close()
fo.close()
```

Ý tưởng:

Ta biết khi xây dựng hoán vị thì các thành phần xây dựng không được phép lặp lại. Vì vậy ta phải sử dụng một mảng đánh dấu d[] để đánh dấu cho các giá trị đã được chọn, d[j] = 0 có nghĩa j chưa được chọn, d[j] = 1 nghĩa j đã được chọn.

CHUYÊN ĐỀ 6: QUY HOẠCH ĐỘNG

**Nguyên lý cơ bản của quy hoạch động*

- Chia bài toán cần giải thành các bài toán con.
- Sử dụng bảng để lưu trữ lời giải của các bài toán con đã được giải.

**Để xác định giải thuật một bài toán quy hoạch động cần xác định các yếu tố sau:*

- Tên và ý nghĩa các biến phục vụ công thức lặp.
- Cách khai báo các biến đó.
- Công thức lặp chuyển từ một bước sang bước tiếp theo.
- Giá trị khởi tạo của các biến tham gia tính lặp.
- Tham số điều khiển lặp thay đổi từ đâu đến đâu.
- Kết quả lưu ở đâu, làm thế nào để xuất ra kết quả.

***Bài toán áp dụng:* Dãy con tăng dài nhất.**

Cho dãy số nguyên A = a1,a2...,an. Dãy con của A là một cách chọn trong dãy A một số phần tử giữ nguyên thứ tự.

Yêu cầu: Hãy tìm một dãy con của A tăng dần có số lượng phần tử nhiều nhất.

Dữ liệu vào: file văn bản QHĐ.INP, gồm 2 dòng:

Dòng 1: ghi số N là số lượng phần tử của dãy N<=1000

Dòng2: ghi N số nguyên a_i cách nhau ít nhất một dấu cách |ai|<=100000 Kết quả: ghi ra file văn bản QHD.OUT, gồm 2 dòng:

Dòng 1: ghi M là số lượng phần tử của dãy con

Dòng 2: ghi lần lượt giá trị của các số trong dãy A ban đầu

Ví dụ:

QHD.INP	QHD.OUT
10 1 2 4 8 9 5 7 8 20 9	7 1 2 4 5 7 8 9

Ý tưởng:

- Dữ liệu: + Sử dụng mảng D: d[i] độ dài dãy con tăng dài nhất chứa phần tử a[i] tính từ phần tử 1 đến phần tử i.

+ Sử dụng mảng P: p[i] là chỉ số phần tử mà a[i] cần móc nối ở bước thứ i, để

dãy con tăng dần kết thúc ở a[i].

- Khai báo: Các phần tử của mảng P, D cùng kiểu với N.

- Khởi tạo D[1]=1; P[1]=0;

- Công thức lặp:

D[i]=1; P[i]=0

Duyệt các giá trị j=1 đến i-1:

Nếu D[j]+1>D[i] thì P[i]=j; D[i]=D[j]+1;

- Phạm vi lặp i=2 đến n.

- Kết quả: + M= max của mảng D.

+ Duyệt dãy con từ cuối về đầu để lấy kết quả.

Chương trình giải:

there | qhd1.cpp

```
#include<bits/stdc++.h>
using namespace std;
int n,a[1001],d[1001],p[1001];
int kq,vet[1001],vt;
/*******************************/
void open()
{    freopen("qhd.inp","r",stdin);
    freopen("qhd.out","w",stdout);
    ios_base::sync_with_stdio(NULL);
    cin.tie(0);
    cout.tie(0);
}
/*********************************/
int main()
{    open();
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    d[1]=1; p[1]=0;
for(int i=2;i<=n;i++)
{    d[i]=1; p[i]=0;
    for(int j=1;j<=i-1;j++)
        if(a[j]<a[i])
            if(d[j]+1>d[i])
            {
                p[i]=j;
                d[i]=d[j]+1;
            }
}    kq=1;
for(int i=1;i<=n;i++)
    if(d[i]>=kq)
    {    kq=d[i];
        vt=i;
    }
cout<<kq<<'\n';
  for(int i=kq;i>=1;i--)
  {    vet[i]=vt;
      vt=p[vt];
  }
  for(int i=1;i<=kq;i++)
    cout<<a[vet[i]]<<' ';
return 0;
}
```

Code với Python:

```python
# Đọc dữ liệu
with open("qhd.inp","r") as fi:
    n = int(fi.readline())
    a = [int(x) for x in fi.readline().split()]
    a = [0] + a

#Quy hoạch động
d=[0]*(n+1)
p=[-1]*(n+1)
for i in range(1,n+1):
    d[i] = 1;
    p[i] = 0;
    for j in range(1,i):
        if a[i]>a[j] and d[i] < d[j]+1:
            d[i] = d[j]+1
            p[i] = j
```

```python
#Truy vết
vmax = 1
for i in range(2,n+1):
    if d[i] > d[vmax]:
        vmax = i
ans = []
while vmax>0:
    ans = [a[vmax]] + ans
    vmax= p[vmax]

#Ghi kết quả
with open("qhd.out","w")as fo:
    fo.write(str(len(ans))+'\n')
    for x in ans:
        fo.write(str(x)+' ')
```

PHỤ LỤC

MỘT SỐ BÀI TẬP LUYỆN TẬP, MỞ RỘNG THEO CHỦ ĐỀ HƯỚNG DẪN GIẢI VÀ CHƯƠNG TRÌNH THAM KHẢO

1) Bài tập chủ đề số học:

Những bài toán về số học rất phổ biến trong các đề thi Tin học tuy nhiên khi xuất hiện trong các đề thi nó được phát biểu dưới các dạng khác nhau qua việc mô tả các tình huống thực tế.

Rất nhiều bài toán số học, khi nhìn vào các em nhìn thấy và dùng ngay câu lệnh lặp (vét cạn) để giải quyết vì thế thường mất điểm ở các bài tập dường như đơn giản này. Việc phân tích đề và nhận ra nội dung cốt lõi của bài toán, tìm cách thu hẹp phạm vi lặp hết mức có thể để giải quyết một cách tối ưu là rất quan trọng.

Bài tập về số học trong Tin học rất đa dạng. Trong số đó, dạng toán thường gặp như cấp số cộng, cấp số nhân, dãy số cũng rất phổ biến, vấn đề là ta cần lưu ý các công thức toán học, các quy luật cơ sở để biến đổi linh hoạt vận dụng khi cần.

Bài 1: Xếp Gạch.

Công ty XD chuyên sản suất gạch xây tường, khi các viên gạch ra lò chờ xuất bán, các viên gạch được sắp thành hình tháp. Có N hàng gạch, hàng trên cùng có A viên gạch. Biết hàng dưới nhiều hơn hàng kề trên B viên. Hỏi để xếp được N hàng gạch đó người ta đã dùng bao nhiêu viên gạch?

Dữ liệu: Vào từ tệp GACH.INP:

Một dòng chứa 3 số nguyên N, A, B.

Kết quả: Ghi ra tệp GACH.OUT

Số nguyên duy nhất – số lượng viên gạch đã dùng.

Ví dụ

GACH.INP	GACH.OUT
5 1 1	15

Giải thích: N=5, A=1, B=1 thì tổng số viên gạch đã dùng là 1+2+3+4+5=15 viên gạch.

Giới hạn: $N \le 2 \times 10^{12}$; $A \le 20$; $B \le 20$.

Hướng dẫn:

Đây chính là bài toán tính tổng N số hạng của cấp số cộng với số hạng đầu là A, công sai B. Đầu tiên tính số gạch ở hàng N, rồi tính tổng của N hàng.

Chương trình:

```cpp
#include<bits/stdc++.h>
using namespace std;
int N,A,B;
long long Hcuoi,tong;
/***********************/
int main()
{
    freopen("Gach.inp","r",stdin);
    freopen("Gach.out","w",stdout);
    cin>>N>>A>>B;
    Hcuoi=A+B*(N-1);
    tong=float(N)/2*(A+Hcuoi);
    cout<<tong;
    return 0;
}
```

Code với Python

```python
inp = open('gach.inp','r')
out = open('gach.out','w')

n, a, b = [int(x) for x in inp.read().split()]

last = a  + a * (n - 1)
ans = (a + last) * n // 2

out.write(str(ans))

inp.close()
out.close()
```

Bài 2: Số nhà

An trên đường đi học về đi qua một con phố có n ngôi nhà được đánh số từ 1 tới n. Những ngôi nhà bên trái của dãy phố được đánh số lẻ, ở bên phải được đánh số chẵn. An vừa đi vừa nhìn sang bên trái và tính nhẩm tổng các số nhà trên con phố. Bạn hãy giúp An tính tổng này nhé.

Dữ liệu vào: file văn bản SN.INP, ghi số N là số lượng phần tử của dãy $N<=10^9$.

Kết quả: ghi ra file văn bản SN.OUT, gồm một số nguyên duy nhất là kết quả của bài toán.

Ví dụ:

SN.INP	SN.OUT
19	100
10	25

Hướng dẫn: Đây là bài toán tính tổng các số lẻ thuộc đoạn từ 1 đến n. Do n chưa xác định là chẵn hay lẻ nên kq=((n+1)/2*((n+1)/2);

Chương trình:

here | sonha.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n,kq;
int main()
{
    freopen("sonha.inp","r",stdin);
    freopen("sonha.out","w",stdout);
     cin>>n;
     kq=((n+1)/2)*((n+1)/2);
     cout<<kq;
     return 0;
}
```

Code với Python:

```python
inp = open('sn.inp','r')
out = open('sn.out','w')

n = int(inp.read())

ans = ((n + 1) // 2) * ((n + 1) // 2)
out.write(str(ans))

inp.close()
out.close()
```

Bài 3: Chuyến Đi *(https://ucode.vn/)*

Trong thành phố City có *n* trường học. Tất cả các trường này đều nằm trên một đường thẳng và được đánh số tăng dần từ 1 đến *n*. Khoảng cách giữa hai trường học cạnh nhau bằng 1 km. Vì tất cả các con đường trong thành phố đều là đường một chiều, nên có thể đi từ trường *a* đến trường *b* chỉ khi *a*<*b*.

Năm học mới chuẩn bị bắt đầu, An quyết định thực hiện một chuyến đi thăm tất cả *n* trường học bằng xe máy. Dung tích bình xăng của chiếc xe này là *v* lít, và chiếc xe tiêu tốn đúng 1 lít xăng cho mỗi 1 km đường đi. Khi bắt đầu cuộc hành trình, An đang ở tại trường học số 1 với bình xăng rỗng và bạn ấy muốn đến trường học số *n*.

Trước cổng mỗi trường học đều có một trạm xăng. Ở cổng trường học *i*, giá của 1 lít xăng là *i* đô-la. An không muốn lãng phí tiền xăng, do đó bạn ấy muốn bỏ ra số tiền tối thiểu mua xăng để hoàn thành chuyến đi. Các bạn hãy giúp An tính số tiền tối thiểu này.

Dữ liệu: Vào từ file văn bản CHUYENDI.INP

- Một dòng duy nhất chứa hai số nguyên *n* và *v*.

Kết quả: Ghi ra file văn bản CHUYENDI.OUT

- Một số nguyên dương duy nhất là số tiền tối thiểu An cần bỏ ra để mua xăng cho chuyến đi.

Ví dụ:

CHUYENDI.INP	CHUYENDI.OUT
4 2	**4**

CHUYENDI.INP	CHUYENDI.OUT
7 6	**6**

Hạn chế:

$2 \le n \le 10^9$; $1 \le v \le 10^9$ 50% số điểm: $n, v \le 10^3$; 25% số điểm: $n, v \le 10^6$; 25% số điểm: n, $v \le 10^9$

Hướng dẫn:

- Trường hợp n<=v thì ta đổ n – 1 lít sẽ đến được tất các trường.
- Trường hợp n>v: Đầu tiên ta đổ đầy bình v lít. Khi tới n-v-1 địa điểm đầu ta đều đổ bổ sung 1 lít cho phần xăng bị hao. khi đó giá sẽ thấp nhất.

Chương trình:

here | *chuyendi.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,v,s,d;
/************************/
int main()
{
    freopen("chuyendi.inp","r",stdin);
    freopen("chuyendi.out","w",stdout);
    cin>>n>>v;
    if(n<=v)
    {
        cout<<n-1;
        return 0;
    }
    else
    {
        s=(n-v-1)*(n-v+2)/2;
        s=s+v;
    }
    cout<<s;
    return 0;
}
```

Code với Python:

```python
inp = open('chuyendi.inp','r')
out = open('chuyendi.out','w')

n, v = [int(x) for x in inp.read().split()]

if n <= v:
    out.write(str(n - 1))
else:
    s = (n - v - 1) * (n - v + 2) // 2
    s = s + v
    out.write(str(s))

inp.close()
out.close()
```

Bài 4: Đóng gói tinh nghệ

An là nhân viên giao hàng ở xưởng làm tinh bột nghệ. Nhiệm vụ của An lần này là phải giao đúng n kg tinh nghệ cho một đại lý. Ở xưởng, tinh nghệ được đóng gói trong 2 loại túi: túi đựng được 3 kg và túi 5 kg, số lượng tinh nghệ trong mỗi túi phải được đóng đúng với sức chứa của nó, không thừa và không thiếu.

Ví dụ, để giao 18 kg tinh nghệ An có thể mang 6 túi loại 3 kg hoặc 3 túi loại 5 kg và 1 túi loại 3 kg. An luôn luôn muốn chọn phương án sao cho số túi cần mang là ít nhất.

Yêu cầu: Cho n. Hãy xác định số túi ít nhất cần mang. Nếu không có cách mang thì đưa ra số -1.

Dữ liệu vào: Nhập vào từ bàn phím số nguyên n.

Kết quả: Đưa ra màn hình một số nguyên – kết quả xác định được.

Ví dụ:

Input	*Output*
18	4

30% số điểm ứng với $n<=10^4$ 60% số điểm ứng với $n<=10^8$ 100% số điểm ứng với $n<=10^{18}$

Ý tưởng giải quyết bài toán:

Cách 1: Thử tất cả các trường hợp bằng 2 vòng lặp đơn giản:

Gọi a là số lượng túi 3kg tối đa dùng để chứa n kg tinh nghệ => a=n/3 Gọi b là số lượng túi 5kg tối đa dùng để chứa n kg tinh nghệ => b= n/5

Duyệt tìm kết quả:

```
Min:=LLO
NG_MAX; for (int i= 0; i<=
a; i++)
    for (int j= 0; j<= b; j++)
    if ((i*3+j*5=n) && (i+j<Min))
```

Min:=i+j; Kết quả tìm được là Min

Cách 2: Cải tiến cách 1, chỉ sử dụng một vòng lặp

Nếu i là số túi loại 3 kg thì ta dễ dàng tính được số túi loại 5 kg là (n-i*3)/5 Cải tiến như sau:

```
Min:=LLONG_MAX;
for (int i= 0; i<= a; i++)
{       j=(n-i*3) / 5;
        if ((i*3+j*5=n) && (i+j<Min)) Min:=i+j;
}
```

Cách 3: Nhận xét sau: để sử dụng số túi ít nhất thì ta phải đóng với số túi 5Kg nhiều nhất có thể. Từ đó ta thấy không cần sử dụng vòng lặp để làm.

```
a= n / 5;       //số túi 5kg, b số
túi 3Kg Du= n % 5; //Du chỉ
nhận giá trị 0 đến 4 Dựa vào
Du ta tính số túi còn lại
Du = 0 thì b=0;
Du = 1 thì bớt a đi
1 và b=2. Du = 2
thì bớt a đi 2 và
b=4.
Du = 3 thì a không
thay đổi, b=1 Du = 4
thì a bớt đi 1 và b = 3
```

Chú ý: Có một số trường hợp vô nghiệm khi n bé là n= 1, 2, 4, 7.

```
#include <bits/stdc++.h>
using namespace std;
long long n, a,b;
int Du;
int main()
{       cin>>n;
     if (n==1||n==2||n==4||n==7)
          cout<<-1;
     else
     {
          a= n / 5;     Du= n % 5;
          if(Du==0) b=0;
          if(Du==1)
               {a-=1; b=2;}
          if(Du==2)
               {a-=2; b=4;}
          if(Du==3) b=1;
          if(Du==4)
               {a-=1; b=3;}
          cout<<a+b;
     }
   return 0;
}
```

Code với Python:

```python
n = int(input())

if n == 1 or n == 2 or n == 4 or n == 7:
    print(-1)
else:
    a = n // 5
    du = n % 5
    if du == 0:
        b = 0

    if du == 1:
        a -= 1
        b = 2

    if du == 2:
        a -= 2
        nb = 4

    if du == 3:
        b = 1

    if du == 4:
        a -= 1
        b = 3

    print(a + b)
```

Chương trình tối ưu theo cách 3 thể hiện như sau:

- Trong bài toán trên với yêu cầu thời gian chương trình là 1 giây, nếu ta sử dụng cách 1 thì chỉ có điểm ở 30% test đầu. Sử dụng cách 2 có thể có được 60% số điểm. Để đạt 100% số điểm ta phải sử dụng cách 3.

Bài 5: Kiểm tra số Fibonaxi

Cho số tự nhiên N. Hãy Kiểm tra số N có phải là số Fibonacci hay không?

Dữ liệu vào: số nguyên N ($1<=n<=10^5$).

Dữ liệu ra: thông báo "n la so fibonacci" hoặc "n khong phai la so fibonacci".

Ví dụ

Input	Output
13	n la so fibonacci
10	n khong phai la so fibonacci

Hướng dẫn:

Để kiểm tra 1 số cho trước có phải là một số trong dãy Fibonacci hay không ta chỉ cần dùng công thức tìm số Fibonacci thứ i với i chạy từ 1 và tăng 1. Đến khi kết quả bằng số n hoặc lớn hơn là được. Nếu bằng n thì kết luận n là số Fibonacci, và nếu kết quả là lớn hơn thì kết luận n không phải.

Chương trình

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int f[30],n;
    f[0]=1; f[1]=1;
    cin>>n;
    int i=1;
    while (f[i]<n){
        i++;
        f[i] = f[i-1] + f[i-2];
    }
    if (n==f[i]) cout<<n<<" la so fibonacci";
    else cout<<n<<" khong phai la so fibonacci";
}
```

Code với Python:

```python
n = int(input())
a = 1
b = 1
while (a <= n):
    if (a == n):
        print(n, 'la so fibonacci')
        exit()
    a, b = b, a + b

print(n, 'khong phai la so fibonacci')

```

Bài 6: Phân tích N thành tổng các số Fibonaxi.

Cho số tự nhiên N. Hãy phân tích số N thành tổng ít nhất các số Fibonacci.

Dữ liệu vào: số nguyên N ($1<=n<=10^5$).

Dữ liệu ra: các số fibonacci cách nhau một dấu cách.

Ví dụ

Input	Output
33	21 8 3 1

Hướng dẫn:

Ta chỉ cần tìm số Fibonacci lớn nhất có thể được mà nhỏ hơn hoặc bằng n. Đó chính là 1 trong các số trong kết quả phân tích. Đặt n thành n – số vừa tìm được và tiếp tục quá trình cho đến khi n = 0.

Ví dụ phân tích số n= 33. Ta tiến hành tìm kiếm số thỏa mãn là 21, tiếp theo n giảm xuống còn 12 ta sẽ tìm được số tương ứng là 8, n tiếp tục giảm còn 4 và tìm được số tương ứng là 3, giảm n xuống còn 1 và tìm được số 1. Lúc này n giảm xuống còn 0. Dừng giải thuật. Kết quả thu được 33 = 21 + 8 + 3 + 1.

```
#include <bits/stdc++.h>
using namespace std;
int timfibo(int n)
{
    if (n<4) return n;
    int i=1, j=2;
    while (i+j<=n){
        int t = j;
        j = i+ j;
        i = t;
    }
    return j;
}
int main()
{
    int n;
    cin>>n;
    while (n>0){
        int i = timfibo(n);
        cout<<i<<" ";
        n-=i;
    }
}
```

Code bởi Python:

```
n = int(input())
f = {}

f[0] = 1
f[1] = 1
i = 1
while (f[i] < n):
    f[i + 1] = f[i] + f[i - 1]
    i += 1

while (n > 0):
    if (n >= f[i]):
        n -= f[i]
        print(f[i], end = ' ')
    i -= 1
```

Bài 7: Thỏ trên đảo hoang

Một thuyền thám hiểm đã để sót lại trên đảo hoang giữa đại dương một cặp thỏ mới sinh. Thỏ là một loài mắn đẻ. Từ 3 tháng tuổi trở đi, mỗi tháng một cặp thỏ sẽ sinh được một cặp thỏ con. Sau N (N<=45) tháng đoàn thám hiểm trên đường quay về đã ghé lại đảo.Vào những giờ phút nghỉ ngoi hiếm hoi, các nhà thám hiểm đã giải trí bằng cách tổ chức thống kê số lượng thỏ trên đảo. Hãy viết chương trình tính xem có bao nhiêu cặp thỏ trên đảo.

Hướng dẫn:

Với tháng thứ I, thì số thỏ = số thỏ ở tháng thứ i-1 (số thỏ hiện có) cộng số thỏ ở tháng i-2 (vì số thỏ này đã đến tuổi sinh trưởng, mỗi cặp sẽ sinh ra 1 cặp thỏ con. Như vậy yêu cầu bài toán chính là yêu cầu tính số Fibonaxi thứ n.

art here | Fibo1.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
int n;
long long f, f1, f2;
int main()
{freopen("fibo1.inp","r",stdin);
 freopen("fibo1.out","w",stdout);
     cin>>n;
     f1=f2=1;
     for(int i=3;i<=n;i++)
    {    f=(f1+f2)% 10;
         f1=f2;
         f2=f;
     }
    cout<<f;
   return 0;
}
```

Code với Python:

```python
n = int(input())

a = 1
b = 1
for i in range(3, n + 1):
    a, b = b, a + b

print(b)
```

Bài 8. Breakfast

Ở quán ăn sáng nọ có n khách quen. Mỗi vị khách cứ sau một số y ngày nhất định sẽ đến quán ăn đó để ăn sáng. Biết rằng xuất phát điểm ban đầu tất cả các vị khách quen sẽ đến ăn sáng vào ngày đầu tiên. Bạn hãy giúp chủ quán tính xem sau bao nhiêu ngày thì tất cả các vị khách quen của quán mới lại cùng đến ăn sáng cùng ngày và khi đó mỗi vị khách đã đến quán ăn bao nhiều lần.

Dữ liệu vào:

- Dòng đầu chứa số nguyên n ($2 \leq n < 100$)
- Dòng thứ hai chứa n số nguyên y. ($1 \leq y < 100$)

Dữ liệu ra:

- Dòng đâu tiên ghi ra số ngày mà tất cả các vị khách cùng đến quán ăn.
- Dòng thứ hai chứa n số là số lần một vị khách đã đến quán ăn cho tới lúc tất cả cùng đến.

Ví dụ:

Breakfast.inp	Breakfast.out
3 2 3 4	12 6 4 3

Hướng dẫn:

Đây là bài toán cơ bản áp dụng tính bội chung nhỏ nhất của dãy. Ta cần xây dựng hàm tìm bội chung nhỏ nhất của 2 số bcnn(x,y) bằng công thức bcnn(x,y)=x*y/ gcd(x,y)). Gọi B[i] là bội chung nhỏ nhất của dãy i số hạng, ta có công thức B[i]=bcnn(B[i-1],A[i]). Khi đó B[n] sẽ là bội chung nhỏ nhất của dãy. Số lần mà vị khách thứ i đã đến quán sẽ bằng B[n]/A[i].

Chương trình:

rt here | *Breakfast.cpp

```
#include<bits/stdc++.h>
using namespace std;
int m,n,u;
/*********************************/

int main()
{
    freopen("Breakfast .inp","r",stdin);
    freopen("Breakfast .out","w",stdout);
    cin>>m>>n;
    int a=__gcd(m,n);
    cout<<a<<endl;
    cout<<m/a<<' '<<n/a;
    return 0;
}
```

Code với Python:

```python
import math

n = int(input())
a = list(map(int,input().split(' ')))
lcm = a[0]
for i in a:
    p = math.gcd(lcm, i)
    lcm = lcm * i // p

print(lcm)
for i in a:
    print(lcm // i, end = ' ')
```

Bài 9. Câu đố:

Trong tiết học toán, thầy giáo đưa ra một câu đố cho cả lớp và bạn nào tìm được đáp án nhanh nhất sẽ được thưởng quà và câu đố như sau:

Trong số tất cả các cặp số tự nhiên phân biệt từ 1 đến n, hãy tìm ước số chung lớn nhất có thể có của các số nguyên trong cặp. Nói cách khác nhiệm vụ của các bạn là hãy tìm giá trị lớn nhất của ước chung lớn nhất của a và b, trong đó a,b thuộc đoạn [1, n].

Sau khi nghe xong câu đố bạn Thắng cũng rất muốn nhận quà nhưng chưa tìm được đáp án. Các bạn hãy giúp bạn Thắng nhé!

INPUT: - Dòng đầu tiên chứa số nguyên dương t ($t \leq 100$), số lượng các trường chợp cần xét.

- t dòng tiếp theo, mỗi dòng chứa 1 số nguyên dương n ($2 \leq n \leq 10^6$)

OUTPUT: gồm t dòng, mỗi dòng chứa số nguyên dương là ước số chung lớn nhất của ước số chung lớn nhất của các cặp số thuộc đoạn [1, n]

Ví dụ:

Caudo.inp	Caudo.out
2 3 5	1 2

Giải thích:

- ở trường hợp 1: n = 3, ước chung lớn nhất của mọi cặp (a, b) ≤ n là: gcd(1,2) = gcd(1,3) = gcd (2,3) = 1 nên kết quả là 1.

- ở trường hợp 2: n = 5, ước chung lớn nhất của mọi cặp (a, b) ≤ n là: gcd(1,2) = gcd(1,3) = gcd(1,4) = gcd(1,5) = gcd(2,3) = gcd(2,5) = gcd (3,4) = gcd(3,5) = gcd(4,5) = 1, gcd(2, 4) = 2 nên kết quả là 2.

Hướng dẫn:

Với mỗi số nguyên n ta nhận thấy trong các số 1,2,…,n cặp số có ước chung lớn nhất là cặp số i và 2i có uocs chung lớn nhất băng i với 2i là giá trị lớn nhất sao cho 2i≤n. Nên i=int(n/2) là kết quả bài toán.

rt here | Caudo.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long n,x;
int main()
{
    freopen("Caudo.inp","r",stdin);
    freopen("Caudo.out","w",stdout);
    cin>>n;
    for(long i=1;i<=n;i++)
    {
        cin>>x;
        cout<<int(x/2)<<endl;
    }
}
```

Code với Python:

```python
n = int(input())

for i in range(n):
    x = int(input())
    print(x // 2)
```

Bài 10. Double prime.

Số nguyên tố là một số nguyên dương có 2 ước dương là 1 và chính nó. Ví dụ: 7, 13, 17, … là những số nguyên tố; còn các số 14, 8, 25, … không phải là những số nguyên tố.

Số đảo ngược của một số là số được viết theo thứ tự ngược lại của số đó.

Ví dụ: 13 đảo ngược của nó là 31; 145 đảo ngược của nó là là 541.

Một số n được gọi là Double prime nếu như n là số nguyên tố và số đảo ngược của

n cũng là số nguyên tố. Ví dụ: 7, 13 là các số Double prime còn các số 8, 41 không phải là các số Double prime.

Cho một số nguyên dương n. Em hãy kiểm tra xem số n có phải là số Double prime không?

INPUT: Một dòng duy nhất chứa số nguyên dương n ($n \leq 2*10^9$)

OUTPUT: In ra số **1** nếu đó là số Double prime, và in ra số **0** nếu đó không phải là số Double prime.

Ví dụ:

Ví dụ 1		Ví dụ 2	
Doubleprime.inp	Doubleprime.out	Doubleprime.inp	Doubleprime.out
17	1	83	0

Hướng dẫn:

Trong bài này ta phải xây dựng hàm kiểm tra nguyên tố và hàm tính số đảo ngược của 1 số. Sau đó kiểm tra tính Double.

Chương trình

there | *doublesprime.cpp

```
#include<bits/stdc++.h>
using namespace std;
long long n;
/*****************************/
long long dao(long long n)
{
    long long x=0;
    while(n>0)
    {
        x=x*10+n%10;
        n=n/10;
    }
    return x;
}
/*****************************/
```

```cpp
bool ngto(long long x)
{
    if(x<=1) return false;
    if((x==2)||(x==3)) return true;
    if((x%2==0)||(x%3==0)) return false;
    if(x<25) return true;
    long i=5;
    while(i*i<=x)
    {   if((x%i==0)||(x%(i+2)==0))
            return false;
        i=i+6;
    }
    return true;
}
/*****************************************/
int main()
{
    freopen("Doubleprime.inp","r",stdin);
    freopen("Doubleprime.out","w",stdout);
    cin>>n;
    if((ngto(n)==true)&&(ngto(dao(n))==true))
        cout<<'1';
    else cout<<'0';
    return 0;
}
```

Code với Python:

```python
def isPrime(x):
    if x < 2:
        return 0
    i = 2
    while i * i <= x:
        if x % i == 0:
            return 0
        i += 1

    return 1

n = int(input())
rev = int(str(n)[::-1])

print(isPrime(n) and isPrime(rev))
```

Bài 11. Super prime

Một số tự nhiên *N* được gọi là Super prime nếu bản thân nó là một số nguyên tố và tất cả các số thu được bằng cách xóa lần lượt các chữ số bên phải của nó đều là số nguyên tố.

Ví dụ: Số 317 là một số Super prime vì: 317 là 1 số nguyên tố.

Xóa 1 chữ số bên phải: 31 là 1 số nguyên tố. Xóa 2 chữ số bên phải: 3 là 1 số nguyên tố.

Cho 2 số nguyên *a*, *b*. Hãy liệt kê tất cả các số Super prime thuộc đoạn [*a*, *b*].

INPUT: Một dòng ghi 2 số nguyên dương a, b ($0<a,b<10^7$)

OUTPUT: Liệt kê theo thứ tự tăng các số Super prime thuộc đoạn [*a*, *b*], mỗi số trên một dòng, hoặc ghi "NO" trong trường hợp không có số nào thuộc đoạn đó.

Superprime.inp	Superprime.out
6 30	7 23 29

Hướng dẫn:

Trong bài này ta phải xây dựng hàm sàng các số nguyên tố từ 1 đến b và xây dựng hàm kiểm tra 1 số có phải số Super prime không.

Duyệt các giá trị i từ a đến b xem số nào là số Super prime để xuất ra và tăng biến dem lên 1.

Khi kết thúc vòng duyệt nếu dem=0 thì có nghĩa là trong đoạn a đến b không có số Super prime nào.

Chương trình:

there | *Superprime.cpp

```
#include<bits/stdc++.h>
using namespace std;
const int e=10000000;
long d,c,x,dem=0,a[10000001];
/*********************************/
int sang()
{   long i=e;
    fill(a+1,a+1+i,1);
    a[0]=0; a[1]=0; i=2;
    while(i*i<=e)
    {   for(long j=i*2;j<=e;j=j+i)
            a[j]=0;
        i++;
        while(a[i]==0) i++;
    }
}
/*********************************/
bool ktra(long x)
{   while(x!=0)
    {   if(a[x]==0) return false;
        x=x/10;
    }
    return true;
}
```

```cpp
/**********************************/
int main()
{
freopen("Superprime.inp","r",stdin);
freopen("Superprime.out","w",stdout);
    sang();
    cin>>d>>c;
    long i=d;
    while(a[i]==0) i++;
    while(i<=c)
    {   if(ktra(i)==1)
        {   cout<<i<<endl;
            dem++;
        }
        i++;
        while(a[i]==0) i++;
    }
    if(dem==0)
    {   cout<<"NO";
        return 0;
    }
    return 0;
}
```

Code với Python:

```python
a, b = map(int,input().split(' '))

isPrime = [0] * (b + 1) # isPrime = 0 -> prime / isPrime = 1 -> composite

def primeSieve(x):
    isPrime[0] = 1
    isPrime[1] = 1
    i = 2
    while i * i <= x:
        j = i * i
        while j <= x:
            isPrime[j] = 1
            j += i
        i += 1

primeSieve(b)

def check(x):
    while x > 0:
        if isPrime[x] == 1:
            return 0
        x = x // 10
    return 1

for i in range(a, b + 1):
    if check(i):
        print(i)
```

Bài 12. Factorial

Giai thừa N, ký hiệu N! là tích tất cả các số nguyên từ 1 đến N. Giai thừa N tăng rất nhanh, ví dụ 5!=120, 10!=3628800. Một cách để xác định số lớn như vậy, người ta chỉ ra số lần xuất hiện các số nguyên tố trong phân tích của nó ra thừa số nguyên tố.

Ví dụ, số 825 có thể xác định bởi dãy (0 1 2 0 1) có nghĩa là

$825 = 2^0.3^1.5^2.7^0.11^1.$

Cho một số nguyên dương N□1000. Hãy tìm cách biểu diễn số N! dưới dạng số lần xuất hiện các số nguyên tố trong phân tích n! ra các thừa số nguyên tố.

INPUT: Gồm nhiều dòng, mỗi dòng chứa 1 số nguyên N (2□N□1000).

OUTPUT: Gồm nhiều dòng, mỗi dòng tương ứng với 1 dòng trong file dữ liệu vào là dãy số thể hiện biểu diễn dưới dạng phân tích thành số nguyên tố của n! (phần tử cuối cùng của dãy phải là số dương).

Ví dụ:

Factorial.inp	Factorial.out
13	10 5 2 1 1 1

Hướng dẫn:

Sử dụng kĩ thuật sàng nguyên tố và kĩ thuật phân tích thừa số nguyên tố.

Vì kết quả của n! quá lớn nên ta sẽ không thể tính n! rồi mới phân tích thừa số nguyên tố. Ta sẽ phân tích lần lượt các số từ 1 đến n thành các thừa số nguyên tô và đánh dấu vào 1 mảng.

Trong quá trình phân tích ta lưu lại giá trị của thừa số lớn nhất là maxx. Cuối cùng chúng ta sẽ duyệt từ 1 đến maxx và in ra số mũ của các số nguyên tố $\leq maxx$ tạo thành n!.

Chương trình:

art here | *Factorial.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[1001];
/**********************/
int sang()
{
    int k=1000;
    fill(a+1,a+1+k,1);
    a[1]=0;
    int i=2;
    while(i*i<=k)
    {   a[i]=1;
        for(int j=2*i;j<=k;j=j+i) a[j]=0;
        i++;
        while(a[i]==0) i++;
    }
}
/*********************************/
int ptich(int n)
{   int i=2,x,dem;
    while(i<=n)
    {   x=n;
        dem=0;
        while(x>=i)
        {   dem=dem+x/i;
            x=x/i;
        }
        cout<<dem<<' ';
        i++;
        while(a[i]==0) i++;
    }
}
/*********************************/
int main()
{    freopen("Factorial.inp","r",stdin);
    freopen("Factorial.out","w",stdout);
    sang();
    while(cin>>n)
    {   ptich(n);
        cout<<endl;
    }
    return 0;
}
```

Code với Python:

```python
n = int(input())

f = {}

def factor(x):
    i = 2
    while x > 1:
        while x % i == 0:
            x = x // i
            if i not in f:
                f[i] = 0
            f[i] += 1
        i += 1

for i in range(2, n + 1):
    factor(i)

for key, value in f.items():
    print(value, end = ' ')
```

Bài 13. Non-prime

Cho P là tập hợp các ước số dương không nguyên tố của số nguyên dương n. Hãy tìm số phần tử của tập hợp P.

INPUT: Một dòng duy nhất là giá trị của n ($1 \le n \le 10^{14}$) OUTPUT: Một dòng duy nhất là số phần tử của P

Ví dụ:

Nonprime.inp	*Nonprime.out*
20	4
180	25

Ràng buộc

Subtask1: 40% test đầu tiên có $n \le 10^6$

Subtask2: 60% test còn lại không có ràng buộc gì

Hướng dẫn giải:

Phân tích số nguyên n ra thừa số nguyên tố với số mũ như sau

$n = a1^{k1}a2^{k2}a3^{k3} \dots ai^{ki}$ khi đó số ước của số n là:

$$souoc = (k1+1)(k2+1)(k3+1)\dots(ki+1)$$

Khi đó số ước không nguyên tố của n sẽ bằng $souoc - i$

Chương trình:

rt here | *nonprime.cpp

```
#include<bits/stdc++.h>
using namespace std;
long long n,dem=0,souoc=1,smu,kq;
/************************/
int main()
{    freopen("nonprime.inp","r",stdin);
     freopen("nonprime.out","w",stdout);
     cin>>n;
    int i=2;
     while(i*i<=n)
     {     if(n%i==0)
           {    dem++;
                smu=0;
                while(n%i==0)
                {    n=n/i;
                     smu++;
                }
                souoc=souoc*(smu+1);
           }
           i++;
     }
     if(n>1)
     {    souoc=souoc*2;
          dem++;
     }
     kq=souoc-dem;
     cout<<kp;
     return 0;
}
```

Code với Python:

```python
n = int(input())

f = {}

def factor(x):
    i = 2
    while x > 1:
        while x % i == 0:
            x = x // i
            if i not in f:
                f[i] = 0
            f[i] += 1
        i += 1

factor(n)

num = 1
for key, value in f.items():
    num *= (value + 1)

print(num - len(f))
```

Bài 14. Triple prime

Cho số tự nhiên ***N*** ($N \leq 10^6$). Hãy lập trình tìm tất cả bộ ba số ***nguyên tố*** x, y, z thỏa mãn:

$$\begin{cases} x < y < z \leq N \\ x^2 + y^2 = z \end{cases}$$

INPUT: Một số nguyên dương ***N*** duy nhất

OUTPUT: Gồm nhiều dòng, mỗi dòng chứa một bộ ba số nguyên tố tìm được.

Trong trường hợp không tìm được bộ ba số nguyên tố thỏa mãn đề bài thì đưa ra -1.

Tripleprime.inp	Tripleprime.out
30	2 3 13 2 5 29
12	-1

Hướng dẫn:

Nhận thấy x,y,z là các sô nguyên tố thỏa mãn đề bài nên chắc chắn y,z phải là 2 số nguyên tố lẻ vì vậy chắc chắn x phải là số nguyên tố chẵn

nên x=2.

Ta sử dụng kĩ thuật sàng các số nguyên tố ≤ N sau đó chạy biến i từ 3 cho đến khi $2^2 + i^2 > N$ thì dừng lại. Trong quá trình duyệt nếu gặp cặp số nào thỏa mãn yêu cầu đề bài thì in ra bộ 3 số đó.

Chương trình:

there | *Tripleprime.cpp

```
using namespace std;
int n,x,y,z,dem=0,a[1000001];
/*****************************/
void sang(int x)
{
    fill(a+1,a+1+x,1);
    a[1]=0;
    int i=2;
    while(i*i<=x)
    {
        for(int j=i*i;j<=x;j=j+i)
            a[j]=0;
        i++;
        while(a[i]==0) i++;
    }
}
/*******************************/
int main()
{
freopen("Tripleprime.inp","r",stdin);
freopen("Tripleprime.out","w",stdout);
    cin>>n;
    if(n<13)
    {   cout<<"-1";
        return 0;
    }
    sang(n);
    y=3;
    z=4+y*y;
    while(z<=n)
    {
        cout<<"2 "<<y<<' '<<z<<endl;
        y++;
        z=4+y*y;
        while(a[y]==0 && a[z]==0)
        {   y++;
            z=4+y*y;
        }
    }
    return 0;
}
```

Code với Python:

```
import math
fi=open("Tripleprime.INP","r")
fo=open("Tripleprime.OUT","w")

N=int(fi.read())
A=[0,0]

def SANGNT(N):
    for i in range(2,N+1):
        A.append(1)
    t=int(math.sqrt(N))+1
    for i in range(2,t):
        if(A[i]==1):
            j=i*i
            while j<=N:
                A[j]=0
                j=j+i

SANGNT(N)
dem=0

for i in range(N-1):
    if A[i]==1:
        for j in range(i+1,N):
            if A[j]==1:
                for k in range(j+1,N+1):
                    if A[k]==1:
                        if (i**2+j**2==k):
                            fo.write('{} {} {} \n'.format(i,j,k))
                            dem+=1
if dem==0: fo.writelines('-1')

fo.close()
fi.close()
```

Bài 15. Count prime

Cho M truy vấn, mỗi truy vấn gồm 2 giá trị l_i, r_i ($1 \le l_i \le r_i \le 10^6$). Với mỗi truy vấn bạn phải trả lời câu hỏi: có bao nhiêu số nguyên tố thuộc đoạn $[l_i,r_i]$.

INPUT: Dòng 1 chứa M ($1 \le M \le 10^6$)

M dòng tiếp theo, mỗi dòng chứa hai số li và ri.

OUTPUT: Mỗi dòng chứa 1 câu trả lời tương ứng với truy vấn.

Countprime.inp	Countprime.out
3 4 10 7 20 2 30	2 5 10

Hướng dẫn:

Ở bài này chúng ta phải sử dụng kĩ thuật sàng nguyên tố và sử dụng mảng f: f[i] là số số nguyên tố ≤ i.

Khi đó kết quả của mỗi truy vấn $[li, ri]$ sẽ là $f[r] - f[l]$.

tart here | *Countprime.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
int m,l,r,f[1000005];
bool a[1000005];
/*****************************************/
void sang(int n)
{   fill(&a[0],&a[0]+sizeof(a),true);
    a[1]=false;
    for(int i=2;i*i<=n;i++)
        if(a[i])
            for(int j=i*i;j<=n;j=j+i)
                a[j]=false;
}
/*****************************************/
int main()
{   freopen("Countprime.inp","r",stdin);
    freopen("Countprime.out","w",stdout);
    ios_base::sync_with_stdio(NULL);
    cin.tie(0);
    cout.tie(0);
    sang(1000001);
    fill(f,f+1000001,0);
    f[1]=0;
    for(int i=2;i<=1000000;i++)
    {   f[i]=f[i-1];
        if (a[i]) f[i]++;
    }
    cin>>m;
    for(int i=1;i<=m;i++)
    {       cin>>l>>r;
        cout<<f[r]-f[l-1]<<'\n';
    }
    return 0;
}
```

Code với Python:

```python
import math
fi=open("Countprime.INP","r")
fo=open("Countprime.OUT","w")

So=int(fi.readline())
A=[0,0]

def SANGNT(BG,EN):
    for i in range(BG,EN+1):
        A.append(1)
    t=int(math.sqrt(EN))+1
    for i in range(2,t):
        if(A[i]==1):
            j=i*i
            while j<=EN:
                if j<len(A): A[j]=0
                j=j+i
BG=0
EN=0
ENmax=0
for i in range(1,So+1):
    ENc=EN
    Str=(fi.readline())
    Str=Str.replace('\n','')
    day=Str.split(' ')
    sodau=int(day[0])
    socuoi=int(day[1])
    if socuoi>ENmax:
        ENmax=socuoi
        if socuoi>EN: EN=socuoi
        if i==1: BD=2
        else:
            if sodau>ENc: BG=sodau
            else: BG=ENc
        SANGNT(BG,EN)
    dem=0
    print(sodau,' ',socuoi)
    for vitri in range(sodau,socuoi+1):
        if A[vitri]==1: dem+=1
    fo.write('{}\n'.format(dem))
#     print(dem)
#     SANGNT(BG,N)
fi.close()
fo.close()
```

2) Bài tập chủ đề xử lý dãy số.

Bài 16. Tưới nước

Đầu giờ lên lớp có ***n*** bạn học sinh làm trực nhật do mãi chơi nên các bạn quên tưới nước cho chậu cây cảnh. Thấy vậy thầy chủ nhiệm yêu cầu 1 bạn tưới cho chậu cây nhưng các bạn đùn đẩy công việc cho nhau. Sau một lúc bạn Nam đưa ra một giải

pháp như sau: Các bạn chuẩn bị ***n*** thẻ bài, các thẻ bài được xếp chồng lên nhau và đánh số từ 1 đến ***n*** mỗi bạn sẽ chọn một số từ 1 đến n và Nam được ưu tiên chọn số 1. Các thẻ sẽ được sắp theo thứ tự ngẫu nhiên thẻ thứ ***i*** ghi số nguyên dương ***ai***. Các bạn khác (trừ Nam) sẽ lần lượt rút 2 thẻ trên cùng, sau đó đặt lại thẻ có số nhỏ hơn và giữ cho mình thẻ có số lớn hơn. Sau ***n***-1 lần rút thì còn 1 thẻ và người chọn số được ghi trên thẻ đó sẽ phải đi tưới nước cho chậu hoa.

Yêu cầu: Sau khi thực hiện xong ***n***–1 lần rút thẻ. Các bạn muốn biết mỗi người đã giữ những thẻ bài có chỉ số nào?

INPUT: tuoinuoc.inp

- Dòng 1 chứa số nguyên dương ***n*** ($n \le 10^5$)
- Dòng thứ 2 chứa n số nguyên, số thứ ***i*** là số nguyên dương ***ai*** ($1 \le ai \le 10^9$)

OUTPUT: tuoinuoc.out

- Một dòng gồm ***n*** – 1 số, số thứ i là chỉ số của thẻ bài mà bạn thứ i đang giữ .

Ví dụ:

tuoinuoc.inp	tuoinuoc.out
8 6 3 5 8 4 7 9 1	1 3 4 5 6 7 2

Hướng dẫn:

Cứ mỗi cặp số a[1] và a[2] ta so sánh 2 số này xem số nào lớn hơn rồi in ra chỉ số của số đó và sử dụng biến luu để lưu chỉ số của số bé hơn rồi tiếp so sánh a[luu] và a[i] với $3 \le i \le n$ tương tự như cặp số a[1] và a[2].

Chương trình:

art here | *tuoinuoc.cpp

```
#include<bits/stdc++.h>
using namespace std;
long n,x,i,mn,luui;
int main()
{
    freopen("tuoinuoc.inp","r",stdin);
    freopen("tuoinuoc.out","w",stdout);
    cin>>n;
    cin>>x>>mn;
    if(x<mn)
    {   cout<<'2'<<endl;
        luui=1;
        mn=x;
    }
```

```
15          else
16          {    cout<<'1'<<endl;
17               luui=2;
18          }
19          for(long i=3;i<=n;i++)
20          {     cin>>x;
21               if(x>mn)  cout<<i<<endl;
22               else
23               {    cout<<luui<<endl;
24                    luui=i;
25                    mn=x;
26               }
27          }
28          return 0;
29     }
```

Code với Python:

```
1   import math
2   fi=open("tuoinuoc.INP","r")
3   fo=open("tuoinuoc.OUT","w")
4
5   So=int(fi.readline())
6   daysostr=fi.readline()
7   daysostr=daysostr.replace('\n','')
8   dayso=daysostr.split(' ')
9
10  def sosanh(So):
11      luu=0
12      for i in range(1,So):
13          if dayso[i]>dayso[luu]:
14              fo.write(format(i+1)+' ')
15          else:
16              fo.write(format(luu+1)+' ')
17              luu=i
18          #if dayso[i]
19
20  print(dayso)
21  sosanh(So)
22
23  fi.close()
24  fo.close()
```

Bài 17. Bottle

Cho N bình chứa nước lần lượt có thể tích là các số a_1.. a_N. Khi xếp các bình theo một dãy thì sẽ tạo thành 1 khối. Nếu xếp lần lượt các bình chứa nước theo trình tự đó thì thể tích cả khối là $a_1 + a_2 + ... + a_N + \max(0, a_2 - a_1) + \max(0, a_3 - a_2) + ... + \max(0, a_N - a_{N-1})$. Nhiệm vụ của bạn là tìm cách xếp sao cho tổng thể tích chứa của cả khối là lớn nhất có thể.

Dữ liệu: Đọc từ file **Bottle.inp**

- Dòng đầu ghi số nguyên dương N ($0 < n \le 10^5$).
- N dòng sau mỗi dòng ghi một số a_i ($1 \le i \le N$ và $1 \le a_i \le 10000$).

Kết quả: ghi ra file **Bottle.out**: Ghi trên một dòng kết quả là thể tích lớn nhất tìm được.

Ví dụ

Bottle.inp	Bottle.out
4 5 4 1 7	24

Hướng dẫn:

- Sắp xếp mảng theo thứ tự tăng dần.

- Thể tích lớn nhất sẽ bằng tổng của dãy n số và tổng cuả các hiệu số a[n-i]-a[i] với $1 \le i \le n/2$.

art here | *Bottle.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long n,a[100000];
long long t=0;
/***************************/
int main()
{    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    freopen("Bottle.inp","r",stdin);
    freopen("Bottle.out","w",stdout);
    cin>>n;
    for(long i=1;i<=n;i++)
    {   cin>>a[i];
        t=t+a[i];
    }
    sort(a+1,a+1+n);
    for(int i=n;i>n/2;i--)
    t=t+a[i]-a[n-i+1];
    cout<<t;
    return 0;
}

```

Code với Python:

```
import math
fi=open("Bottle.INP","r")
fo=open("Bottle.OUT","w")

daysostr=fi.read()
dayso=daysostr.split('\n')

dayso=sorted(dayso)
S=0
cuoi=len(dayso)-1


for i in range(cuoi):
    S=S+int(dayso[i+1])-int(dayso[i])+int(dayso[i])
fo.write(format(S))

fi.close()
fo.close()
```

Bài 18. Dãy số:

Người ta cho trước một dãy N số nguyên A_1 A_2 A_3 … A_N và một số nguyên K.

Yêu cầu: Em hãy cho biết có bao nhiêu đoạn con có từ một phần từ trở lên (dãy con liên tiếp) có tổng số bằng K.

Dữ liệu vào: Từ file văn bản DAYSO.INP có cấu trúc như sau:

- Dòng đầu tiên ghi số nguyên dương N là số lượng phần tử của dãy (N□ 6.10^4) và số nguyên K (|K| □ 6.10^{10})

- Dòng thứ 2 ghi N số nguyên A_i, mỗi số cách nhau ít nhất một dấu cách (|A_i| □ 3.10^4)

Kết quả: Ghi ra file văn bản DAYSO.OUT, Số lượng dãy con liên tiếp có tổng bằng K.

***Ví dụ*:**

DAYSO.INp	DAYSO.OUT
11 6 1 2 3 4 2 3 4 5 6 2 3	3

Chú ý: Có 50% số test có N□ 3.10^4

Hướng dẫn:

- Sử dụng mảng A để lưu mảng tiền tố của dãy số. Duyệt hết mọi đoạn con chỉ số đầu là i, chỉ số cuối là j. Nếu a[j]-a[i-1]==k thì đoạn con i đến j là một dãy thỏa mãn.

rt here | *Day so.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n, k, s=0, a[1000000], m;
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("DAYSO.INP", "r", stdin);
    freopen("DAYSO.OUT", "w", stdout);
    cin>>n>>k;
    for(int i=1; i<=n; i++)
    {   cin>>m;
        a[i]=a[i-1]+m;
    }
    for(int i=1; i<=n; i++)
    for(int j=i; j<=n; j++)
    if(a[j]-a[i-1]==k) s++;
    cout<<s;
}
```

Code với Python:

```python
import math
fi=open("DAYSO.INP","r")
fo=open("DAYSO.OUT","w")

day01=fi.readline()
day01=day01.replace('\n','')
day01=day01.split(' ')

N=int(day01[0])
K=int(day01[1])
S=0
print(N,K)

day02=fi.readline()
day02=day02.replace('\n','')
day02=day02.split(' ')
print(day02)
day02.insert(0,'0')

for i in range(1,N+1):
    day02[i]=int(day02[i])+int(day02[i-1])
day02[0]=0
print(day02)

for i in range(0,N):
    for j in range(i+1,N+1):
        if day02[j]-day02[i]==K: S+=1
fo.write(format(S))
fi.close()
fo.close()
```

Bài 19. Trò chơi với dãy số

Hai bạn học sinh trong lúc nhàn rỗi nghĩ ra trò chơi sau đây. Mỗi bạn chọn trước một dãy số gồm n số nguyên. Giả sử dãy số mà bạn thứ nhất chọn là: $b_1, b_2,$

..., b_n còn dãy số mà bạn thứ hai chọn là $c_1, c_2, ..., c_n$.

Mỗi lượt chơi mỗi bạn đưa ra một số hạng trong dãy số của mình. Nếu bạn thứ nhất đưa ra số hạng b_i (1 <= i <= n), còn bạn thứ hai đưa ra số hạng c_j (1 ≤ j ≤

n) thì giá của lượt chơi đó sẽ là $|b_i+c_j|$.

Ví dụ: Giả sử dãy số bạn thứ nhất chọn là 1, -2; còn dãy số mà bạn thứ hai chọn là 2, 3. Khi đó các khả năng có thể của một lượt chơi là (1, 2), (1, 3), (-2, 2), (-2, 3). Như vậy, giá nhỏ nhất của một lượt chơi trong số các lượt chơi có thể là 0 tương ứng với giá của lượt chơi (-2, 2).

Yêu cầu: Hãy xác định giá trị nhỏ nhất của một lượt chơi trong số các lượt chơi có thể.

INPUT: vào từ file văn bản SEQGAME.INP

- Dòng đầu là số nguyên dương ($1 \le n \le 10^5$)
- Dòng thứ hai chứa các số là dãy b ($|b_i| \le 10^9$)
- Dòng thứ hai chứa các số là dãy c ($|c_i| \le 10^9$)

OUTPUT: ghi ra file văn bản SEQGAME.OUT giá trị nhỏ nhất tìm được

SEQGAME.INP	SEQGAME.OUT
2 1 -2 2 3	0

Ràng buộc: 60% số test ứng với 60% số điểm có 1=<n =<1000 ***Hướng dẫn:*** Sắp xếp 2 xâu theo thứ tự tăng dần. Sử dụng 2 biến chạy i trên mảng b và biến j trên mảng c, i bắt đầu bằng 1 và j bằng đầu bằng n. Tổng giá trị

t=b[i]+c[j] nếu t >0 thì ta tăng i lên 1 và nếu t<0 thì giảm j đi 1. Cứ làm như vậy đến khi i=n và j=1 thì giá trị nhỏ nhất trong quá trình duyệt sẽ là kết quả bài toán.

Chương trình:

art here | *seqgame.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
long n,a[100001],b[100001],mn,t,d,c;
/***************************/
int main()
{
freopen("seqgame.inp","r",stdin);
freopen("seqgame.out","w",stdout);
    cin>>n;
for(long i=1;i<=n;i++) cin>>a[i];
    for(long i=1;i<=n;i++) cin>>b[i];
/***************************/
    mn=abs(a[1]+b[1]);
    if(mn==0)
    {   cout<<mn;
        return 0;
    }
    sort(a+1,a+1+n);
    sort(b+1,b+1+n);
/***************************/
      if(a[n]+b[n]<=0)
      {   cout<<a[n]+b[n];
          return 0;
      }
      if(a[1]+b[1]>=0)
      {   cout<<a[1]+b[1];
          return 0;
      }
      d=1;   c=n;
      while((d<=n)&&(c>=1))
      {   t=a[d]+b[c];
          if(t==0)
          {   cout<<t;
              return 0;
          }
          if(abs(t)<mn) mn=abs(t);
          if(t>0) c--;
          if(t<0) d++;
      }
      cout<<mn;
      return 0;
}
```

Code với Python:

```
import math
fi=open('seqgame.INP','r')
fo=open('seqgame.out','w')

N=int(fi.readline())
b=fi.readline()
b=b.replace('\n','')
b=b.split(' ')
for i in range(N):
    b[i]=int(b[i])
c=fi.readline()
c=c.replace('\n','')
c=c.split(' ')
for i in range(N):
    c[i]=int(c[i])

b=sorted(b)
c=sorted(c)
print(b)
print(c)

i=0
j=N-1
MIN=abs(b[i]+c[j])
t=b[i]+c[j]
test=True
if MIN==0:
    fo.write(format(MIN))
    test=False
else:
    if b[0]+c[0]>=0:
        fo.write(format(b[0]+c[0]))
        test=False
    else:
        if b[N-1]+c[N-1]<=0:
            fo.write(format(b[N-1]+c[N-1]))
            test=False
        else:
            while i<=N-1 and j>=0:
                t=b[i]+c[j]
                if t==0:
                    fo.write(format(t))
                    test=False
                    break
                if abs(t)<MIN: MIN=abs(t)
                if t>0: j-=1
                if t<0: i+=1

if test: fo.write(format(MIN))
fi.close()
fo.close()
```

Bài 20: PERFECT *(Đề thi HSG lớp 11 - NH 2016-2017)*

Trong một buổi học toán Bông được học khái niệm về số có tính chất đặc biệt:

Đó là số hoàn hảo. Số hoàn hảo là số có tổng tất cả các dương nhỏ hơn nó bằng chính nó.

Ví dụ: Số 6 là là số hoàn hảo vì nó có tổng các ước 1 + 2 + 3 = 6, số 8 không phải là số hoàn hảo vì 1 + 2 + 4 = 7, ($7 \neq 8$).

Yêu cầu: Cho dãy số a1, a2,... an. Hãy giúp Bông đếm xem trong dãy có bao nhiêu số có tổng các chữ số là số hoàn hảo.

Dữ liệu vào: Từ file văn bản PERFECT.INP gồm:

- Dòng đầu tiên là số nguyên dương n ($n \leq 100$).
- Dòng số 2 ghi n số nguyên a1, a2,... an ($0 \leq ai \leq 10^9$).

Kết quả: Ghi ra file PERFECT.OUT Một số duy nhất là kết quả của bài toán. Ví dụ:

PERFECT.INP	PERFECT.OUT
3 6 123 28	2

Hướng dẫn:

Sử dụng hàm tính các chữ số của một số và kiểm tra xem tổng các chữ số có bằng 6 hoặc 28 không. Vì chỉ có 2 số hoàn hảo ≤ 81 (tổng các chữ số của a[i] không thể vượt quá 81).

tart here | *perfect.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,x,res=0;
/*************************/
int tcs(int x)
{    int tcs=0;
     while(x>0)
     {    tcs+=x%10;   x/=10;   }
     return tcs;
}
/***********************/
bool ktra(int x)
{    if(x==6 || x==8) return true;
     return false;
}
/*********************************/
void open()
{     freopen("perfect.inp","r",stdin);
     freopen("perfect.out","w",stdout);
}
/*********************************/
int main()
{     open();
     cin>>n;
     for(int i=1;i<=n;i++)
     {     cin>>x;
          if(ktra(tcs(x))==1)  res++;
     }
     cout<<res;
     return 0;
}
```

Code với Python:

```python
import math
fi=open('PERFECT.INP','r')
fo=open('PERFECT.out','w')
def perfect(so):
    gtd=so
    tcs=1
    for i in range(2,int(so**0.5)+1):
        if so%i==0: tcs+=i+so//i
    if tcs==gtd:
        print(so)
        return True
    else: return False


N=int(fi.readline())
dayso=fi.readline()
dayso=dayso.strip()
dayso=dayso.split(' ')
print(dayso)
sl=0
for i in dayso:
    if perfect(int(i)): sl+=1
fo.write(format(sl))

fi.close()
fo.close()
```

Bài 21: Xếp kiện hàng

Kho hàng của một công ty nọ có các kiện hàng kich thước giống nhau được xếp thành N ($1 \le N \le 10000$) cột sắp thành một dãy đánh số thứ tự 1,2, … , N. Ban đầu, các cột đều có số kiện hàng bằng nhau nên có chiều cao bằng nhau, tuy nhiên, một anh công nhân mới vào làm đó di chuyển một số kiện hàng giữa các cột, làm cho chiều cao của các cột không bằng nhau nữa.

Cho số lượng kiện hàng của các cột, hãy giúp công ty tinh số lượng kiện hàng ít nhất phải di chuyển để khôi phục lại trạng thái các cột như ban đầu, nghĩa là có chiều cao bằng nhau.

Dữ liệu: Vào từ file văn bản SAPHANG.inp.

- Dòng 1: số nguyên N là số lượng cột kiện hàng,
- Dòng 2: chứa n số nguyên a_i là số kiện hàng trong cột thứ i.

Kết quả: Ghi ra file văn bản SAPHANG.out

- Một dòng duy nhất ghi một số nguyên là số kiện hàng cần phải di chuyển

để khôi phục lại trạng thái các cột có chiều cao bằng nhau.

Ví dụ

SAPHANG.inp	SAPHANG.out
4 2 10 7 1	7

Hạn chế: $1 \le N \le 10^6$; $1 \le a_i \le 10^6$

Hướng dẫn:

- Tính số kiện hàng s trong mỗi cột cần sắp, s chính là trung bình cộng của các số trong dãy. Kết quả của bài toán chính là số đơn vị của các cột cao hơn giá trị trung bình.

there | *saphang.cpp

```
#include<bits/stdc++.h>
using namespace std;
int n,a[10005],s=0,sl=0;
/************************/
int main()
{
    freopen("saphang.inp","r",stdin);
    freopen("saphang.out","w",stdout);
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        s=s+a[i];
    }
    s=s/n;
    for(int i=1;i<=n;i++)
        if(a[i]>s) sl=sl+a[i]-s;
    cout<<sl;
    return 0;
}
```

Code với Python:

```
import math
fi=open('SAPHANG.INP','r')
fo=open('SAPHANG.out','w')

N=int(fi.readline())
dayso=fi.readline()
dayso=dayso.strip()
dayso=dayso.split(' ')
print(dayso)
S=0
sl=0
for i in dayso:
    S+=int(i)
S=S//N
for i in dayso:
    if int(i)>S: sl+=int(i)-S

fo.write(format(sl))

fi.close()
fo.close()
```

Bài 22: HOÀN HẢO

Số X được gọi là số hoàn hảo nếu tổng các ước của X bằng hai lần chính nó. Ví dụ các số hoàn hảo:

- 6 có các ước 1, 2, 3, 6 (1+2+3+6 = 2*6)
- 28 có các ước 1, 2, 4, 7, 14, 28 (1+2+4+7+14+28 = 2*28).

Yêu cầu: Hãy xây dựng chương trình tìm ra các số hoàn hảo từ nhỏ đến lớn có trong một dãy số nguyên cho trước.

Dữ liệu: Vào từ file văn bản HOANHAO.INP có cấu trúc như sau:

- Dòng đầu tiên là số nguyên dương N
- Dòng thứ hai là N số nguyên dương có trong dãy

Kết quả: Đưa ra file văn bản HOANHAO.OUT các số hoàn hảo có trong tệp theo thứ tự tăng dần trên một dòng. Các số các nhau bởi một dấu cách trống, số cuối cùng có thêm một dấu cách trống.

Ví dụ:

HOANHAO.INP		HOANHAO.OUT
11 5 8 6 81 22 11 5 28 68 6 6		6 6 6 28

Chú ý:số lượng các số hoàn hảo không vượt quá 10000.

Có 50% test có số lượng phần tử nhỏ hơn 10^3; mỗi phần tử nhỏ hơn 10^5

Có 25% test có số lượng phần tử lên đến 10^7 phần tử; mỗi phần tử nhỏ hơn 10^5

Hướng dẫn:

- Trong phạm vi 1 đến 10^5 chỉ có 4 số hoàn hảo là: 6 28 496 8128

- Ta dùng mảng b để lưu 4 giá trị này. Mảng A để lưu số lượng số tương ứng mảng b

- Đọc, kiểm tra tính hoàn hảo rồi cập nhật vào mảng A.

- Duyệt mảng A để xuất các giá trị tương ứng ở mảng B

Chương trình:

art here | Day Hoanhao.cpp

```
#include<bits/stdc++.h>
using namespace std;
int n, i, m, s=0, a[4]={0, 0, 0, 0};
int b[4]={6, 28,496, 8128};
int main()
{
    freopen("HOANHAO.INP", "r", stdin);
     freopen("HOANHAO.OUT", "w", stdout);
     cin>>n;
     for(i=1; i<=n; i++)
     {  cin>>m;
        for(int j=0; j<=3; j++)
         if (m==b[j]) a[j]++;
     }
     for(i=0; i<=3; i++)
     {   for (int j=1;j<=a[i]; j++ )
         cout<<b[i]<<' ';
     }
     return 0;
}
```

Code với Python:

```python
import math
fi=open('HOANHAO.INP','r')
fo=open('HOANHAO.out','w')
def perfect(so):
    gtd=so
    tcs=1
    for i in range(2,int(so**0.5)+1):
        if so%i==0: tcs+=i+so//i
    if tcs==gtd:
        print(so)
        return True
    else: return False


N=int(fi.readline())
dayso=fi.readline()
dayso=dayso.strip()
dayso=dayso.split(' ')
print(dayso)
kq=[]
for i in dayso:
    if perfect(int(i)): kq.append(int(i))
kq=sorted(kq)
for i in kq: fo.write('{} '.format(i))

fi.close()
fo.close()
```

3) Bài tập chủ đề xử lý xâu ký tự

***Bài 23:* Tìm kiếm và thay thế:**

Trong khi soạn thảo văn bản một số người thường có thói quen gõ tắt cho nhanh nhưng với văn bản hành chính các từ này là không hợp lệ, một số khác lại thường xuyên gõ lỗi 1 số từ nào đó. Để sửa những lỗi này các phần mềm soạn thảo đều có chức năng tìm kiếm và thay thế.

Vậy em hãy viết chương trình nhập vào từ bàn phím xâu ký tự s, tìm và thay thế tất cả các cụm ký tự s1 thành s2.

Ví dụ: Xâu gốc: s = "HS sinh lop 12 thi HS gioi"

s1= "HS"; s2= "Hoc sinh";

Xâu kết quả: S = "Hoc sinh lop 12 thi Hoc sinh gioi"

Hướng dẫn:

Cách 1:

Dùng hàm tìm kiếm s.find(s1) để xác định vị trí xuất hiện xâu s1 trong xâu s: vt = s.find(s1) Nếu vt != -1 tức là có xâu s1 trong xâu s thì thay thế xâu s1 bởi xâu s2 bằng hàm s.replace(vt,l1,s2); hoặc thay thế bằng lệnh xóa và chèn: s.erase(vt,l1); s.insert(vt,s2);

Lưu ý: Trong xâu có thể có rất nhiều cụm từ cần thay thế nên ta phải sử dụng lệnh while để thay thế hết: trong khi s.find(s1) != -1 thì tiếp tục thay thế ***Cách 2:***

Để duyệt hết tất cả các khả năng cần thay thế ta sử dụng lệnh for kết hợp lệnh s.substr() để sao chép từng đoạn con có độ dài bằng độ dài xâu s1 rồi so sánh với s1. Nếu kết quả bằng s1 thì ta tiến hành thay thế bằng các câu lệnh tương tự cách 1.

Chương trình:

Cách 1:

```cpp
#include <bits/stdc++.h>
using namespace std;
string s,s1,s2;
int l,ll,vt;
/*************************/
int main()
{
     getline(cin,s);
     getline(cin,s1);
     getline(cin,s2);
/*************************/
     ll=s1.length();
     vt=s.find(s1);
     while(vt!=-1)
          {    s.erase(vt,ll);
               s.insert(vt,s2);
               vt=s.find(s1);
          }
/*************************/
cout<<s;
return 0;
}
```

Cách 2:

```cpp
#include <bits/stdc++.h>
using namespace std;
string s,s1,s2;
int l,ll,i;
/*************************/
int main()
{
     getline(cin,s);
     getline(cin,s1);
     getline(cin,s2);
/*************************/
     l = s.length();
     ll=s1.length();
     for (i=0; i<=l-ll; i++)
     if (s.substr(i,ll) == s1)
     s.replace(i,ll,s2);
/*************************/
cout<<s;
return 0;
}
```

Hướng dẫn Python:

Cách 1:

Dùng hàm tìm kiếm s.find(s1) để xác định vị trí xuất hiện xâu s1 trong xâu s: vt = s.find(s1) Nếu vt != -1 tức là có xâu s1 trong xâu s thì thay thế xâu s1 bởi xâu s2 bằng hàm s.replace(s1,s2);

Lưu ý: Trong xâu có thể có rất nhiều cụm từ cần thay thế nên ta phải sử dụng lệnh while để thay thế hết: trong khi s.find(s1) != -1 thì tiếp tục thay thế

Cách 2:

Để duyệt hết tất cả các khả năng cần thay thế ta sử dụng lệnh for kết hợp lệnh s.substr() để sao chép từng đoạn con có độ dài bằng độ dài xâu s1 rồi so sánh với s1. Nếu kết quả bằng s1 thì ta tiến hành thay thế bằng các câu lệnh tương tự cách 1.

Chương trình:

```
s=input()
s1=input()
s2=input()
l1=len(s1)
vt=s.find(s1)
while vt!=-1:
    s=s.replace(s1,s2)
    vt=s.find(s1)
print(s)
```

***Bài 24:* Chuyển đổi phông chữ:**

Khi soạn thảo và trình bày văn bản, việc chuyển đổi đoạn văn chữ hoa sang thường hay chữ thường sang chữ hoa là một nhu cầu rất phổ biến. Trong hệ soạn thảo văn bản Word, nếu ta dùng bảng mã TCVN3 thì việc chuyển đổi rất đơn giản, chỉ cần chọn lại phông chữ tương ứng còn khi dùng bảng mã Unicode thì ta phải sử dụng công cụ chuyển đổi của phần mềm hỗ trợ gõ chữ việt.

Em hãy viết chương trình nhập vào 1 xâu ký tự s, tạo xâu s1 chứa các tự trong xâu s nhưng ở dạng chữ hoa, tạo xâu s2 chứa các tự trong xâu s nhưng ở dạng chữ thường. Xuất kết quả xâu s1, s2 ra màn hình.

Hướng dẫn:

Cách 1: *Dùng hàm* toupper(), tolower() để chuyển đổi từng ký tự.

Duyệt từng ký tự của xâu s, với mỗi ký tự đó ta dùng hàm toupper() chuyển sang ký tự hoa rồi nối vào xâu s1, dùng hàm tolower() chuyển sang chữ thường rồi nối vào xâu s2.

Hoặc gán s1 = s; s2 =s; sau đó duyệt từng ký tự của xâu s1 dùng hàm toupper() chuyển sang ký tự hoa, duyệt từng ký tự của xâu s2 dùng hàm tolower() đê chuyển sang ký tự thường.

Cách 2: *Dùng hàm* transform() để chuyển đổi tất cả các ký tự.

- transform(s1.begin(), s1.end(),s1.begin(), ::toupper); chuyển tất cả các ký tự trong xâu s1 thành chữ hoa.

- transform(s2.begin(), s2.end(),s2.begin(), ::tolower); chuyển tất cả các ký tự trong xâu s2 thành chữ thường.

Chương trình:

Cách 1:

```
#include <bits/stdc++.h>
using namespace std;
string s,s1,s2; char c;
/**********************/
int main(){
getline(cin,s);
/**********************/
int l=s.length();
s1 = ""; s2 = s;
for(int i=0;i<l;i++)
    {
     c=toupper(s[i]);
     s1=s1+ c;
     s2[i]=tolower(s[i]);
    }
/**********************/
cout<<s1<<endl;
cout<<s2;
return 0;
}
```

Cách 2:

```
#include <bits/stdc++.h>
using namespace std;
string s,s1,s2;
/**********************/
int main(){
getline(cin,s);
/**********************/
int l=s.length();
s1 = s; s2 = s;
transform(s1.begin(),s1.end(),s1.begin(),::toupper)
transform(s2.begin(),s2.end(),s2.begin(),::tolower)
/**********************/
cout<<s1<<endl;
cout<<s2;
return 0;
}
```

Hướng dẫn: *Dùng hàm* **s.upper()** để chuyển đổi xâu s sang in hoa và hàm **s.lower()** để chuyển đổi xâu sang in thường

Chương trình:

```
s=input()
s1=s.upper()
s2=s.lower()
print(s1)
print(s2)
```

***Bài 25*: Công tác quản lý thi:**

Trong công tác tổ chức các kỳ thi, để đảm bảo tính khách quan, công bằng cho các thí sinh, một công đoạn hết sức quan trọng là lập danh sách dự thi và đánh số báo danh. Để đánh số báo danh, danh sách phải được sắp xếp với tên theo bảng chữ cái.

Em hãy viết chương trình giúp cán bộ làm công tác thi, nhập vào số nguyên n là số lượng thí sinh, sau đó đưa vào họ và tên của n thí sinh, đưa ra danh sách thí sinh đó nhưng đã được sắp xếp với tên theo bảng chữ cái.

Hướng dẫn:

- Dùng mảng hoten để lưu danh sách thí sinh: hoten[i] ghi họ và tên thí sinh i.
- Với mỗi thí sinh i, tách lấy phần tên lưu vào mảng: ten[i].
- Sắp xếp mảng tên theo thứ tự tăng dần, với mỗi lượt tráo đổi 2 phần tử trong mảng tên thì tráo đổi luôn 2 phần tử tương ứng trong mảng hoten.
- Xuất ra kết quả là danh sách thí sinh trong mảng hoten.

*** Chương trình tham khảo:**

```cpp
#include <bits/stdc++.h>
using namespace std;
string ten[1001],hoten[1001];
int n,l,j;
/**********************************/
int main()
{
cin>>n;
cin.ignore(32767,'\n');
/******* Nhập và tách tên *******/
for(int i=1;i<=n;i++)
    {
    getline(cin,hoten[i]);
    l=hoten[i].length();
    j=l-1;
    while(hoten[i][j]!=' '){j--;}
    ten[i]=hoten[i].substr(j+1,l-j);
    }
/************ Sắp xếp ************/
for(int i=1;i<=n-1;i++)
    for(int j=n-1;j>=i;j--)
        if(ten[j]>ten[j+1])
            {
            swap(ten[j],ten[j+1]);
            swap(hoten[j],hoten[j+1]);
            }
/**********************************/
cout<<"Danh sach thi sinh da sap xep la:" << endl;
for(int i=1;i<=n;i++){
cout<<hoten[i]<<endl;
}
return 0;
}
```

Hướng dẫn Python:

Dùng List hoten[] để lưu danh sách thí sinh: hoten[i] ghi họ và tên thí sinh i.

Dùng phương thức hoten.append(s) để thêm vào list hoten

- Với mỗi thí sinh i, tách lấy phần tên lưu vào list: ten[i].
- Sắp xếp list ten[] theo thứ tự tăng dần, với mỗi lượt tráo đổi 2 phần tử trong list ten thì tráo đổi luôn 2 phần tử tương ứng trong list hoten.
- Xuất ra kết quả là danh sách thí sinh trong list hoten.

*** Chương trình tham khảo:**

```
n=int(input())
ten=[]
hoten=[]
for i in range(n):
    s=input()
    hoten.append(s)
    s1=s.split()
    m=len(s1) -1
    ten.append(s1[m])
for i in range(n-1):
    for j in range(i+1,n):
        if ten[i]>ten[j]:
            ten[i],ten[j]=ten[j],ten[i]
            hoten[i],hoten[j]=hoten[j],hoten[i]
for i in range(n):
    print(hoten[i])
```

Bài 26: **Đếm số lần xuất hiện mỗi loại ký tự.**

Cho xâu s (có độ dài không vượt quá 10^3) chỉ gồm các ký tự từ 'A' đến 'Z'. Cho biết có bao nhiêu loại ký tự xuất hiện trong s và đưa ra một ký tự xuất hiện nhiều nhất trong s cùng với số lần xuất hiện của ký tự đó

Hướng dẫn:

- Các ký tự từ 'A' đến 'Z' có mã ASCII tương ứng từ 65 đến 90. Nên ta sử dụng mảng T gồm 91 phần tử với kiểu chỉ số từ 0 đến 90, kiểu giá trị *int* để lưu số lần xuất hiện: T[65] …T[90] tương ứng số lần xuất hiện ký tự 'A'… 'Z'.

- Lần theo các giá trị của mảng T trên đoạn từ 65 đến 90 ta được số lượng các ký tự khác nhau (tức số lượng phần tử có giá trị khác 0 trong mảng T) và tìm giá trị lớn nhất của mảng T ta sẽ tìm được ký tự xuất hiện nhiều lần nhất.

Về nguyên tắc, ta chỉ cần sử dụng mảng với 26 phần tử kiểu *int* với chỉ số từ 0 đến 25, khi đó với mỗi ký tự ta chuyển sang mã ASCII rồi dịch chuyển chỉ số 65 đơn vị để lưu vào đầu mảng

```cpp
using namespace std;
string s;
int i,li, l,d=0, m=0, T[91];
int main()
{
    cout<<"Nhap xau: ";
    getline(cin, s);
    /*****************************/
    l=s.size();
    memset(T,0,sizeof(T));
    for (i=0; i<l; i++)
    T[s[i]]++;
    /*****************************/
    for (i=65; i<=90; i++)
        if(T[i]!=0)
        { d++;
          if(T[i]> m)
           {   m = T[i];
               li=i;
           }
        }
    /*****************************/
   cout<<"So loai KT khác nhau " <<d<<endl;
   cout<<"Ky tu "<<char(li);
cout<<" xuat hien nhieu lan nhat "<<m<< "lan";
return 0;
}
```

Code với Python:

```python
s=input()
a=[0]*91
for i in range(len(s)):
    a[ord(s[i])]=a[ord(s[i])]+1
m=max(a)
dem=0
for i in range(len(a)):
    if a[i]!=0:
        dem=dem+1
    if a[i]==m:
        ch=chr(i)
print("CO ", dem," LOAI KI TU XUAT HIEN TRONG XAU")
print("KI TU XUAT HIEN NHIEU NHAT LA", ch, m, "LAN")
```

Bài 27: Mã hóa Xê Da (sách bài tập tin học 11)

Để giữ bí mật người ta phải mã hóa các thông tin trước khi truyền đi hoặc lưu trữ. Một trong những cách mã hóa sớm nhất được sử dụng rộng rãi thời cổ đại là cách mã hóa do Xê Da đề xuất: trong thông điệp, người ta thay đổi chữ cái bằng chữ cái đứng sau

nó K vị trí trong bảng chữ cái. Việc tìm kiếm thay thế được tiến hành vòng tròn theo bảng chữ cái. Nếu bảng chữ cái có N chữ, thì sau chữ cái thứ N-1 là chữ cái N, sau chữ cái N là chữ cái thứ nhất,… Cách mã hóa này gọi là mã Xê Da. Các kí tự ngoài bảng chữ cái vẫn được giữ nguyên.

Ví dụ, bảng chữ cái tiếng Anh có 26 chữ cái. Nếu K = 2 thì có nghĩa là a được thay thế bằng c, b được thay thế bằng d, . . ., y được thay thế bằng a, z được thay thế bằng b. Các chữ cái in hoa sẽ được thay thế bằng chữ cái in hoa tương ứng. Trong trường hợp này, từ 'TIN HOC' sẽ được mã hóa thành 'VKP JQE' .

Hãy lập trình: Nhập vào từ bàn phím số nguyên K (1<K<=26) và xâu S không quá 255 kí tự. Mã hóa theo quy tắc mã Xê Da và đưa kết quả ra màn hình.

Hướng dẫn:

Để dễ xử lý hơn khi dịch k ký tự theo vòng tròn, ta dùng thêm 2 biến phụ S1 để lưu 2 lần bảng chữ cái in hoa. S2 lưu 2 lần bảng chữ cái in thường.

Duyệt từng ký tự trong xâu s, tìm vị trí xuất hiện của nó trong xâu s1, s2. Nếu tìm thấy thì ta lấy ký tự cách nó k ký tự trên mảng tương ứng.

```
#include <bits/stdc++.h>
using namespace std;
string s, s1="", s2="", kq="";
int k,l, vt;
int main()
{    cout<<"nhap do dich k:";
     cin>>k;
     cin.ignore(32767, '\n');
     cout<<"Nhap xau: ";
     getline(cin, s); l=s.size();
     /*****************************/
     for(int i=65;i<=90;i++)
          {    s1 = s1 + char(i);
               s2 = s2 + char(i+32);
          }
          s1=s1 + s1 + s2 + s2;
     /*****************************/
     for (int i=0; i<l; i++)
          {vt=s1.find(s[i]);
           if (vt==-1)
               kq = kq+s[i];
           else kq=kq + s1[vt+k];
           }
     cout<<"Xau ket qua:" <<kq;
return 0;
}
```

Code với Python:

```
s=input()
k=int(input())
s1="abcdefghijklmnopqrstuvwxyz"
s1=s1+s1
s2="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
s2=s2+s2
s3=""
for i in range(len(s)):
    if s[i] in s1:
        vt=s1.find(s[i])
        s3=s3+s1[vt+k]
    if s[i] in s2:
        vt=s2.find(s[i])
        s3=s3+s2[vt+k]
print(s3)
```

Bài 28: Xâu con dài nhất

Cho xâu S, tìm xâu con liên tục dài nhất của xâu S mà không chứa chữ số nào. Kết quả đưa ra vị trí đầu và xâu con đó. Nếu có nhiều xâu con dài nhất thì đưa ra xâu đầu tiên.

Hướng dẫn:

Có rất nhiều giải thuật để duyệt tìm xâu con dài nhất thỏa mãn một điều kiện nào đó.

Cách 1: Duyệt vét cạn thử với mọi xâu con để tìm xâu con dài nhất.

Sử dụng hai vòng *for* lồng nhau duyệt từ đầu đến hết xâu với mỗi cặp chỉ số đầu i và chỉ số j. Ta kiểm tra có thỏa mãn điều kiện hay không, rồi so sánh tìm xâu con thỏa mãn dài nhất.

Cách 2: Duyệt theo các xâu con với độ dài từ lớn đến bé.

Ta duyệt các xâu con với độ dài từ độ dài xâu mẹ giảm dần. Khi gặp xâu con đầu tiên thỏa mãn điều kiện thì đây chính là xâu cần tìm và ta kết thúc việc duyệt bằng lệnh break;

Cách 3: Duyệt phát triển xâu con:

Lần lượt duyệt lần lượt các ký tự trong xâu, dùng biến lmax để lưu độ dài xâu lớn nhất đã tìm được, nếu gặp ký tự thỏa mãn (không phải số) ta tăng dần độ dài xâu con. Khi gặp ký tự không thỏa mãn (ký tự số) thì ta thu được một xâu con thỏa mãn điều kiện, đem độ dài xâu này so sánh với lmax đã lưu để cập nhật lại lmax mới.

Với bài toán này ta sử dụng cách 3 duyệt theo phát triển xâu con với độ phức tạp o(n).

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
string s,kq;
int i, l, vtc=0, lmax=0, d=0;
int main()
{
     cout<<"Nhap xau: ";
     getline(cin, s);
     s=s+"0";
     /*****************************/
     l=s.size();
     for(i=0; i<l; i++)
     {if (isdigit(s[i]))
       {if (d>lmax)
          {    vtc=i;
               lmax=d;
               d=0;}
       }
     else d++;
    }
     /*****************************/
     kq=s.substr(vtc-lmax,lmax);
     cout<<"do dai xau:"<<lmax<<endl;
     cout<<"Xau ket qua:" <<kq;
return 0;
}
```

Hướng dẫn Python:

Có rất nhiều giải thuật để duyệt tìm xâu con dài nhất thỏa mãn một điều kiện nào đó.

Cách 1: Duyệt vét cạn thử với mọi xâu con để tìm xâu con dài nhất.

Sử dụng hai vòng *for* lồng nhau duyệt từ đầu đến hết xâu với mỗi cặp chỉ số đầu i và chỉ số j. Ta kiểm tra có thỏa mãn điều kiện hay không, rồi so sánh tìm xâu con thỏa mãn dài nhất.

Cách 2: Duyệt theo các xâu con với độ dài từ lớn đến bé.

Ta duyệt các xâu con với độ dài từ độ dài xâu mẹ giảm dần. Khi gặp xâu con đầu tiên thỏa mãn điều kiện thì đây chính là xâu cần tìm và ta kết thúc việc duyệt bằng lệnh break;

Cách 3: Duyệt phát triển xâu con:

Lần lượt duyệt lần lượt các ký tự trong xâu, dùng biến lmax để lưu độ dài xâu lớn nhất đã tìm được, nếu gặp ký tự thỏa mãn (không phải số not s[i].isdecimal()) ta tăng dần độ dài xâu con. Khi gặp ký tự không thỏa mãn (ký tự số) thì ta thu được một xâu con

thỏa mãn điều kiện, đem độ dài xâu này so sánh với lmax đã lưu để cập nhật lại lmax mới.

Với bài toán này ta sử dụng cách 3 duyệt theo phát triển xâu con với độ phức tạp o(n).

Chương trình:

```
s=input()
kq=""
s1=""
vt=0
lmax=0
l=0
i=0
while i<=len(s)-1:
    if not s[i].isdecimal():
        s1=s1+s[i]
        l=l+1
    else:
        if l>lmax:
            lmax=l
            kq=s1
            s1=""
            l=0
            vt=i-lmax
    i=i+1
print(vt)
print(kq)
```

Bài 29: Đếm xâu con *(có chồng lấn và không chồng lấn lên nhau)*

Viết chương trình nhập vào 2 xâu ký tự s1, s2. Hãy cho biết số lần xuất hiện xâu s2 trong xâu s1 không chồng lấn và số lần xuất hiện xâu s2 trong s1 có chồng lấn.

Ví dụ: s1= “abababababaab” S2= “aba”

Kết quả: Số lần xuất hiện không chồng lấn là: 2 Số lần xuất hiện có chồng lấn là: 4

Hướng dẫn:

Cách 1: Sử dụng hàm tìm kiếm sự xuất hiện xâu s2 trong xâu s1. Khi tìm thấy s2 trong s1, ta xóa (hoặc thay thế bởi xâu “”) toàn phần tìm thấy hoặc 1 ký tự tại vị trí tìm thấy rồi tìm tiếp. Nếu dung câu lệnh find() có tham số vị trí bắt đầu thì không cần xóa.

Cách 2: Ta có thể dùng lệnh while duyệt sao chép đoạn con của s1 có độ dài bằng l2 (độ dài xâu s2) để so sánh với s2, trong yêu cầu 1 nếu tìm thầy thì ta nhảy qua l2 vị trí tiếp theo.

Chương trình giải theo các 1:

```cpp
#include <bits/stdc++.h>
using namespace std;
string s1,s2;
int i,k, l1, l2,d1=0, d2=0;
int main()
{
    cout<<"Nhap xau s1: ";
    getline(cin, s1);
    cout<<"Nhap xau s2: ";
    getline(cin, s2);
    /*****************************/
    l2=s2.size();
    k=0;    d1=0; d2=0;
    i=s1.find(s2,k);
    while (i!=-1)
    {
        d2++;
        k=i+1;
        i=s1.find(s2,k);
    }
    i=s1.find(s2);
    while (i!=-1)
    {
        d1++;
        s1.replace(0,i+l2,"");
        i=s1.find(s2);
    }
    /*****************************/
    cout<<"so lan k chong lan"<<d1<<endl;
    cout<<"so lan chong lan"<<d2<<endl;
return 0;
}
```

Hướng dẫn Pyhon:

Cách 1: Sử dụng hàm tìm kiếm sự xuất hiện xâu s2 trong xâu s1. Khi tìm thấy s2 trong s1, ta xóa (hoặc thay thế bởi xâu “”) toàn phần tìm thấy(đối với đếm số lần không chồng lấn) hoặc 1 ký tự tại vị trí tìm thấy(đối với đếm số lần có chồng lấn) rồi tìm tiếp.

Cách 2: Ta có thể dùng lệnh while duyệt sao chép đoạn con của s1 có độ dài bằng l2 (độ dài xâu s2) để so sánh với s2, trong yêu cầu 1 nếu tìm thầy thì ta nhảy qua l2 vị trí tiếp theo.

Chương trình giải theo các 1:

```
s1=input()
s2=input()
d=0
d1=0
s3=s1
s4=s1
while s3.find(s2)!=-1:
    d=d+1
    vt=s3.find(s2)
    s3=s3[vt+len(s2):len(s3)]
while s4.find(s2)!=-1:
    d1=d1+1
    vt=s4.find(s2)
    s4=s4[vt+1:len(s4)]
print(d)
print(d1)
```

Bài 30: Đảo từ trong xâu

Cho xâu ký tự s, từ là một xâu con liên tiếp không chứa dấu cách. Một xâu ký tự có thể gồm nhiều từ.

Em hãy viết chương trình nhập vào xâu ký tự s. Đưa ra xâu đó với các từ được viết theo chiều ngược lại.

Ví dụ: s = “di xe dap” thì kết quả là: kq =“dap xe di”

Hướng dẫn:

Cách 1: Duyệt trên từng ký tự của xâu: Duyệt ngược xâu từ sau ra trước, mỗi lần tìm thấy dấu cách hoặc ký tự đầu tiên của xâu, thì đọc các ký tự từ vị trí đó đến vị trí dấu cách đã gặp trước đó.

Cách 2: Duyệt tìm vị trí chứa các dấu cách copy xâu con giữa 2 dấu cách rồi nối vào xâu kết quả**.**

Chương trình:

Chương trình này được viế theo cách 2

```cpp
#include <bits/stdc++.h>
using namespace std;
string s, kq="", tu;
int k,l;
int main()
{
     cout<<"Nhap xau: ";
     getline(cin, s);
     s=" "+s+" ";
     /*****************************/
    k=0;
    l=s.find(" ",1);
    while (l!=-1)
    { tu = s.substr(k+1,l-k);
      kq=tu +kq;
      k=l;
      l=s.find(" ",k+1);
    }
     /*****************************/
    cout<<"Xau ket qua:" <<kq;
return 0;
}
```

Hướng dẫn với Python: Dùng hàm str() để chuyển n dạng số qua dạng xâu s. Chuyển từng kí tự của s qua dạng số (sử dụng hàm int()). Kiểm tra nếu số chẵn thì cộng vào biến Tong ta được kết quả

Chương trình:

```python
n=int(input())
s=str(n)
tong=0
for ch in s:
    if int(ch)%2==0:
        tong=tong+int(ch)
print(tong)
```

Bài 31: Tổng chữ số

Cho số nguyên dương N (số chữ số của N không quá 10^6). Hãy tính tổng các số chẵn trong dãy.

Hướng dẫn: Số nguyên N có giá trị quá lớn nên ta không thể xử lý N theo kiểu số. vì vậy ta đọc N vào xâu ký tự để giải quyết. Để kiểm tra tính chẵn lẻ và tính tổng các chữ số, ta xử lý trên mã ASCII của các ký tự số.

Chương trình:

```cpp
#include <bits/stdc++.h>
using namespace std;
string s;
int i,l, tong;
int main()
{
    cout<<"Nhap so nguyen ";
    getline(cin, s);
/*****************************/
    l=s.size();
    tong = 0;
    for(i=0; i<l; i++)
        if (s[i]%2==0)
        tong=tong+s[i]-48;
/*****************************/
cout<<"Tong cac chu so chan:"<<tong;
return 0;
}
```

Hướng dẫn với Python: Dùng hàm str() để chuyển n dạng số qua dạng xâu s. Chuyển từng kí tự của s qua dạng số (sử dụng hàm int()). Kiểm tra nếu số chẵn thì cộng vào biến Tong ta được kết quả

Chương trình:

```python
n=int(input())
s=str(n)
tong=0
for ch in s:
    if int(ch)%2==0:
        tong=tong+int(ch)
print(tong)
```

Bài 32: SỐ MỘT

Xét các số nguyên dương K có dạng biểu diễn ở hệ thập phân có N chữ số và chỉ bao gồm các chữ số 1, ví dụ với N = 2 có K = 11, N = 3 có K = 111.

Yêu cầu: Với N cho trước, tính K^2 ($1 \le N \le 10^3$)

Ví dụ: Với N = 9 thì K =111111111 Kết quả: K^2 = 12345678987654321

Hướng dẫn:

Vì số lượng chữ số N của số K là rất lớn, nên ta không thể xử lý theo dữ liệu kiểu số mà phải xử lý theo dữ liệu kiểu xâu.

Nhớ nguyên tắc thực hiện phép nhân một số với số có nhiều chữ số: ta lấy số đó nhân lần lượt với mỗi chữ số của số nhân từ phải sang trái, với mỗi kết quả ta đặt dịch sang trái thêm một hàng, cộng dọc các kết quả theo hàng từ phải sang trái ta có kết quả.

Với bài toán này số K chỉ gồm các chữ số 1, nên kết quả của hàng i (i tăng dần từ 1 đến n sau đó giảm dần từ n-1 về 1) chính là tổng của i số 1 và giá trị nhớ của hàng bên trái.

```
#include <bits/stdc++.h>
using namespace std;
long n,i,cs, nho=0;
string  kq="";
/*****************************/
int main()
{
     cout<<"Nhap so chu so cua k:";
     cin>>n;
     nho = 0;
     for(int i=1;i<=n;i++)
    {
         cs = (i + nho)%10;
         kq = char(cs+48) + kq;
         nho = (i+nho)/10;
    }
    for(int i=n-1; i>=1; i--)
    {
         cs = (i + nho)%10;
         kq = char(cs+48) + kq;
         nho = (i+nho)/10;
    }
/*****************************/
     cout<<"Ket qua la:"<<kq;
return 0;
}
```

Hướng dẫn: **Python** có thể xử lí dữ liệu vơi N rất lớn nên với bài này ta chỉ cần tạo xâu k gồm N kí tự 1 (k=”1”*n)

Chuyển k qua dạng số int(k) sau đó in kết quả (k*k)

```
n=int(input())
s="1"*n
k=int(s)
print(k*k)
```

Bài 33: Ghép số

Cho n số nguyên dương a1, a2, . . .,an (1 < n ≤ 100), mỗi số không vượt quá 10^9 . Từ các số này người ta tạo ra một số nguyên mới bằng cách ghép tấtcả các số đã cho, tức là viết liên tiếp các số đã cho với nhau. Ví dụ, với n = 4 và các số 123, 124, 56, 90 ta có thể tạo ra các số mới sau: 1231245690, 1241235690,

5612312490, 9012312456, 9056124123,... Có thể dễ dàng thấy rằng, với n = 4,

ta có thể tạo ra 24 số mới. Trong trường hợp này, số lớn nhất có thể tạo ra là 9056124123.

Yêu cầu: Cho n số nguyên số a1, a2, . . .,an nằm trên cùng 1 dòng. Hãy xác định số lớn nhất có thể tạo ra khi ghép các số đã cho thành một số mới.

Hướng dẫn:

Lưu các số dưới dạng mảng kiểu xâu, thực hiện sắp xếp mảng theo thứ tự tăng dần theo tiêu chí sắp xếp là phần tử a[i] đứng trước phần từ a[j] khi (a[i] ghép với a[j]) > (a[j] ghép với a[i]).

Chương trình:

here | 2-2-3 bai3.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
string a[101];
int n,i;
string s,x, kq="";
/*****************************/
int main()
{
    cout<<"Nhap day n so: ";
    getline(cin, s);
    stringstream ds(s);
    i=0;
    while (ds>>x)
    {   i++;
        a[i]=x;
    }
    n=i;
/*****************************/
for(int i=1;i<=n-1;i++)
    for(int j=i+1;j<=n; j++)
        if(a[i]+a[j]<a[j]+a[i])
           swap(a[i],a[j]);
/*****************************/
    for(i=1; i<=n; i++)
        kq=kq+a[i];
    cout<<"So lon nhat la:"<<kq;
return 0;
}
```

Code với Python:

```
n=int(input())
a=list(map(int,input().split()))
b=[]
for i in range(n):
    b.append(str(a[i]))
b.sort()
s=""
i=n-1
while i>=0:
    s=s+b[i]
    i=i-1
print(s)
```

Bài 34: Xóa số:

Cho 2 số nguyên dương N và k (số chữ số của N không quá 10^6, k nhỏ hơn số chữ số của N) Yêu cầu: Hãy tìm cách xóa k chữ số của N để các chữ số còn lại (vẫn dữ nguyên thứ tự) tạo thành một số có giá trị lớn nhất.

Hướng dẫn:

Vì giá trị của N quá lớn nên ta phải giải quyết với kiểu dữ liệu kiểu xâu. Việc xóa đi k chữ số để lấy các chữ số còn lại thì kết quả cũng tương đương với việc chọn N-k chữ số đúng trật tự để thu được kết quả là số lớn nhất.

Để tìm chữ số lớn nhất cần lấy, ta xây dựng hàm vtm(i,j) đưa vào vị trí đầu và cuối i-j, lấy ra vị trí phần tử lớn nhất đầu tiên thuộc đoạn i-j. Lần đầu tiên lấy phần tử max chỉ số vt trong đoạn [0, k], sau đó tìm max với chỉ số đầu [vt+1, k+1]…

Chương trình:

```cpp
#include <bits/stdc++.h>
using namespace std;
string s, kq="";
long i,k,vt, d, tong;
long vtm(long i, long j)
{   long p;
    p=i;
    for(int l=i; l<=j; l++)
        if (s[p]<s[l])
            p=l;
    return p;
  }
int main()
{
    cout<<"Nhap so nguyen N: ";
    getline(cin, s);
    cout<< "nhap k: ";
    cin>>k;
/*****************************/
    i=0; d=s.size()-k;
    for(long l=1; l<=d; l++)
    {
        vt=vtm(i,k);
        kq=kq+s[vt];
        i=vt+1;
        k=k+1;
    }
/*****************************/
cout<<"So lon nhat con lai là:"<<kq;
return 0;
}
```

Code với Python:

```
#Nhập dữ liệu
S = input("Nhập số nguyên dương N = ")
k = int(input("Nhập số chữ số cần xóa k = "))

#Tham lam
N = [x for x in S]
while k>0:
    timthay = False
    for i in range(len(N)-1):
        if N[i] < N[i+1]:
            timthay = True
            N.pop(i)
            k = k - 1
            break
    if timthay == False:
        break
while k>0:
    N.pop(len(N)-1)
    k = k-1

#Ghi kết quả
print("Kết quả N sau khi xóa: "+"".join(N))
```

Bài 35. Thành phố xanh sạch

(Đề thi HSG lớp 12 - Tỉnh Nghệ An - Năm học 2021-2022)

Thành phố của Bình có nhiều con đường được trồng cây xanh. Mỗi cây xanh được đặt tên bằng một chữ cái Latinh hoa. Theo Bình, một đoạn đường được gọi là xinh đẹp Nếu đoạn đường đó chỉ trong một loại cây(Tức là trên đoạn đường đó, các cây được trồng ở vị trí liên tiếp, có tên giống nhau và thuộc một con đường)

Yêu cầu:

Hãy giúp Bình tìm đoạn đường xanh đẹp gồm nhiều cây xanh nhất trong tất cả các con đường của thành phố.

Dữ liệu: cho trong file **Xanhdep.Inp** gồm:

- Dòng 1 ghi số nguyên dương N (N≤100) là số con đường trong thành phố.

- N dòng tiếp theo, mỗi dòng ghi một xâu kí tự gồm các chữ cái Latinh hoa mô tả tên của các cây xanh được trồng liên tiếp từ đầu con đường đến cuối con đường. số lượng cây trên mỗi con đường không lớn hơn 10^4.

*Kết quả: G*hi ra file **Xanhdep.Out**

Một số nguyên là số lượng cây xanh trên đoạn đường xanh đẹp gồm nhiều cây xanh nhất trong các con đường của thành phố

Ví dụ:

Xanhdep.Inp	Xanhdep.Out	Giải thích
3 ABBBABAAH **HHHHH**AHHHA EEAE	5	đoạn đường xanh đẹp gồm nhiều cây nhất là 5 cây(HHHHH) tong con đường thứ 2

Giới hạn:

- Có 80% số test ứng với N≤ 10 và số cây trên mỗi con đường ko quá 100 cây;
- Có 20% số test không có giới hạn gì thêm.

Hướng dẫn giải

Sử dụng kĩ thuật duyệt phát triển đoạn con với chỉ số đầu i và chỉ số cuối j. Trong khi phần tử thứ j còn bằng phần tử thứ i thì giá trị j được tăng lên. Khi gặp phần tử khác phần tử thứ i ta so sánh độ dài đoạn con vừa tìm được với độ dài max trước đó. Nếu độ dài đoạn con tìm được lớn hơn độ dài max trước đó thì độ dài max sẽ được thay đổi thành độ dài đoạn con vừa tìm được, và chỉ số đầu ibắt đầu đoạn mới bằng j.

Chương trình:

art here | *Xanhdep.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;;
int n,res=0,sd;
string s;
/***********************/
void open()
{     freopen("Xanhdep.inp","r",stdin);
      freopen("Xanhdep.out","w",stdout);
      ios_base::sync_with_stdio(NULL);
      cin.tie(0);
      cout.tie(0);
}
/************************************/
int solve(string s)
{     int l=s.length(),i=1,j=1,res=0,d;
      s=' '+s+' ';
      while(j<=l)
      {     while(s[j]==s[i]) j++;
            d=j-i;
            if(d>res) res=d;
            i=j;  j++;
      }     return res;
}
/**********************************/
int main()
{     open();
      cin>>n;
      for(int i=1;i<=n;i++)
      {     cin>>s;
            sd=solve(s);
            if(sd>res) res=sd;
      }
      cout<<res;
      return 0;
}
```

Code với Python:

```python
# Đọc dữ liệu
with open("XANHDEP.INP", "r") as fi:
    N = int(fi.readline())
    X = [s for s in fi.read().split('\n')]

# Tìm đoạn giống nhau dài nhất trong mỗi xâu
def xuly(s):
    d = 0
    n = len(s)
    j = 0
    for i in range(n):
        while j<n and s[i]==s[j]:
            j = j + 1
        d = max(d,j-i)
    return d

# Tìm kết quả
ans = 0
for i in range(N):
    ans = max(ans, xuly(X[i]))

with open("XANHDEP.OUT","w") as fo:
    fo.write(str(ans))
```

Bài 36: NGÔN NGỮ MUMBA

Mỗi từ trong ngôn ngữ của bộ tộc Mumba là một xâu kí tự hình thành chỉ từ hai kí tự ***a*** và ***b*** theo quy tắc sau:

- Không chứa 2 kí tự ***b*** liên tiếp
- Không có ba từ con giống nhau đứng liên tiếp trong một từ, như vậy ***aaa*** không phải là một từ Mumba (có 3 từ con ***a*** liên tiếp), ***aabababa*** cũng không phải là một từ Mumba (có 3 từ con ***ab*** liên tiếp), còn ***aba*** là một từ Mumba.

Yêu cầu:

Cho *n* xâu kí tự, mỗi xâu chỉ gồm các kí tự chữ cái tiếng Anh (‘a’..’z’), có độ dài không quá 25 kí tự. Hãy đếm xem trong *n* xâu đã cho thì có bao nhiêu xâu là một từ trong ngôn ngữ Mumba.

Dữ liệu vào: vào từ file văn bản Mumba.INP gồm:

- Dòng đầu ghi hai số nguyên dương n ($n \leq 100$).
- N tiếp theo, mỗi dòng ghi một xâu kí tự.

Kết quả: ghi ra file văn bản Mumba.OUT gồm 1 số duy nhất là số lượng từ Mumba đếm được.

Mumba.INP	Mumba.OUT
3 aabababa cba abaa	1

Hướng dẫn:

Với mỗi xâu ta chỉ cần duyệt và kiểm tra xem trên mỗi xâu có tồn tại các kí tự nào khác ‘a’,’b’ không và có tồn tại các xâu con "aaa", "bb", "ababab" hay không.

tart here | *MUMBA.cpp

```
#include<bits/stdc++.h>
using namespace std;
# define tep "bai2"
int t,dem = 0;   string s;
/**********************/
void open()
{    freopen("MUMBA.inp","r",stdin);
     freopen("MUMBA.out","w",stdout);
  }
/************************************/
bool test(string s)
{    int l=s.length(),k;
     string x;
     for(int i=0;i<l;i++)
if(s[i]!='a' && s[i]!='b') return false;
     x="aaa";     k=s.find(x);
     if(k>=0) return false;
     x="bb";      k=s.find(x);
     if(k>=0) return false;
     x="ababab"; k=s.find(x);
     if(k>=0) return false;
     return true;
}
/**********************************/
int main()
{    open();   cin>>t;
     getline(cin,s);
     for(int i=1;i<=t;i++)
     {    getline(cin,s);
          if(test(s)==true)
          { dem++;    cout<<i<<' ';   }
     }
     cout<<dem;
     return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("MUMBA.INP", "r") as fi:
    N = int(fi.readline())
    X = [x for x in fi.read().split('\n')]

# Kiểm tra từ MUMBA
def MUMBA(S):
    na = S.count('a')
    nb = S.count('b')
    n = len(s)
    if na + nb < n or S.count("aaa") > 0 or S.count("bb") > 0:
        return 0
    d = 2
    while d*3<=n:
        for i in range(n-d*3+1):
            if s[i:i+d-1] == s[i+d:i+2*d-1] == s[i+2*d:i+3*d-1]:
                return 0
        d = d + 1
    return 1

# Đếm từ MUMBA
ans = 0
for s in X:
    ans = ans + MUMBA(s)
with open("MUMBA.OUT","w") as fo:
    fo.write(str(ans))
```

4) Bài tập chủ đề đệ quy, quay lui

Bài 37: Dãy chia hết

Xét một dãy gồm N số nguyên tuỳ ý. Giữa các số nguyên đó ta có thể đặt các dấu + hoặc - để thu được các biểu thức số học khác nhau. Ta nói dãy số là chia hết cho K nếu một trong các biểu thức thu được chia hết cho K. Hãy viết chương trình xác định tính chia hết của một dãy số đã cho.

Dữ liệu vào: Lấy từ một file văn bản có tên là DIV.INP có cấu trúc như sau:

- Dòng đầu là hai số N và K ($2 \le N \le 10\,000$, $2 \le K \le 100$), cách nhau bởi dấu trống.
- Các dòng tiếp theo là dãy N số có trị tuyệt đối không quá 10 000 cách nhau bởi dấu trống hoặc dấu xuống dòng.

Dữ liệu ra: Ghi ra file văn bản DIV.OUT số 1 nếu dãy đã cho chia hết cho K và

số 0 nếu ngược lại.

Ví dụ:

DIV.INP	DIV.OUT	DIV.INP	DIV.OUT
4 6 1 2 3 5	0	4 7 1 2 3 5	1

Hướng dẫn:

- Khởi tạo tong=a[i];
- Duyệt các phần tử từ 2 đến n bằng hàm đệ quy.
- Với mỗi phần tử i ta đề xuất 2 khả năng của j:

  j=0 có tương ứng phép cộng:

  tong=tong + a[i], j=1 có tương ứng

  phép trừ: tong=tong - a[i],

  khi i=n thì kiểm tra, nếu chia hết thì ghi nhận và

  thoát ra nếu không thì lùi lại.

rt here | *DiV.cpp

```
#include<bits/stdc++.h>
using namespace std;
int n, k,kt=0, tong,a[10001];
int Try(int i)
{   for( int j=0;j<=1;j++)
        {     if(j==0) tong=tong+a[i];
              else     tong=tong-a[i];
         if(i==n)
              {if (tong%k==0)
                   {    kt=1; return 0;}
              }
         else Try(i+1);
         if(j==0)tong=tong-a[i];
         else tong = tong + a[i];
        }
    }
int main()
{freopen("Div.inp","r",stdin);
 freopen("Div.out","w",stdout);
cin>>n>>k;
for(int i=1; i<=n; i++)
    cin>>a[i];
tong=a[1];
Try(2);
cout<<kt;
return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("DIV.INP", "r") as fi:
    N,K = map(int,fi.readline().split())
    A = [int(x) for x in fi.readline().split()]

# Quy hoạch động
F = []
for i in range(K):
    r = [0]*N
    F.append(r)

F[A[0]%K][0] = 1
for i in range(1,N):
    for k in range(K):
        s = (k+A[i])%K
        x = (k-A[i]+K)%K;
        F[s][i] = max(F[s][i],F[k][i-1])
        F[x][i] = max(F[x][i],F[k][i-1])

# Ghi kết quả
with open("DIV.OUT","w") as fo:
    fo.write(str(F[0][N-1]))
```

Bài 38: Xâu MOO *(Đề thi thử cụm Yên Thành NH 2021-2022)*

Dãy xâu MOO là dãy vô hạn xâu được định nghĩa như sau:

- $S_0 = 'moo'$
- $\forall k \geq 0$: $S_{k+1} = S_k \oplus 'mo \ldots o' \oplus S_k$,

trong đó xâu $'mo \ldots o'$ có $k + 3$ kí tự $'o'$; phép toán $\oplus$ là phép nối xâu.

Chẳng hạn, dưới đây là một số phần tử trong dãy:

- $S_0 = 'moo'$, o $S_1 = 'moomooomoo'$,
- $S_2 = 'moomooomoomoooomoomooomoo'$,

Yêu cầu:

Với mỗi số nguyên dương N, tất cả các xâu trong dãy có độ dài không nhỏ hơn N đều có kí tự thứ N giống nhau, hãy xác định kí tự đó.

Dữ liệu: Vào từ tệp MOO.INP**:**

Một dòng duy nhất ghi số nguyên N ($1 \leq N \leq 10^{12}$).

Kết quả: Ghi ra tệp MOO.OUT.

- Một dòng duy nhất ghi ký tự $'m'$ hoặc $'o'$.

Ví dụ

MOO.INP	MOO.OUT
11	**m**

Giới hạn: $1 \leq N \leq 10^9$

Gợi ý:

Đối với bài toán này ta có thể tạo ra xâu S với quy tắc trên cho đến khi độ dài của xâu lớn hơn hoặc bằng N. Khi đó ta chỉ cần cần xuất ra S[n]. Tuy nhiên với cách này ta chỉ giải quyết được với n< 10^9. Trường hợp n>10^9 lại không giải quyết được. Vậy ta có thể sử dụng đệ quy và tư tưởng tìm kiếm nhị phân để giải

quyết bài toán như sau.

Chương trình:

art here | *moo.cpp

```
#include<bits/stdc++.h>
using namespace std;
long long k,n, i, a[100] ;
/***************************/
void tao_mang(long long n)
{
    int j=0; a[j]=3;
    while (a[j]<n)
    {   j++;
        a[j]=2*a[j-1]+j+3;
    }
    k=j;
}
/***********************************/
void ghi(long long n)
{
    if (n==1) cout<<"m";
    else cout<<"o";
}
/***********************************/
void thu(long long k, long long n)
{   if (k==0) ghi(n);
    else
       { if ((a[k-1]<n)&&(n<=a[k-1]+k+3))
         {   n=n-a[k-1];
             ghi(n);
         }
        else
        {   if(n>a[k-1]+k+3)
                n=n-a[k-1]-k-3;
            thu(k-1,n);
        }
       }
}
/***********************************/
int main()
{   freopen("moo.inp","r",stdin);
    freopen("moo.out","w",stdout);
    cin>>n;
    tao_mang(n);
    thu(k,n);
    return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("MOO.INP","r") as fi:
    N = int(fi.readline())

# Tạo danh sách độ dài các xâu, xâu đầu tiên là a[1]
n = 0
a = [0]
while a[n] < N:
    t = a[n]*2 + n + 3
    a.append(t)
    n = n + 1

# Tìm kí tự thứ N bằng đệ quy
def TimKT(N,n):
    print(n,a[n],N)
    if a[n] == N:
        return 'o'
    if a[n] > N:
        return TimKT(N,n-1)
    if a[n] + n + 3 < N:
        return TimKT(N-a[n]-n-3,n-1)
    if N-a[n] == 1:
        return 'm'
    return 'o'

#Ghi kết quả
with open("MOO.OUT", "w") as fo:
    fo.write(str(TimKT(N,n)))
```

Bài 39: Tạo sơn tổng hợp.

Từ N loại sơn ban đầu có số hiệu là 1, 2, ..., N (1≤N≤9), người ta có thể tạo ra rất nhiều loại sơn tổng hợp khác nhau bằng cách trộn một số loại sơn nào đó lại với nhau theo một liều lượng nào đó của mỗi loại. Khi tham gia trộn để được một loại sơn tổng hợp nào đó, các loại sơn khác nhau được đưa vào từ các vị trí khác nhau và liều lượng của mỗi loại sơn là bao nhiêu phụ thuộc vào thứ tự vị trí đưa vào của loại sơn đó. Liều lượng của mỗi loại sơn mà khác nhau trong khi trộn thì cho ra các loại sơn tổng hợp khác nhau. Hãy liệt kê ra tất cả các phương án trộn M loại sơn (M≤N) trong N loại sơn đã cho để có được các loại sơn tổng hợp.

Dữ liệu vào: Tệp SON.INP gồm 2 số N, M.

Dữ liệu ra: Là tệp văn bản SON.OUT có cấu trúc: Mỗi dòng ghi số hiệu của M loại sơn theo thứ tự khi đưa vào trộn để tạo ra một loại sơn tổng hợp nào đó. Dòng cuối cùng ghi số lượng các loại sơn tổng hợp tạo ra.

Ví dụ:

SON.INP	SON.OUT
3 2	1 2 1 3 2 1 2 3 3 1 3 2 6

Hướng dẫn:

Đây là bài toán xuất ra tất cả chỉnh hợp chập M của N và số lượng chỉnh hợp. Vậy nên ta dùng thuật toán đệ quy quay lui để giải quyết bài toán. Bài này tương tự bài Hoán vị các số từ 1 đến n, chỉ khác ở chỗ điều kiện để xuất ra là i=M và cần dùng thêm biến dem để đếm số lượng loại sơn.

Chương trình:

there X Son.cpp X

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  long long n,m, dem =0 ,x[11],d[11];
4
5  void Xuat()
6  {   dem++;
7      for(int i=1;i<=m;i++) cout<<x[i];
8      cout<<'\n';
9  }
```

```
long long Try(long long i)
{    for(int j=1;j<=n;j++)
        if(d[j]==0)
        {x[i]=j; d[j]=1;
        if(i==m) Xuat();
        else Try(i+1);
        d[j]=0;
        }
}
int main()
{ freopen("son.inp","r",stdin);
  freopen("son.out","w",stdout);
cin>>n>>m;
Try(1);     cout<<dem;
return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("SON.INP", "r") as fi:
    N,M = map(int,fi.read().split())

X = [0]*M
S = set()
kq = ""
dem = 0
# Đệ quy sinh chỉnh hợp
def Chon(n):
    global kq, dem
    if (n==M):
        for i in range(M):
            kq = kq + str(X[i]) + " "
        kq = kq + '\n'
        dem = dem + 1
        return
    for i in range(1,N+1):
        if not (i in S) :
            X[n] = i
            S.add(i)
            Chon(n+1)
            S.discard(i)
Chon(0)
# Ghi kết quả
with open("SON.OUT", "w") as fo:
    fo.write(kq)
    fo.write(str(dem))
```

5) Bài tập chủ đề quy hoạch động:

Bài 40: Dãy con đan dấu dài nhất:

Cho dãy số nguyên A = a1,a2...,an. Hãy xoá đi ít nhất các phần tử để những phần tử còn lại tạo thành một dãy đan dấu có nhiều phần tử nhất.

Dữ liệu vào: file văn bản QHĐ.INP, gồm 2 dòng:

Dòng 1: ghi số N là số lượng phần tử của dãy N<=1000

Dòng2: ghi N số nguyên a_i cách nhau ít nhất một dấu cách

|ai|<=100000 Kết quả: ghi ra file văn bản QHD.OUT, gồm 2 dòng:

Dòng 1: ghi M là số lượng phần tử của dãy con

Dòng 2: ghi lần lượt giá trị của các số trong dãy A ban đầu

Ví dụ:

DanDau.INP	DanDau.OUT
7 1 4 -3 2 7 -9	4 1 -3 2 -9

Hướng dẫn:

Gọi f[i] là kết quả trả lời cho bài toán số lượng phần tử dãy con đan dấu dài nhất kết thúc tại a[i] là bao nhiêu (phần tử a[i] phải có mặt trong dãy và đứngsau cùng ấy)

Kết quả của bài toán gốc là

max(f[i]) Khởi tạo là f[1] =

1

Công thức truy hồi: f[i] = max(f[j<i],0)+ 1 với a[j]*a[i]<0

Chương trình:

art here X DanDau.cpp X

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[1001],f[1001],p[1001];
int res,vet[1001],vt;
/*********************************/
void open()
{   freopen("DanDau.inp","r",stdin);
    freopen("DanDau","w",stdout);
    ios_base::sync_with_stdio(NULL);
    cin.tie(0);
    cout.tie(0);
}
/*********************************/
int main()
{   open();  cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    f[1]=1; p[1]=0;
    for(int i=2;i<=n;i++)
    {   f[i]=1; p[i]=0;
        for(int j=1;j<=i-1;j++)
            if(a[j]*a[i]<0)
                if(f[j]+1>f[i])
                  {p[i]=j; f[i]=f[j]+1;}
    }    res=1;
    for(int i=1;i<=n;i++)
        if(f[i]>res)
        {   res=f[i]; vt=i;    }
    cout<<res<<'\n';
    for(int i=res;i>=1;i--)
      {   vet[i]=vt; vt=p[vt]; }
    for(int i=1;i<=res;i++)
        cout<<a[vet[i]]<<' ';
    return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("DANDAU.INP", "r") as fi:
    N = int(fi.readline())
    A = [int(x) for x in fi.read().split()]

# Quy hoạch động
F = [0]*(N)
P = [0]*(N)
for i in range(N):
    F[i] = 1
    P[i] = -1
    for j in range(i):
        if A[i]*A[j]<0 and F[i] < F[j]+1:
            F[i] = F[j] + 1
            P[i] = j

#Ghi kết quả
vmax = 0;
for i in range(N):
    if F[i]>F[vmax]:
        vmax = i;
kq = ""
while vmax > -1:
    kq = str(A[vmax]) + " " + kq
    vmax = P[vmax]
with open("DANDAU.OUT", "w") as fo:
    fo.write(str(max(F))+'\n'+kq)
```

Bài 41: Xếp Va ly

Có n đồ vật, vật thứ i có trọng lượng Ai và giá trị Bi. Hãy chọn ra một số các đồ vật, mỗi vật một cái để xếp vào 1 vali có trọng lượng tối đa W sao cho tổng giá trị của vali là lớn nhất.

Input: Cho trong file văn bản Valy.INP

- Dòng đầu tiên ghi hai số N, W (N,W≤100)
- N dòng tiếp theo, dòng thứ i ghi hai số a_i và b_i lần lượt là giá trị và thể tích của đồ vật thứ i (a_i, b_i≤100)

Output: Ghi ra file văn bản Valy.OUT

- Dòng đầu tiên ghi tổng giá trị lớn nhất có thể cho vào trong túi

Ví dụ:

valy.INP	valy.OUT

5 10 20 3 19 1 30 7 24 3 15 6	63 3 1 2 4

Hướng dẫn:

Gọi g[i] là giá trị lớn nhất của vali khi có trọng lượng $1 \le i \le w$ sau mỗi lần xét đồ vật có khối lượng m, giá trị x. Mỗi lần sẽ đọc cặp giá trị m và x của đồ vật, biến i duyệt từ i=w về i=m khi đó giá trị lớn nhất của mỗi trọng lượng g[i]=max(g[i],g[i-m]+x).

Chương trình:

art here | *Valy.cpp

```
#include<bits/stdc++.h>
using namespace std;
int n,w,g[101],x,m,kq=0;
/*****************************/
int main()
{   freopen("Valy.inp","r",stdin);
    freopen("Valy.out","w",stdout);
    cin>>n>>w;
    fill(g,g+w,0);
    for(int i=1;i<=n;i++)
    {
        cin>>x>>m;
        for(int j=w;j>=m;j--)
                g[j]=max(g[j],g[j-m]+x);
    }
    cout<<g[w];
    return 0;
}
```

Code với Python:

```
# Đọc dữ liệu
with open("VALY.INP", "r") as fi:
    N,W = map(int,fi.readline().split())
    A = [0]*(N+1)
    B = [0]*(N+1)
    for i in range(1,N+1):
        A[i],B[i] = map(int,fi.readline().split())

# Quy hoạch động
F=[]
for i in range(N+1):
    w = [0]*(W+1)
    F.append(w)
for i in range(1,N+1):
    for j in range(1,W+1):
        F[i][j] = max(F[i-1][j],F[i][j-1])
        if j>=B[i]:
            F[i][j] = max(F[i-1][j-B[i]]+A[i],F[i][j])
# Ghi kết quả
with open("VALY.OUT", "w") as fo:
    fo.write(str(F[N][W])+'\n')
```

Bài 42: Xâu con chung dài nhất

Xâu con của xâu X thu được bằng cách xóa đi một vài ký tự của X và giữ nguyên vị trí của các ký tự còn lại. Ví dụ: 'abc' là xâu con của xâu 'adcberc', và không phải là xâu con của xâu 'adcber'.

Cho hai xâu ký tự là X và Y.

Tìm xâu con chung có đồ dài lớn nhất của hai xâu X và Y. Input: **XCCDN.INP**

- Dòng 1 xâu X
- Dòng 2 xâu Y
- Cả 2 xâu X và Y có số lượng ký tự không quá 100

Output: XCCDN.OUT

- Xâu con chung dài nhất của hai xâu X, Y

XCCDN.INP	XCCDN.OUT
AGTXAGT GAXTA	GXT

Hướng dẫn:

Gọi a[i,j] là độ dài xâu con chung dài nhất của xâu X[i] gồm i kí tự phần đầu của X (Xi=X[1..i]) và xâu Y[j] gồm j kí tự phần đầu của Y (Yj=Y[1..j]). Ta có công thức quy hoạch động như sau:

a[0,j] = a[i,0] = 0

a[i,j] = a[i−1,j−1] + 1 nếu X[i]=Y[j]

a[i,j] = max(a[i−1,j], a[i,j−1]) nếu X[i]≠Y[j].

Chương trình:

rt here | *XCCDN.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
string x,y;
int a[105][105],lx,ly,dem;
/*************************************/
int main()
{   freopen("XCCDN.inp","r",stdin);
    freopen("XCCDN.out","w",stdout);
    getline(cin,x);      getline(cin,y);
    lx=x.length();       ly=y.length();
    x=' '+x;             y=' '+y;
    dem=1;
    for(int i=1;i<=ly;i++)
        for(int j=1;j<=lx;j++)
            if(x[j]==y[i])
                a[i][j]=a[i-1][j-1]+1;
            else a[i][j]=max(a[i-1][j],a[i][j-1]);
    for(int i=1;i<=ly;i++)
        if(a[i][lx]!=a[i-1][lx]) cout<<y[i];
    return 0;
}
```

Code với Python:

```python
# Đọc dữ liệu
with open("XCCDN.INP", "r") as fi:
    X,Y = fi.read().split('\n')

# Quy hoạch động
n = len(X)
m = len(Y)
F = list()
for i in range(n+1):
    f = [""]*(m+1)
    F.append(f)
for i in range(1,n+1):
    for j in range(1,m+1):
        if X[i-1] == Y[j-1]:
            F[i][j] = F[i-1][j-1] + X[i-1]
        elif len(F[i-1][j]) >= len(F[i][j-1]):
            F[i][j] = F[i-1][j]
        else:
            F[i][j] = F[i][j-1]

# Ghi kết quả
with open("XCCDN.OUT","w") as fo:
    fo.write(F[n][m])
```