MỤC LỤC

LỜI MỞ ĐẦU

Để phục vụ cho việc ôn luyện thi Olympic khối chuyên tin năm 2012, các thành viên của đội tuyển Olympic chuyên tin năm 2011, Khoa Công nghệ thông tin - Đại học Hàng Hải sẽ tổng hợp lại các bài tập đã giải được, bao gồm các bài tập trên trang Giải bài trực tuyến: http://vn.spoj.pl/problems/oi/, các bài thi Olympic các năm trước và một số bài tập khác.

Các bài ôn luyện sẽ được phân vào các phần khác nhau bao gồm:

- PHẦN I. SỐ HỌC
- PHẦN II. ĐỒ THỊ
- PHẦN III. QUY HOẠCH ĐỘNG
- PHẦN IV. CÁC BÀI TOÁN KHÁC

Ngôn ngữ lập trình được sử dụng trong các bài chủ yếu là C/C++.

Trong quá trình biên soạn sẽ không thể tránh khỏi những sai sót, đội tuyển rất hoan nghênh sự đóng góp từ tất cả các bạn. Mọi ý kiến phản hồi xin gửi về hòm thư: olptin@vimaru.edu.vn hoặc trungvdt49@gmail.com . Chân thành cảm ơn!

PHẦN I. SỐ HỌC

BÀI 1. 3620. Số phong phú - http://vn.spoj.pl/problems/NKABD/

Trong số học, số phong phú là các số mà tổng các ước số của số đó (không kể chính nó) lớn hơn số đó. Ví dụ, số 12 có tổng các ước số (không kể 12) là 1 + 2 + 3 + 4 + 6 = 16 > 12. Do đó 12 là một số phong phú.

Bạn hãy lập trình đếm xem có bao nhiêu số phong phú trong đoạn [L,R].

Dữ liệu

Gồm 2 số L, R (1 <= L <= R <= 10^5)

Kết quả

Gồm 1 số nguyên duy nhất là số số phong phú trong đoạn [L, R].

Chú ý

Có 50% số test có 1 <= L <= R <= 10^3

Ví dụ

Dữ liệu

1 50

Kết quả

9

Giải thích:

Từ 1 đến 50 có 9 số phong phú là: 12, 18, 20, 24, 30, 36, 40, 42, 48

Time: 1s

Giải

Cách giải thông thường là duyệt từng số nguyên từ L đến R, tính tổng các ước của số đó không kể chính nó và kiểm tra nếu tổng đó lớn hơn số đó thì tăng biến đếm lên 1.

Để liệt kê các ước và tính tổng tất cả các ước của một số a không kể a ta chỉ việc kiểm tra lần lượt các số từ 1 cho đến a/2, nếu a chia hết thi in ra ước và cộng thêm ước vào biến tổng, đoạn mã như sau:

```
int main()
{
    int a;
    int i;
    int tong=0;
    scanf("%d",&a);
    for(i=1;i<=a/2;++i)
    {
       if(a%i==0)
       {
          printf("%d ",i);
          tong += i;
       }
    }
    printf("\n%d",tong);
    system("pause");
    return 0;
}
```

Để ý một chút ta thấy, không nhất thiết phải duyệt từ 1 cho đến a/2 mà chỉ cần duyệt các số i chạy từ 2 cho đến căn bậc 2 của a là đủ, nếu a chia hết cho i thì tong = tong + i + a/i (chú ý là nếu a/i = căn bậc 2 của a thì tong = tong + i). Đoạn mã như sau:

```
#include <stdio.h>
#include <math.h>
int main()
{
```

```
    int a;
    int i;
    int tong=1;
    scanf("%d",&a);
    int cb2 = (int)sqrt(a);
    printf("Cac uoc cua %d la:\n 1 ",a);
    for(i=2;i<cb2;++i)
    {
       if(a%i==0)
       {
          printf("%d %d ",i,a/i);
          tong += i+a/i;
       }
    }
    if(a%cb2==0)
    {
       tong += cb2;
       printf("%d ",cb2);
       if(a != cb2*cb2)
       {
          tong+=a/cb2;
          printf("%d",a/cb2);
       }
    }
    printf("\nTong uoc cua %d = %d",a,tong);
    //system("pause");
    return 0;
}
```

Tuy nhiên nếu mỗi lần xét 1 số trong đoạn [L,R] lại chạy hàm kiểm tra xem số đó có phải là số phong phú không thì sẽ rất mất thời gian. Thay vì điều đó ta sẽ dùng thuật toán khá giống với tư tưởng của sàng nguyên tố với nhận xét nếu một số a viết dưới dạng i*j thì i và j chính là ước của a. Đoạn mã đầu tiên như sau (time=0.08s):

```
#include <stdio.h>
int a[100001];
int main()
{
    int l,r,i,j,k=0;
    scanf("%d%d",&l,&r);
    for(i = 1; i<=r; i++)
      a[i] = 1;
    for(i = 2; i<=r/2; i++)
    {
      for(j = 2; j*i<=r; j++)
        a[j*i] += i;
    }
    for(i=l; i<=r; i++)
      if(a[i] > i) k++;
    printf("%d",k);
    //system("pause");
```

```
        return 0;
    }
```

Đoạn chương trình trên xử lý vẫn chưa tối ưu, theo như nhận xét bên trên ta chỉ cần xét các số i chạy từ 2 cho đến căn bậc 2 của R là đủ. Đoạn mã tiếp theo (time=0.04s):

```
#include<stdio.h>
#include<math.h>
int a[100001];
int main()
{
    int l,r,i,j,k=0;
    scanf("%d%d",&l,&r);
    for(i = l; i<=r; i++)
       a[i] = 1;
    k=(int)sqrt(r);
    for(i = 2; i<=k; i++)
    {
       a[i*i]+=i;
       for(j=i+1; j*i<=r; j++)
          a[j*i] += i + j;
    }
    k=0;
    for(i=l; i<=r; i++)
       if(a[i] > i) k++;
    printf("%d",k);
    //system("pause");
    return 0;
}
```

Cải tiến tiếp đi một chút với nhận xét chỉ cần xét cho các số bắt đầu từ L. Đoạn mã như sau (time=0.03s = time của test lớn nhất [L,R] = [1,100000]):

```
#include<stdio.h>
#include<math.h>
int a[100001];
int main()
{
    int l,r,i,j,k=0,x;
    scanf("%d%d",&l,&r);
    for(i = l; i<=r; i++)
       a[i] = 1;
    k=(int)sqrt(r);
    for(i = 2; i<=k; i++)
    {
       x=l/i;
       j=(x>i && x<=k)?x:i;
       if(i==j)
       {
          a[i*i] += i;
          j++;
       }
```

```
            for(; j*i<=r; j++)
               a[j*i] += i + j;
        }
        k=0;
        for(i=l; i<=r; i++)
            if(a[i] > i) k++;
        printf("%d",k);
        //system("pause");
        return 0;
}
```

Bài này time tốt nhất là **0.00s** viết bằng C++ của tài khoản bk20101531.

Trên đây là bài khởi động với tư tưởng phân tích từng bước, tối ưu thuật toán và cũng là tư tưởng sẽ xuyên suốt các bài tập khác trong tập san này.

Tác giả: Vũ Đình Trung

Bài 2. 1783. Tìm số nguyên tố - http://vn.spoj.pl/problems/PNUMBER/

Hãy tìm tất cả các số nguyên tố trong đoạn [A,B] .

Input

Gồm 2 số nguyên A và B cách nhau bởi 1 dấu cách ($1 \leq A \leq B \leq 200000$) .

Output

Ghi ra tất cả các số nguyên tố trong đoạn [A,B]. Mỗi số trên 1 dòng .

Ví dụ

Input:

1 10

Output:

2

3

5

7

Time: 5s

Đây là một bài trong mục http://vn.spoj.pl/problems/acm/, bắt buộc kết quả đưa ra phải trong thời gian cho phép và phải đúng hoàn toàn, sai 1 test coi như là sai cả, khác với trong http://vn.spoj.pl/problems/oi/ đúng test nào ăn điểm test đấy.

Giải

1. Cách giải thông thường:

- Viết 1 hàm kiểm tra số nguyên tố. Có nhiều cách để kiểm tra số a có phải là số nguyên tố không:
  - Thô sơ nhất là kiểm tra nếu a có chia hết cho bất kỳ một số nguyên nào trong đoạn [2,a/2] thì nó không phải là số nguyên tố.
  - Cách thứ hai với nhận xét nếu a là số nguyên tố thì nó sẽ không chia hết cho bất kể số nguyên nào trong đoạn [2, sqrt(a)]
  - Cách thứ ba với nhận xét nếu a là số nguyên tố thì nó sẽ không chia hết cho bất kể số nguyên tố nào trong đoạn [2, sqrt(a)]
  - Các cách trên bạn đọc có thể dễ dàng cài đặt, tuy nhiên vẫn còn một số cách khác nhanh hơn nhưng ta không bàn đến ở đây vì cài đặt nó cũng khác phức tạp.
- Tiếp theo chỉ việc xét các số trong đoạn [A,B], nếu hàm kiểm tra nó đúng là số nguyên tố thì in ra.

Nhận xét: cách giải thông thường thì khả năng quá thời gian là rất cao mặc dù time cho tới 5s với giới hạn của bài này.

2. Dùng sàng nguyên tố

Việc dùng sàng nguyên tố sẽ giúp ta nhanh chóng in ra các số nguyên tố trong một giới hạn mà bộ nhớ cho phép cài đặt.

- Sàng thông thường là ta xét các số i trong đoạn [2,sqrt(B)], nếu i là số nguyên tố thì các bội của i không phải là số nguyên tố tức là i*j với j trong đoạn [i, B/i] không phải là số nguyên tố. Cuối cùng ta chỉ cần duyệt lại mảng đánh dấu và in ra các số nguyên tố. Đoạn mã như sau (time: 0.08s):

```
#include <stdio.h>
#include <math.h>
char a[200000];
void sang(int n)
{
    int i,j,u;
    int x = (int)sqrt(n);
    a[0]=1;
    a[1]=1;
    for(i=2;i<=x;i++)
```

```
        {
            u=n/i+1;
            if(a[i]==0)
            {
                for(j=i;j<u;++j)
                {
                    if(a[j*i]==0)
                        a[j*i]=1;
                }
            }
        }
    }
    int main()
    {
        int i,k,A,B;
        scanf("%d%d",&A,&B);
        sang(B);
        for(i = A; i<=B; ++i)
        {
            if(a[i]==0)
                printf("%d\n",i);
        }
        system("pause");
        return 0;
    }
```

❖ Sàng cải tiến bộ nhớ là sàng mà ta chỉ xét các số lẻ vì số nguyên tố đều là số lẻ ngoại trừ 2, do đó bộ nhớ để đánh dấu giảm đi một nửa. Cài đặt của thuật toán này có phần phức tạp hơn thuật toán trên một chút, với thời gian sàng tương đương nhau nhưng nhanh hơn về thời gian duyệt để in ra. Đoạn mã như sau (time: 0.05s):

```
#include <stdio.h>
#include <math.h>
char a[100000];
void sang(int n)
{
    int i,j,k,u;
    int x = (int)sqrt(n);
    for(i=1;;i++)
    {
        k=i*2+1;
        if(k>x) break;
        u=(n+1)/k;
        if(a[i]==0)
        {
            for(j = k;j<u+1;j+=2)
            {
                if(a[(j*k-1)/2]==0)
                    a[(j*k-1)/2]=1;
            }
        }
```

```
        }
    }
    int main()
    {
        int i,k,A,B;
        scanf("%d%d",&A,&B);
        if(A<=2)
        {
            A = 1;
            printf("2\n");
        }
        else
            A = A/2;
        sang(B);
        B = B/2 + B%2;
        for(i = A; i<B; ++i)
        {
            if(a[i]==0)
                printf("%d\n",i*2+1);
        }
        //system("pause");
        return 0;
    }
```

❖ Ngoài ra còn có các sàng cải tiến khác về bộ nhớ có thể tham khảo thêm tại blog của tác giả: http://my.opera.com/trung8290/blog/sang-nguyen-to.

Tác giả: Vũ Đình Trung

Bài 3. 3632. Số thân thiện - http://vn.spoj.pl/problems/NKNUMFRE/

Số tự nhiên có rất nhiều tính chất thú vị. Ví dụ với số 23, số đảo ngược của nó là 32. Hai số này có ước chung lớn nhất là 1. Những số như thế được gọi là số thân thiện, tức là số 23 được gọi là số thân thiện, số 32 cũng được gọi là số thân thiện.

Hãy nhập vào 2 số nguyên a,b ($10 \le a \le b \le 30000$). Hãy đếm xem trong khoảng từ a đến b (kể cả a và b) có bao nhiêu số thân thiện.

Dữ liệu

Bao gồm một dòng chứa 2 số a,b. Hai số được cách nhau bằng một khoảng trắng

Kết quả

Bao gồm một dòng là kết quả của bài toán.

Ví dụ

Dữ liệu

20 30

Kết quả

3

Time: 1s

Giải

Để giải bài này trước tiên ta phải viết 2 hàm là hàm đảo ngược 1 số nguyên và hàm tìm ước chung lớn nhất.

❖ Hàm đảo ngược số nguyên

- Ta có thể làm theo cách chuyển số nguyên về dạng chuỗi (dùng hàm sprintf() hoặc itoa()) , đảo ngược chuỗi và lại chuyển chuỗi về dạng số nguyên (dùng hàm atoi()). Cách này bạn đọc tự cài đặt.
- Cách thứ hai là chuyển số nguyên về mảng các chữ số nguyên sau đó tính lại. Đoạn mã như sau:

```
int daonguoc(int n)
{
    int i,k=0,hs[5],m=0,cs=1;
    while(n != 0)
    {
       hs[k++] = n%10;
       n = n/10;
    }
    for(i = k-1; i>=0; i--)
    {
       m += hs[i]*cs;
       cs *= 10;
    }
    return m;
}
```

❖ Hàm tìm ước chung lớn nhất

- Cài đặt đệ quy:

```
int ucln(int a, int b)
{
    if (a==0 ||b==0) return a+b;
    if(a==b) return a;
    if(a>b) return ucln(a-b,a);
    return ucln(a, b-a);
}
```

Hoặc:

```
int ucln(int a, int b)
{
    if (a==0 ||b==0) return a+b;
    if(a%b==0) return b;
    return ucln(b, a%b);
}
```

- Cài đặt không đệ quy:

```
int ucln(int a, int b)
{
    if (a==0 ||b==0)return a+b;
    while (a !=b)
    {
       if(a>b)
          a=a-b;
       else
          b=b-a;
    }
    return a;
}
```

❖ Chương trình chính

Theo đề bài nếu i và k=daonguoc(i) có ucln(i,k) =1 thì cả i và k đều gọi là số thân thiện, nhưng ta phải kiểm tra xem k có nằm trong đoạn [a,b] không và k đã xét chưa do đó phải có mảng check[] để dánh dấu. Đoạn mã như sau (time = 0.04s):

```
#include <stdio.h>
char check[30001];

int main()
{
    int i,a,b,k,d=0;
    scanf("%d%d",&a,&b);
    for(i=a; i<=b; ++i)
    {
       if(check[i] == 0)
       {
          k = daonguoc(i);
          if(ucln(i,k)==1)
          {
             d++;
             check[i] = 1;
             if(a<=k && k<=b && check[k] == 0)
             {
                 check[k]=1;
                 d++;
             }
          }
       }
```

```
        }
        printf("%d",d);
        //system("pause");
        return 0;
    }
```

Tác giả: Vũ Đình Trung

Bài 4. 4141. Euler Totient Function - http://vn.spoj.pl/problems/ETF/

Trong số học, hàm Ơ-le của một số nguyên dương n được định nghĩa là số lượng các số nguyên dương nhỏ hơn hoặc bằng n và nguyên tố cùng nhau với n.

Cho số nguyên dương n (1 <= n <= 10^6). Tính giá trị của hàm Ơ-le.

Input

Dòng đầu chứa số nguyên T là số test (T <= 20000)

T dòng tiếp theo, mỗi dòng chứa một số nguyên n.

Output

T dòng, mỗi dòng ghi kết quả của test tương ứng.

Example

Input:

5
1
2
3
4
5

Output:

1
1
2
2
4

Time: 1s

Giải

Hàm Ơle có công thức như sau: $f(n) = n.(1 - 1/p_1).(1 - 1/p_2). .. .(1 - 1/p_k)$

Với $p_1, p_2,.., p_k$ là các ước nguyên tố của n.

Như vậy trước tiên ta phải viết hàm tìm tất cả các ước nguyên tố của n. Trong hàm này ta sẽ duyệt từ các số i từ 2 cho đến căn bậc hai của n, nếu n chia hết cho i thì lưu i vào mảng các ước của n và lặp n=n/i trong khi n vẫn chia hết cho i. Khi duyệt xong nếu n>1 thì n cũng chính là một ước của số ban đầu. Bạn đọc có thể dễ dàng cài đặt hàm này.

Sau khi cài đặt xong hàm tìm tất cả các ước nguyên tố: factor(int n), công việc còn lại rất đơn giản chỉ là duyệt và đếm. Hàm chính như sau (time = 0.79s):

```
int main()
{
    int t,n;
    int i,j;
    int tam;
    scanf("%d",&t);
    for(i=0; i<t; ++i)
    {
        scanf("%d",&n);
        factor(n);
        tam=1;
        for(j=0; j<k; ++j)
        {
            tam*=a[j]-1;
            n/=a[j];
        }
        printf("%d\n",tam*n);
    }
    return 0;
}
```

Tác giả: Vũ Đình Trung

PHẦN II. XỬ LÝ CHUỖI

Bài 1. 4257. First Number - http://vn.spoj.pl/problems/MDIGITS2/

Viết các số thập phân 1,2, ... liên tiếp thu được dãy số như sau :

12345678910111213141516171819202122 ...

etc. Viết chương trình tìm vị trí xuất hiện đầu tiên của số N trong dãy trên.

Input

Gồm duy nhất 1 số N, $1 \leq N \leq 100{,}000$.

Output

Số duy nhất là vị trí xuất hiện đầu tiên của số N trong dãy.

Sample

Input	15	34	142
Output	20	3	73

Time: 1s

Giải

Bài này đơn giản chỉ là ghép chuỗi số và tìm vị trí xuất hiện của chuỗi con. Mục tiêu để các bạn luyện tập dùng thư viện string STL của C++. Chương trình như sau (time: 0.57s):

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    int n;
    string s="";
    char t[7];
    cin>>n;
    for(int i=1; i<=n; ++i)
    {
       sprintf(t,"%d",i);
       s+=t;
    }
    sprintf(t,"%d",n);
    cout<<s.find(t)+1;
    //system("pause");
    return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 2. 3638. Word Counting - http://vn.spoj.pl/problems/WORDCNT/

Nguyên đang viết một phần mềm đếm từ trong một xâu ký tự. Cậu cảm thấy buồn chán sau khi viết xong phần mềm rất nhanh. Bây giờ, cậu muốn tìm P là số lượng lớn nhất các từ có độ dài bằng nhau đứng liên tiếp trong xâu cho trước.

Cho một xâu chỉ chứa các ký tự từ a đến z và ký tự trống. Mỗi từ là một chuỗi các ký tự liên tiếp khác ký tự trống và các từ phân tách nhau bởi ít nhất một ký tự trống. Nhiệm vụ của bạn là viết chương trình giúp Nguyên tìm số P nói trên.

Dữ liệu vào

Dữ liệu vào gồm nhiều bộ dữ liệu tương ứng với nhiều test. Dòng đầu tiên chứa một số nguyên dương không lớn hơn 20 là số lượng các bộ dữ liệu. Các dòng tiếp theo chứa các bộ dữ liệu.

Trên mỗi dòng tiếp theo chứa xâu ký tự có không quá 1000 từ tương ứng với mỗi bộ dữ liệu, mỗi từ có không quá 20 ký tự.

Dữ liệu ra

Với mỗi bộ dữ liệu, ghi ra trên một dòng số P mà Nguyên muốn tìm.

Ví dụ

Dữ liệu vào

2
a aa bb cc def ghi
a a a a a bb bb bb bb c c

Dữ liệu ra

3
5

Time: 1s

Giải

Với bài này ta phải đọc cả dòng (chuỗi có dấu cách) sử dụng hàm cin.getline() của C++ hoặc hàm gets() của C.

Sau đó ta sẽ duyệt và đếm số ký tự của từng từ và lưu vào mảng đếm.

Cuối cùng chỉ việc duyệt từ đầu đến cuối mảng đếm để tìm đoạn con dài nhất có các phần tử liên tiếp bằng nhau.

Đoạn chương trình như sau (time: 0.00s ☺):

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char *s = new char[20002];
    char *c = new char[20002];
    short n;
    cin>>n;
    //bat buoc phai them doan cin.getline(s,20001); sau
     vao de loai bo dong dau tien
```

```
        cin.getline(s,20001);
        for(int i=0; i<n; ++i)
        {
           memset(c,0,20001);
           cin.getline(s,20001);
           short k = 0, max = 0,v=0;
           s[strlen(s)]=0;
           for(int j=0; j<=strlen(s); ++j)
           {
              if(s[j] <='z' && s[j]>= 'a')
                 ++k;
              else
              {
                 if(k!=0)
                 {
                    c[v++] = k;
                    k=0;
                 }
              }
           }
           k = 1;
           c[v++] = 0;
           for(int j=1; j<v; ++j)
           {
              if(c[j] == c[j-1])
                 ++k;
              else
              {
                 if(max < k) max = k;
                 k=1;
              }
           }
           cout<<max<<endl;
        }
        return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 3. Tập số - Chuyên tin 2011

Cho số **n** ở hệ cơ số 10, có không quá 20 chữ số và không chứa các số **0** không có nghĩa ở đầu.

Bằng cách xóa một hoặc một vài chữ số liên tiếp của **n** (nhưng không xóa hết tất cả các chữ số của **n**) ta nhận được những số mới. Số mới được chuẩn hóa bằng cách xóa các chữ số **0** vô nghĩa nếu có.

Tập số nguyên D được xây dựng bằng cách đưa vào nó số **n**, các số mới khác nhau đã chuẩn hóa và khác **n**. Ví dụ, với n = 1005 ta có thể nhận được các số mới như sau:

- Bằng cách xóa một chữ số ta có các số: 5 (từ 005), 105, 105, 100;
- Bằng cách xóa hai chữ số ta có các số: 5 (từ 05), 15, 10;
- Bằng cách xóa 3 chữ số ta có các số: 5 và 1.

Tập D nhận được từ n chứa các số {1005, 105, 100, 15, 10, 5, 1}. Trong tập D này có 3 số chia hết cho 3, đó là các số 1005, 105 và 15.

Yêu cầu: Cho số nguyên **n**. Hãy xác định số lượng số chia hết cho 3 có mặt trong tập D được tạo thành từ **n**.

Dữ liệu: Vào từ file văn bản NUMSET.INP gồm một dòng chứa số nguyên **n**.

Kết quả: Đưa ra file văn bản NUMSET.OUT một số nguyên – số lượng số chia hết cho 3 tìm được.

Ví dụ:

NUMSET.INP	NUMSET.OUT
1005	3

***Time** : 2s*

Giải

Giới hạn bài này khá nhỏ, có thể giải quyết đơn giản bằng cách sử dụng thư viện STL của C++.

Ta đọc số n như là đọc một chuỗi số nguyên (string). Việc cắt bỏ đi các chữ số thì ta sẽ sử dụng hàm cắt chuỗi con substr() của đối tượng string.

Tập D chứa các số đôi một khác nhau, do đó ta sẽ sử dụng cấu trúc set của C++ (cấu trúc này được tối ưu để lưu trữ các giá trị khác nhau và được sắp xếp mặc định theo thứ tự tăng dần).

Cuối cùng ta chỉ việc kiểm tra tổng các chữ số của từng chuỗi số nguyên trong set nếu có chia hết cho 3 thì tăng biến đếm lên. Cài đặt như sau (time : 0.01- 0.02s) :

```
#include<iostream>
#include<set>
#include<cstring>
using namespace std;
int main()
{
    //freopen("NUMSET.INP","rt",stdin);
```

```
    string n,s;
    set<string> st;
    cin>>n;
    int len = n.length();
    st.insert(n);
    for(int i=1; i<len; ++i)
    {
       for(int j=0; j<= len-i; ++j)
       {
          s=n.substr(0,j) + n.substr(j+i);
          while(s[0] == '0') s=s.substr(1);
          st.insert(s);
       }
    }
    set<string>::iterator it;
    int tong;
    int dem=0;
    for(it=st.begin(); it!=st.end(); ++it)
    {
       tong=0;
       for(int i=0; i<(*it).length(); ++i)
          tong+=(*it)[i]-'0';
       if(tong % 3 == 0) dem++;
    }
    //freopen("NUMSET.OUT","wt",stdout);
    cout<<dem;
    //system("pause");
    return 0;
}
```

Tác giả: Vũ Đình Trung

PHẦN III. ĐỒ THỊ

Bài 1. 2722. Gặm cỏ - http://vn.spoj.pl/problems/VMUNCH/

Bessie rất yêu bãi cỏ của mình và thích thú chạy về chuồng bò vào giờ vắt sữa buổi tối.

Bessie đã chia đồng cỏ của mình là 1 vùng hình chữ nhật thành các ô vuông nhỏ với R (1 <= R <= 100) hàng và C (1 <= C <= 100) cột, đồng thời đánh dấu chỗ nào là cỏ và chỗ nào là đá. Bessie đứng ở vị trí R_b,C_b và muốn ăn cỏ theo cách của mình, từng ô vuông một và trở về chuồng ở ô 1,1 ; bên cạnh đó đường đi này phải là ngắn nhất.

Bessie có thể đi từ 1 ô vuông sang 4 ô vuông khác kề cạnh.

Dưới đây là một bản đồ ví dụ [với đá ('*'), cỏ ('.'), chuồng bò ('B'), và Bessie ('C') ở hàng 5, cột 6] và một bản đồ cho biết hành trình tối ưu của Bessie, đường đi được dánh dấu bằng chữ 'm'.

```
      Bản đồ                    Đường đi tối ưu
    1 2 3 4 5 6  <-cột          1 2 3 4 5 6  <-cột
1   B . . . * .             1   B m m m * .
2   . . * . . .             2   . . * m m m
3   . * * . * .             3   . * * . * m
4   . . * * * .             4   . . * * * m
5   * . . * . C             5   * . . * . m
```

Bessie ăn được 9 ô cỏ.

Cho bản đồ, hãy tính xem có bao nhiêu ô cỏ mà Bessie sẽ ăn được trên con đường ngắn nhất trở về chuồng (tất nhiên trong chuồng không có cỏ đâu nên đừng có tính nhé)

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: R và C
- Dòng 2..R+1: Dòng i+1 mô tả dòng i với C ký tự (và không có dấu cách) như đã nói ở trên.

Kết quả

- Dòng 1: Một số nguyên là số ô cỏ mà Bessie ăn được trên hành trình ngắn nhất trở về chuồng.

Ví dụ

Dữ liệu

5 6

B...*.

..*...

.**.*.

..***.

...C

Kết quả

9

Time: 1s

Giải

Đây là một bài duyệt theo chiều rộng BFS thông thường để nhanh chóng tìm ra đường đi tối ưu. Sau đây ta sẽ cài đặt thuật toán này bằng C++ sử dụng hàng đợi queue của STL và xây dựng theo kiểu class. Bạn đọc cũng có thể tự cài đặt bằng C sử dụng mảng và struct.

Từ ô bắt đầu Bessie đứng ta sẽ loang ra 4 bên cạnh là trên, dưới, trái, phải nếu ô đó không có đá, rồi lại loang tiếp, cứ như vậy cho đến khi về đến ô chuồng ta sẽ lấy khoảng cách nhỏ nhất.

Cài đặt như sau (time: 0.05s):

```
#include<iostream>
#include<queue>
#define maxN 100
using namespace std;
class node
{
    public:
       int x;
       int y;
       int d;
       node(int x1, int y1, int d1):x(x1),y(y1),d(d1){}
};
bool a[maxN][maxN];
int main()
{
    int m,n;
    int xb,yb,xc,yc;
    int dx[]={1,-1,0,0};
    int dy[]={0,0,1,-1};
    char c;
    scanf("%d%d",&m,&n);
    for(int i=0; i<m; ++i)
    {
       for(int j=0; j<n; ++j)
       {
          cin>>c;
          if(c=='B')
          {
             xb=j;
             yb=i;
             a[i][j]=true;
          }
          else if(c=='C')
          {
             xc=j;
             yc=i;
```

```
                a[i][j]=false;
            }
            else if(c=='.')
                a[i][j]=true;
            else
                a[i][j]=false;
        }
    }
    queue<node> q;
    node u(xc,yc,1);
    q.push(u);
    int x1,y1;
    int min=10000;
    while(!q.empty())
    {
        u=q.front();
        q.pop();
        if(u.x==xb && u.y==yb)
        {
            if(u.d<min) min=u.d;
        }
        else
        {
            for(int i=0; i<4; ++i)
            {
                x1 = u.x + dx[i];
                y1 = u.y + dy[i];
                if(x1>=0 && x1<n && y1>=0 && y1<m && a[y1]
[x1])
                {
                    a[y1][x1]=false;
                    q.push(node(x1,y1,u.d+1));
                }
            }
        }
    }
    printf("%d",min-1);
    return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 2. 2719. Bãi cỏ ngon nhất - http://vn.spoj.pl/problems/VBGRASS/

Bessie dự định cả ngày sẽ nhai cỏ xuân và ngắm nhìn cảnh xuân trên cánh đồng của nông dân John, cánh đồng này được chia thành các ô vuông nhỏ với R (1 <= R <= 100) hàng và C (1 <= C <= 100) cột. Bessie ước gì có thể đếm được số khóm cỏ trên cánh đồng.

Mỗi khóm cỏ trên bản đồ được đánh dấu bằng một ký tự '#' hoặc là 2 ký tự '#' nằm kề nhau (trên đường chéo thì không phải). Cho bản đồ của cánh đồng, hãy nói cho Bessie biết có bao nhiêu khóm cỏ trên cánh đồng.

Ví dụ như cánh đồng dưới dây với R=5 và C=6:

.#....
..#...
..#..#
...##.
.#....

Cánh đồng này có 5 khóm cỏ: một khóm ở hàng đầu tiên, một khóm tạo bởi hàng thứ 2 và thứ 3 ở cột thứ 2, một khóm là 1 ký tự nằm riêng rẽ ở hàng 3, một khóm tạo bởi cột thứ 4 và thứ 5 ở hàng 4, và một khóm cuối cùng ở hàng 5.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: R và C
- Dòng 2..R+1: Dòng i+1 mô tả hàng i của cánh đồng với C ký tự, các ký tự là '#' hoặc '.' .

Kết quả

- Dòng 1: Một số nguyên cho biết số lượng khóm cỏ trên cánh đồng.

Ví dụ

Dữ liệu

5 6
.#....
..#...
..#..#
...##.
.#....

Kết quả

5

Time: 1s

Giải

Bài này ta có thể duyệt theo chiều sâu DFS hoặc chiều rộng BFS. Ta sẽ duyệt để đếm số thành phần liên thông của đồ thị cũng chính là số lượng khóm cỏ trên cánh đồng.

Thuật toán tương tự bài Gặm cỏ. Cài đặt như sau (time: 0.06s):

```
#include<iostream>
#include<queue>
#define maxN 101
using namespace std;
char a[maxN][maxN];
int main()
{
    int r,c;
    int dx[]={0,0,1,-1};
    int dy[]={1,-1,0,0};
    scanf("%d%d",&r,&c);
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
          cin>>a[i][j];
       }
    }
    queue< pair<int,int> > q;
    pair<int,int> p;
    int kq=0;
    int x,y;
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
          if(a[i][j]=='#')
          {
             a[i][j]='.';
             kq++;
             q.push(make_pair(i,j));
             while(!q.empty())
             {
                p=q.front();
                q.pop();
                for(int k=0; k<4; ++k)
                {
                     x=p.first+dx[k];
                     y=p.second+dy[k];
                     if(a[x][y]=='#')
                     {
                        a[x][y]='.';
                        q.push(make_pair(x,y));
                     }
                }
             }
          }
       }
    }
    cout<<kq;
    return 0;
}
```

Bài 3. 2721. Nước lạnh - http://vn.spoj.pl/problems/VCOLDWAT/

Mùa hè oi ả ở Wisconsin đã khiến cho lũ bò phải đi tìm nước để làm dịu đi cơn khát. Các đường ống dẫn nước của nông dân John đã dẫn nước lạnh vào 1 tập N (3 <= N <= 99999; N lẻ) nhánh (đánh số từ 1..N) từ một cái bơm đặt ở chuồng bò.

Khi nước lạnh chảy qua các ống, sức nóng mùa hè sẽ làm nước ấm lên. Bessie muốn tìm chỗ có nước lạnh nhất để cô bò có thể tận hưởng mùa hè một cách thoải mái nhất.

Bessie đã vẽ sơ đồ toàn bộ các nhánh ống nước và nhận ra rằng nó là một đồ thị dạng cây với gốc là chuồng bò và ở các điểm nút ống thì có chính xác 2 nhánh con đi ra từ nút đó. Một điều ngạc nhiên là các nhánh ống này đều có độ dài là 1.

Cho bản đồ các ống nước, hãy cho biết khoảng cách từ chuồng bò tới tất cả các nút ống và ở các phần cuối đường ống.

"Phần cuối" của một đường ống, có thể là đi vào một nút ống hoặc là bị bịt, được gọi theo số thứ tự của đường ống. Bản đồ có C (1 <= C <= N) nút ống, được mô tả bằng 3 số nguyên: là "phần cuối" của ống E_i (1 <= E_i <= N) và 2 ống nhánh đi ra từ đó là B1_i và B2_i (2 <= B1_i <= N; 2 <= B2_i <= N). Đường ống số 1 nối với chuồng bò; khoảng cách từ phần cuối của đường ống này tới chuồng bò là 1.

Dữ liệu

- Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và C
- Dòng 2..C+1: Dòng i+1 mô tả nút ống i với ba Số nguyên cách nhau bởi dấu cách: E_i, B1_i, và B2_i

Kết quả

- Dòng 1..N: Dòng i chứa 1 số nguyên là khoảng cách từ chuồng tới "phần cuối" của ống thứ i.

Ví dụ

Dữ liệu

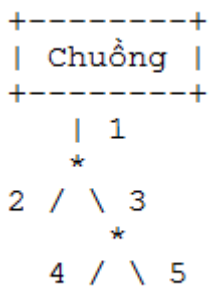
5 2

3 5 4

1 2 3

Giải thích:

Dữ liệu ở trên mô tả bản đồ ống nước sau:



Kết quả

1
2
2
3
3

Giải thích:

Ống 1 luôn cách chuồng 1 đoạn là 1. Ống 2 và 3 nối với ống 1 nên khoảng cách sẽ là 2. Ống 4 và 5 nối với ống 3 nên khoảng cách sẽ là 3.

Time: 1s

Giải

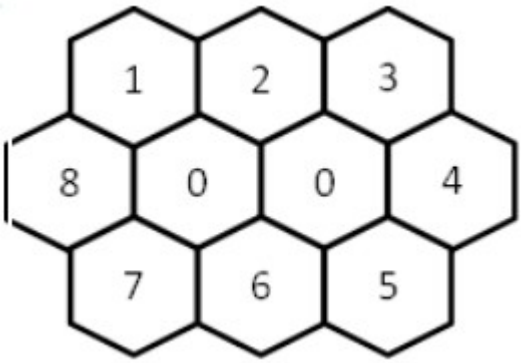
Đây là bài toán duyệt cây nhị phân và tính bậc của nút khá đơn giản. Trước tiên ta sẽ xây dựng cây, sau đó duyệt cây từ gốc bậc bằng 1 và cứ thế lan xuống nút con trái và nút con phải đồng thời tăng bậc lên. Cài đặt như sau (time: 0.44s):

```
#include <stdio.h>
#define MaxN 100000
typedef struct node
{
    int k;
    int l;
    int r;
}node;
node T[MaxN];
void visit(int u, int k)
{
    T[u].k = k;
    if(T[u].l !=0 && T[u].r != 0)
    {
      visit(T[u].l, k+1);
      visit(T[u].r, k+1);
    }
}
```

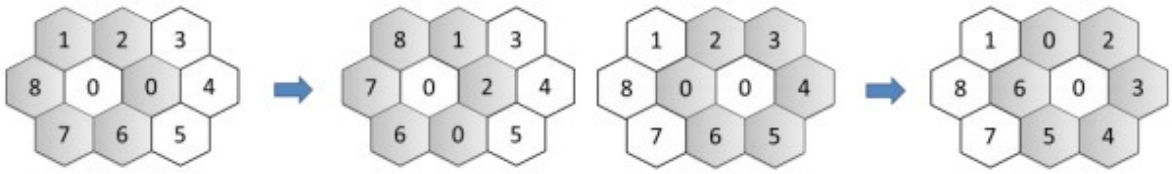
```
int main()
{
    int i,n,c;
    int u,l,r;
    scanf("%d%d",&n,&c);
    for(i=0; i<c; ++i)
    {
      scanf("%d%d%d",&u,&l,&r);
      T[u].l = l;
      T[u].r = r;
    }
    visit(1,1);
    for(i=1; i<=n; ++i)
      printf("%d\n",T[i].k);
    return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 4. Hexgame - Chuyên tin 2011



HEXGAME là một trò chơi xếp hình gồm 10 miếng ghép hình lục giác đều, trên mỗi miếng ghép được điền một số nguyên, có 8 miếng được điền số từ 1 đến 8 và có hai miếng điền số 0. Các miếng liên kết với nhau tạo thành lưới tổ ong. Ban đầu các miếng ghép ở vị trí như hình bên. Tại mỗi bước, chọn một miếng ghép có đúng 6 miếng ghép kề cạnh làm tâm, rồi xoay một nấc 6 miếng ghép kề cạnh đó theo chiều kim đồng hồ. Như vậy chỉ có hai cách chọn tâm. Ví dụ với trạng thái ban đầu nêu trên thì nhận được một trong hai trạng thái dưới đây ứng với cách chọn sau khi xoay một nấc.



Yêu cầu: Cho một trạng thái của trò chơi (nhận được sau một dãy biến đổi từ trạng thái ban đầu), hãy tính số phép biến đổi ít nhất để đưa về trạng thái ban đầu.

Dữ liệu: Vào từ file văn bản HEXGAME.INP có dạng:

- Dòng 1: chứa 3 số ghi trên 3 miếng ghép ở dòng thứ nhất của lưới theo thứ tự từ trái qua phải;
- Dòng 2: chứa 4 số ghi trên 4 miếng ghép ở dòng thứ hai của lưới theo thứ tự từ trái qua phải;
- Dòng 3: chứa 3 số ghi trên 3 miếng ghép ở dòng thứ ba của lưới theo thứ tự từ trái qua phải.

Kết quả: Đưa ra file văn bản HEXGAME.OUT gồm một dòng ghi một số là số phép biến đổi ít nhất.

HEXGAME.INP	HEXGAME.OUT
1 0 2 8 6 0 3 7 5 4	5

Ghi chú: có 50% số test có số phép biến đổi không vượt quá 15.

Time: 2s

Giải

Đây là một bài duyệt đồ thị trạng thái, ta sẽ duyệt qua các nút trạng thái của game cho đến thi gặp nút trạng thái ban đầu. Đề bài yêu cầu tìm số phép biến đổi ít nhất nên ta sẽ sử dụng thuật toán tìm kiếm theo chiều rộng BFS.

Theo đề bài từ 1 trạng thái ta có thể biến đổi được thành 2 trạng thái mới, ta có bảng tham chiếu như sau:

Trạng thái 1										
Chỉ số cũ	0	1	2	3	4	5	6	7	8	9
	1	2	3	8	0	0	4	7	6	5
	8	1	3	7	0	2	4	6	0	5
Chỉ số mới	3	0	2	7	4	1	6	8	5	9
Trạng thái 2										
Chỉ số cũ	0	1	2	3	4	5	6	7	8	9

	1	2	3	8	0	0	4	7	6	5
	1	0	2	8	6	0	3	7	5	4
Chỉ số mới	0	4	1	3	8	5	2	7	9	6

Chương trình sau viết bằng ngôn ngữ C++ với class và queue, bạn đọc có thể tự cài đặt lại bằng C với struct và array để so sánh (time: 0.02s out: 5; 0.33s out: 21):

```
#include<iostream>
#include<queue>
using namespace std;
class node
{
    public:
       int a[10];
       node(){}
       void input()
       {
          //freopen("HEXGAME.INP","rt",stdin);
          for(int i=0; i<10; ++i)
             cin>>a[i];
       }

       //Trang thai 1 lay tam ben trai de quay
       node left()
       {
          node b;
          b.a[0] = a[3];
          b.a[1] = a[0];
          b.a[2] = a[2];
          b.a[3] = a[7];
          b.a[4] = a[4];
          b.a[5] = a[1];
          b.a[6] = a[6];
          b.a[7] = a[8];
          b.a[8] = a[5];
          b.a[9] = a[9];
          return b;
       }

       //Trang thai 2 lay tam ben phải de quay
       node right()
       {
          node b;

          b.a[0] = a[0];
          b.a[1] = a[4];
          b.a[2] = a[1];
          b.a[3] = a[3];
          b.a[4] = a[8];
          b.a[5] = a[5];
          b.a[6] = a[2];
```

```
            b.a[7] = a[7];
            b.a[8] = a[9];
            b.a[9] = a[6];
            return b;
        }
        //Kiem tra trang thai ban dau
        bool isroot()
        {
            if(a[0]!=1 || a[1]!=2 || a[2]!=3 || a[3]!=8 ||
                 a[4]!=0 ||a[5]!=0 || a[6]!=4 || a[7]!=7 ||
                 a[8]!=6 || a[9]!=5)
               return false;
            return true;
        }
};
int main()
{
    queue< pair<node, int> > q;
    node b;
    b.input();
    q.push(make_pair(b,0));
    pair<node, int> u;
    while(true)
    {
        u = q.front();
        q.pop();
        b=u.first;
        if(b.isroot())
        {
            //freopen("HEXGAME","wt",stdout);
            cout<<u.second;
            break;
        }
        else
        {
            //Dua 2 trang thai moi vao hang doi
            q.push(make_pair(b.left(),u.second+1));
            q.push(make_pair(b.right(),u.second+1));
        }
    }
    //system("pause");
    return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 5. 3478. Xây dựng thành phố - http://vn.spoj.pl/problems/NKCITY/

Nước Anpha đang lập kế hoạch xây dựng một thành phố mới và hiện đại. Theo kế hoạch, thành phố sẽ có N vị trí quan trọng, được gọi là N trọng điểm và các trọng điểm này được đánh số từ 1 tới N. Bộ giao thông đã lập ra một danh sách M tuyến đường

hai chiều có thể xây dựng được giữa hai trọng điểm nào đó. Mỗi tuyến đường có một thời gian hoàn thành khác nhau.

Các tuyến đường phải được xây dựng sao cho N trọng điểm liên thông với nhau. Nói cách khác, giữa hai trọng điểm bất kỳ cần phải di chuyển được đến nhau qua một số tuyến đường. Bộ giao thông sẽ chọn ra một số tuyến đường từ trong danh sách ban đầu để đưa vào xây dựng sao cho điều kiện này được thỏa mãn.

Do nhận được đầu tư rất lớn từ chính phủ, bộ giao thông sẽ thuê hẳn một đội thi công riêng cho mỗi tuyến đường cần xây dựng. Do đó, thời gian để hoàn thành toàn bộ các tuyến đường cần xây dựng sẽ bằng thời gian lâu nhất hoàn thành một tuyến đường nào đó.

Yêu cầu: Giúp bộ giao thông tính thời gian hoàn thành các tuyến đường sớm nhất thỏa mãn yêu cầu đã nêu.

Dữ liệu

Dòng chứa số N và M ($1 \leq N \leq 1000$; $1 \leq M \leq 10000$).

M tiếp theo, mỗi dòng chứa ba số nguyên u, v và t cho biết có thể xây dựng tuyến đường nối giữa trọng điểm u và trọng điểm v trong thời gian t. Không có hai tuyến đường nào nối cùng một cặp trọng điểm.

Kết quả

Một số nguyên duy nhất là thời gian sớm nhất hoàn thành các tuyến đường thỏa mãn yêu cầu đã nêu.

Ví dụ

Dữ liệu

5 7
1 2 2
1 5 1
2 5 1
1 4 3
1 3 2
5 3 2
3 4 4

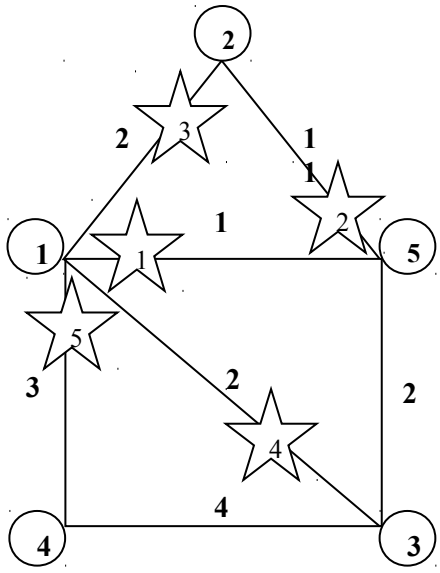
Kết quả

3

Time: 1s

Giải

Đề bài khá dài nhưng tóm lại mục tiêu của ta là phải tìm ra 1 phương án xây dựng hoàn thành trong thời gian ngắn nhất đi qua tất cả các trọng điểm, nghĩa là thời gian lâu nhất để hoàn thành 1 tuyến đường nào đó trong phương án phải là nhỏ nhất trong tất cả các phương án xây dựng có thể. Nói cách khác chính là tìm cây khung nhỏ nhất của đồ thị, sau đó đưa ra trọng số lớn nhất trên cây khung đó. Ta sẽ áp dụng giải thuật **PRIM** như sau.



- Đầu tiên xét các đỉnh kề đỉnh 1 cho vào hàng đợi, chọn tiếp đỉnh 5 vì trọng số cạnh 1->5 là nhỏ nhất trong số các cạnh xuất phát từ đỉnh 1.
- Tiếp tục xét các đỉnh kề 5 cho tiếp vào hàng đợi, chọn tiếp đỉnh 2 với lý do như trên. Cứ thế tiếp tục cho đến khi chọn đủ các đỉnh.
- Trong số các cạnh đã chọn thì cạnh 1->4 có trọng số lớn nhất bằng 3 là kết quả cần tìm của bài toán.

Ta sẽ cài đặt bài này sử dụng hàng đợi ưu tiên priority_queue trong thư viện STL của C++. Chú ý là với hàng đợi ưu tiên thì mặc định nó sẽ sắp xếp giảm dần theo tham số đầu tiên, ví dụ **priority_queue< pair<int, int> > q;** sẽ sắp xếp theo tham số kiểu int đầu tiên. Nếu ta muốn lấy ra giá trị nhỏ nhất ở đầu hàng đợi thì chỉ việc đổi dấu giá trị truyền vào hàng đợi.

Cài đặt của bài này như sau đúng theo tư tưởng thuật toán đã trình bày ở trên (time: 0.60s):

```
#include <iostream>
#include <vector>
#include <queue>
#define MIN -2147483647
using namespace std;
int main()
{
    int n,m;
    int u, v, t;
    cin>>n>>m;
    //doan khai bao sau tuong duong voi mang 2 chieu
    vector< vector< pair<int, int> > > dinh(n);
    //mang danh dau cac dinh da xet
    vector< bool > daxet(n);
    pair<int, int> ts;
    for(int i=0; i<m; ++i)
    {
       cin>>u>>v>>t;
       ts = make_pair(v-1,t);
       dinh[u-1].push_back(ts);
       ts = make_pair(u-1,t);
       dinh[v-1].push_back(ts);
    }
    //khoi tao bien dem so dinh da duyet u
    u = 0;
    //khoi tao bien t luu trong so lon nhat tren tuyen
duong di nho nhat
    t = -MIN;
    //khoi tao hang doi- duyet tu dinh 0
    ts = make_pair(MIN,0);
    priority_queue< pair<int, int> > q;
    q.push(ts);
    while(!q.empty())
    {
       ts = q.top();
       q.pop();
       v = ts.second;
       if(!daxet[v])
       {
          //tang so dinh da duoc chon len
          ++u;
          //luu lai gia trong so max
          if(ts.first < t && ts.first!=MIN)
             t = ts.first;
          //da duyet xong thoat khoi vong lap
          if(u == n) break;
          //danh dau dinh da xet
          daxet[v] = true;
          //dua cac dinh ke voi dinh dang xet vao hang
doi
          for(int i=0; i< dinh[v].size(); ++i)
             if(!daxet[dinh[v][i].first])
```

```
                    q.push(make_pair(-dinh[v][i].second,
dinh[v][i].first));
            }
        }
        cout<<-t;
        return 0;
}
```

Chú ý:

- Để giảm thời gian các bạn hãy thử chuyển toàn bộ cin, cout thành scanf, và printf (time: 0.22s).
- Một cách khai báo hàng đợi khác để mặc định sắp xếp theo chiều tăng dần đó là: ***priority_queue< int, vector< pair<int, int> >, greater< pair<int, int> > >*** (mặc định là ***less< pair<int, int> >***)

Tác giả: Vũ Đình Trung

Bài 6. 3125. Đến trường - http://vn.spoj.pl/problems/QBSCHOOL/

Ngày 27/11 tới là ngày tổ chức thi học kỳ I ở trường ĐH BK. Là sinh viên năm thứ nhất, Hiếu không muốn vì đi muộn mà gặp trục trặc ở phòng thi nên đã chuẩn bị khá kỹ càng. Chỉ còn lại một công việc khá gay go là Hiếu không biết đi đường nào tới trường là nhanh nhất.

Thường ngày Hiếu không quan tâm tới vấn đề này lắm cho nên bây giờ Hiếu không biết phải làm sao cả . Bản đồ thành phố là gồm có N nút giao thông và M con đường nối các nút giao thông này. Có 2 loại con đường là đường 1 chiều và đường 2 chiều. Độ dài của mỗi con đường là một số nguyên dương.

Nhà Hiếu ở nút giao thông 1 còn trường ĐH BK ở nút giao thông N. Vì một lộ trình đường đi từ nhà Hiếu tới trường có thể gặp nhiều yếu tố khác như là gặp nhiều đèn đỏ , đi qua công trường xây dựng, ... phải giảm tốc độ cho nên Hiếu muốn biết là có tất cả bao nhiêu lộ trình ngắn nhất đi từ nhà tới trường. Bạn hãy lập trình giúp Hiếu giải quyết bài toán khó này.

Input

Dòng thứ nhất ghi hai số nguyên N và M.

M dòng tiếp theo, mỗi dòng ghi 4 số nguyên dương K, U, V, L. Trong đó:

K = 1 có nghĩa là có đường đi một chiều từ U đến V với độ dài L.

K = 2 có nghìa là có đường đi hai chiều giữa U và V với độ dài L.

Output

Ghi hai số là độ dài đường đi ngắn nhấn và số lượng đường đi ngắn nhất. Biết rằng số lượng đường đi ngắn nhất không vượt quá phạm vì int64 trong pascal hay long long trong C++.

Example

Input:

3 2

1 1 2 3

2 2 3 1

Output:

4 1

Giới hạn:

$1 \leq N \leq 5000$

$1 \leq M \leq 20000$

Độ dài các con đường ≤ 32000

Time: 1s

Giải

Đây là bài thuộc dạng tìm đường đi ngắn nhất trên đồ thị có trọng số không âm nên ta sẽ sử dụng thuật toán Dijkstra, tuy nhiên phải cải tiến một chút để có thể đưa ra cả số lượng đường đi ngắn nhất tìm được như sau:

Ta sẽ gán nhãn tạm thời cho các đỉnh - nhãn của mỗi đỉnh cho biết độ dài đường đi ngắn nhất hiện thời tới đỉnh đó và sử dụng một mảng để đếm số lượng đường đi ngắn nhất tới mỗi đỉnh.

Tại mỗi lượt ta sẽ lấy ra đỉnh có nhãn nhỏ nhất trong hàng đợi và tiếp tục lấy cho đến khi được đỉnh chưa xét thì gán đỉnh đó thành đã xét. Giả sử đỉnh lấy ra là đỉnh u chưa xét thì lúc này gán u thành đã xét và duyệt qua các đỉnh kề với đỉnh u, giả sử v kề u cập nhật lại nhãn cho đỉnh này như sau

- Nếu nhãn của v > nhãn của u + trọng số cạnh (u,v) thì gán lại nhãn của v = nhãn của u + trọng số cạnh (u,v) và số đường đi ngắn nhất đến v = số đường đi ngắn nhất đến v. Đưa đỉnh v vào hàng đợi.
- Nếu nhãn của v = nhãn của u + trọng số cạnh (u,v) thì gán lại nhãn của v = nhãn của v + nhãn của u.

Cứ như vậy cho đến khi duyệt hết đỉnh cuối cùng thì nhãn của đỉnh cuối cùng - chính là nút giao thông thứ N. Kết quả của bài toán chính là nhãn tại đỉnh N và đếm tại đỉnh N. Cài đặt sử dụng priority_queue như sau (time: 0.56s):

```
#include <iostream>
#include <vector>
#include <queue>
#include <list>
#define MaxN 5002
#define MAX 2147483647
using namespace std;
bool daxet[MaxN];
long long dem[MaxN];
int dist[MaxN];
list< pair<int,int> > dinh[MaxN];
pair<int,int> ts;
int main()
{
    int n,m;
    int k, u, v, d;
    scanf("%d%d",&n,&m);
    for(int i=0; i<m; ++i)
    {
        scanf("%d%d%d%d",&k,&u,&v,&d);
        dinh[u].push_back(make_pair(d,v));
        if(k==2)
            dinh[v].push_back(make_pair(d,u));
    }
    for(int i=1;i<=n;++i)
        dist[i] = MAX;
    list<pair<int,int> >::iterator it;
    priority_queue< pair<int,int>, vector< pair<int,int>
>, greater<pair<int,int> > > q;
    q.push(make_pair(0,1));
    dist[1] = 0;
    dem[1] = 1;
    while(!q.empty())
    {
        ts = q.top();
        q.pop();
        d = ts.first;
        u = ts.second;
```

```
            if(daxet[u]) continue;
            daxet[u]= true;
            for(it=dinh[u].begin(); it!=dinh[u].end(); ++it)
            {
                v = (*it).second;
                k = (*it).first;
                if(dist[v] > d + k)
                {
                    dist[v] = d + k;
                    dem[v] = dem[u];
                    q.push(make_pair(dist[v],v));
                }
                else if(dist[v] == d + k)
                    dem[v]+= dem[u];
            }
            if(u==n)
            {
                printf("%d %lld",dist[n],dem[n]);
                break;
            }
        }
        return 0;
    }
```

Tác giả: Vũ Đình Trung

PHẦN IV. QUY HOẠCH ĐỘNG

Bài 1. 2356. Lát gạch - http://vn.spoj.pl/problems/LATGACH/

Cho một hình chữ nhật kích thước 2xN (1<=N<=100). Hãy đếm số cách lát các viên gạch nhỏ kích thước 1x2 và 2x1 vào hình trên sao cho không có phần nào của các viên gạch nhỏ thừa ra ngoài, cũng không có vùng diện tích nào của hình chữ nhật không được lát.

Input

Gồm nhiều test, dòng đầu ghi số lượng test T (T<=100).

T dòng sau mỗi dòng ghi một số N.

Output

Ghi ra T dòng là số cách lát tương ứng.

Example

Input:

3
1
2
3

Output:

1
2
3

Time: 1s

Giải

Đầu tiên ta xét hình chữ nhật 2x1 thì có 1 cách xếp đó là xếp 1 viên gạch 2x1.

Xét hình chữ nhật 2x2 thì có 2 cách xếp đó là xếp 2 viên 1x2 hoặc 2 viên 2x1.

Xét hình chữ nhật 2xi có các trường hợp sau với f(i) là số cách xếp cho hình chữ nhật 2xi.

<table>
<tr><td rowspan="2">2x(i-1)

f(i-1) cách xếp</td><td>2x1</td></tr>
<tr><td></td></tr>
</table>

<table>
<tr><td rowspan="2">2x(i-2)

f(i-2) cách xếp</td><td>1x2</td></tr>
<tr><td>1x2</td></tr>
</table>

=> f(i) = f(i-1) + f(i-2) với f(1) = 1 và f(2) = 2

=> f(N) là kết quả của bài toán

Công thức truy hồi trên chính là công thức của dãy Fibonacy nổi tiếng.

Giới hạn của bài này là N<=100, do đó f(N) sẽ rất lớn ta phải sử dụng cách cộng số lớn theo kiểu cộng mảng các chữ số.

Cài đặt của hàm cộng 2 vector lưu các chữ số của 2 số nguyên lớn như sau:

```
const int cs=10;
vector<int> cong(vector<int> b1, vector<int> b2)
{
    int n1 = b1.size();
    int n2 = b2.size();
    int n = n1>n2?n1:n2;
    vector<int> kq(n);
    int nho=0,k;
    for(int i=0; i<n; ++i)
    {
            if(i<n1 && i<n2)
            {
```

```
                k = b1[i] + b2[i] + nho;
                kq[i] = k%cs;
                nho =  k/cs;
            }
            else if(i<n1)
            {
                k = b1[i] + nho;
                kq[i] = k%cs;
                nho =  k/cs;
            }
            else if(i<n2)
            {
                k = b2[i] + nho;
                kq[i] = k%cs;
                nho =  k/cs;
            }
        }
        if(nho>0)
        {
           kq.push_back(nho);
        }
        return kq;
}
```

Sau khi đã có hàm cộng số lớn rồi thì mọi việc còn lại khá đơn giản. Cài đặt như sau (time: 0.00s ☺):

```
#include<iostream>
#include<vector>
using namespace std;

int main()
{
    int k,n;
    cin>>k;
    vector< vector<int> > kq;
    vector<int> b1;
    vector<int> b2;
    b1.push_back(1);
    b2.push_back(2);
    kq.push_back(b1);
    kq.push_back(b2);
    for(int i=2; i<100; ++i)
       kq.push_back(cong(kq[i-1],kq[i-2]));
    for(int i=0; i<k; ++i)
    {
       cin>>n;
       for(int j= kq[n-1].size()-1; j>=0; --j)
          cout<<kq[n-1][j];
       cout<<endl;
    }
    //system("pause");
    return 0;
```

```
}
```

Tác giả: Vũ Đình Trung

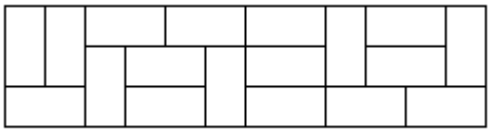
Bài 2. 3883. Lát gạch 3 - http://vn.spoj.pl/problems/M3TILE/

Đếm số cách lát hình chữ nhật 3xn bằng các domino 2x1?

Dữ liệu vào gồm nhiều testcase kết thúc là số -1.

Mỗi testcase là một số nguyên n, $0 \leq n \leq 30$.

Với mỗi test case, in ra số nguyên là đáp số trên một dòng.



SAMPLE INPUT

2
8
12
-1

SAMPLE OUTPUT

3
153
2131

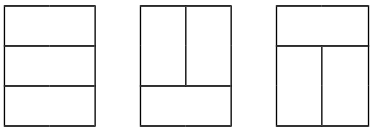
Time: 1s

Chú ý: n=0 kết quả trả về là 1.

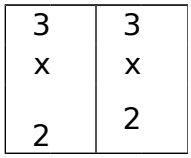
Giải

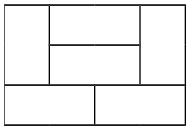
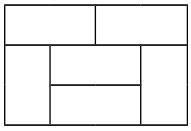
Bài này thì với n lẻ thì số ô 1x1 là lẻ trong khi các con domio đều có 2 ô, tổng của chúng luôn luôn chẵn, nên kết quả bài toán trả về là 0 có cách xếp nào phù hợp. Do đó ta chỉ xét với n chẵn.

Đầu tiên xét hình chữ nhật 3x2, dễ dàng nhận thấy có 3 cách xếp sau :



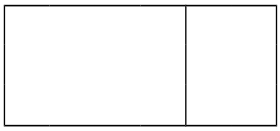
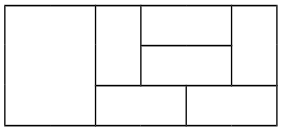
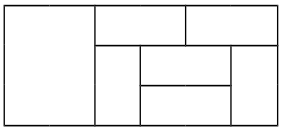
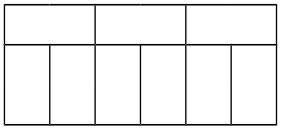
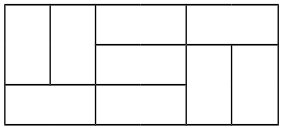
Xét tiếp hình chữ nhật 3x4, có các trường hợp sau:



3 x 3 = 9 cách + 2 cách => 11 cách

Xét tiếp hình chữ nhật 3x6, ta có các trường hợp sau:

Nếu gọi f(k) là số cách lát hình chữ nhật 3xk thì ta có:

f(6) = f(2) * f(6-2) + 2*[f(6-4) + f(6-6)] = 3*11 + 2*(3+1) = 41 cách.

Tổng quát ta có:

f(k) = 3*f(k-2) + 2*[f(k-4) + f(k-6) + .. + f(0)]

⇔ f(k) + f(k-4) = 3*f(k-2) + 3*f(k-4) + 2*[f(k-6) + f(k-8) + .. + f(0)]

⇔ f(k) + f(k-4) = 3*f(k-2) + f(k-2) = 4*f(k-2)

➔ f(k) = 4*f(k-2) - f(k-4)

Như vậy là ta đã tìm được công thức truy hồi ngắn gọn như sau:

f(k) = 4*f(k-2) - f(k-4) với f(0) = 1, f(2) = 3

f(n) chính là lời giải cho bài toán. Cài đặt như sau (time: 0.01s ☺):

```
#include<stdio.h>
int main()
{
    int n,i;
    int f[31]={0};
    f[0]=1;
    f[1]=0;
    f[2]=3;
    f[4]=11;
```

```
    for(i=6;i<=30;i+=2)
    {
        f[i]=4*f[i-2] - f[i-4];
    }
    scanf("%d",&n);
    while(n!=-1)
    {
        printf("%d\n",f[n]);;
        scanf("%d",&n);
    }
    return 0;
}
```

Tác giả: Vũ Đình Trung

Bài 3. 2795. Steps - http://vn.spoj.pl/problems/VSTEPS/

Bậc thang

Bờm chơi trò chơi điện tử Lucky Luke đến màn phải điều khiển Lucky leo lên một cầu thang gồm n bậc.

Các bậc thang được đánh số từ 1 đến n từ dưới lên trên. Lucky có thể đi lên một bậc thang, hoặc nhảy một bước lên hai bậc thang. Tuy nhiên một số bậc thang đã bị thủng do cũ kỹ và Lucky không thể bước chân lên được. Biết ban đầu, Lucky đứng ở bậc thang số 1 (bậc thang số 1 không bao giờ bị thủng).

Chơi đến đây, Bờm chợt nảy ra câu hỏi: có bao nhiêu cách để Lucky leo hết được cầu thang? (nghĩa là leo đến bậc thang thứ n). Bờm muốn nhờ bạn trả lời câu hỏi này.

Dữ liệu

- Dòng đầu tiên: gồm 2 số nguyên n và k, là số bậc của cầu thang và số bậc thang bị hỏng ($0 \le k < n \le 100000$).
- Dòng thứ hai: gồm k số nguyên cho biết chỉ số của các bậc thang bị hỏng theo thứ tự tăng dần.

Kết qủa

In ra phần dư của số cách Lucky leo hết cầu thang khi chia cho 14062008.

Ví dụ

Dữ liệu

4 2
2 3

Kết qủa

0

Dữ liệu

90000 1
49000

Kết qủa

4108266

Time: 1s

Giải

Bài này có công thức giống của dãy Fibonacy. Khi ta đứng ở bậc n-1 thì ta có thể đi lên 1 bậc để đến bậc n, còn khi ta đứng ở bậc n-2 thì ta có thể di chuyển lên 2 bậc để đến bậc n. Tuy nhiên ta phải xét đến các trường hợp mà bậc thang bị hỏng thì không có cách nào bước lên đó được, chỉ có thể nhảy 2 bước để qua và nếu 2 bậc liên tiếp mà bị hỏng thì rõ ràng là phải dừng lại. Chú ý là phải lấy phần dư khi chia số cách cho 14062008. Cài đặt như sau (time: 0.07s):

```
#include<stdio.h>
#define sc 14062008
int main()
{
    int i,n,k;
    unsigned long long a[100001];
    int tmp;
    a[0] = 0;
    a[1] = 1;
    scanf("%d%d",&n,&k);
    for(i=k-1; i>=0; --i)
    {
       scanf("%d", &tmp);
       a[tmp] = -1;
    }
    for(i=2; i<=n; i++)
    {
       if(a[i] == -1)
       {
          //printf("-1\n");
          continue;
       }
       if(a[i-1] == -1 && a[i-2] == -1)
       {
          a[n] = 0;
          break;
       }
```

```
            else if(a[i-1] == -1)
               a[i] = a[i-2];
            else if(a[i-2] == -1)
               a[i] = a[i-1];
            else
               a[i] = (a[i-1] + a[i-2]) % sc;
            //printf("%ld\n",a[i]);
         }
         printf("%ld\n", a[n] % sc);
         //system("pause");
         return 0;
      }
```

Tác giả: Vũ Đình Trung

Bài 4. 2782. Đường đi có tổng lớn nhất - http://vn.spoj.pl/problems/QBMAX/

Cho một bảng A kích thước m x n (1 <= m, n <= 100), trên đó ghi các số nguyên a_{ij} ($|a_{ij}| <= 100$). Một người xuất phát tại ô nào đó của cột 1, cần sang cột n (tại ô nào cũng được).

Quy tắc đi: Từ ô (i, j) chỉ được quyền sang một trong 3 ô (i, j + 1); (i - 1, j + 1); (i + 1, j + 1)

Input

Dòng 1: Ghi hai số m, n là số hàng và số cột của bảng.

M dòng tiếp theo, dòng thứ i ghi đủ n số trên hàng i của bảng theo đúng thứ tự từ trái qua phải

Output

Gồm 1 dòng duy nhất ghi tổng lớn nhất tìm được

Example

Input:

5 7
9 -2 6 2 1 3 4
0 -1 6 7 1 3 3
8 -2 8 2 5 3 2
1 -1 6 2 1 6 1
7 -2 6 2 1 3 7

Output:

41

Time: 1s

Giải

Đây là bài dạng quy hoạch động trên bảng. Quy tắc của bài này có thể thấy rõ trong bảng bên dưới:

j \ i	1	2	3	4	5	6	7
1	9	-2	6	2	1	3	4
2	0	-1	6	7	1	3	3
3	8	-2	8	2	5	3	2
4	1	-1	6	2	1	6	1
5	7	-2	6	2	1	3	7

Như vậy là từ 1 ô ta có thể đi sang các ô trên, phải, dưới ở cột bên kề bên phải, cũng có nghĩa là 1 ô có thể được đến từ các ô trên, trái, dưới ở cột kề bên trái.

Vì ta phải tìm tổng lớn nhất nên tại mỗi ô ta phải tìm tổng lớn nhất trong số tổng của các ô trên, trái, dưới ở cột kề bên trái, rồi cộng với giá trị tại ô đó là ra tổng lớn nhất phải tìm tại ô đó. Với bảng giá trị trên ta có bảng quy hoạch như sau:

j \ i	0	1	2	3	4	5	6	7
0	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN
1	0	9	7	14	16	24	27	35
2	0	0	8	14	23	24	31	34
3	0	8	6	16	18	28	31	36
4	0	1	7	13	18	19	34	35
5	0	7	5	13	15	19	22	41
6	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN	INT_MIN

Kết quả của bài toán là giá trị lớn nhất tại cột ngoài cùng bên phải, ở đây là giá trị 41.

Cài đặt như sau (time: 0.02s):

```
#include <stdio.h>
#define INT_MIN -2147483647
int m,n;
int a[100][100];
int b[102][101];
```

```
int main()
{
    int i,j,s;
    scanf("%d%d",&m,&n);
    for(i=0; i<m; ++i)
    {
       for(j=0; j<n; ++j)
          scanf("%d",&a[i][j]);
    }
    for(i=0; i<=m; ++i)
       b[i][0] = 0;
    for(i=1; i<=n; ++i)
    {
       b[0][i] = INT_MIN;
       b[m+1][i] = INT_MIN;
    }
    for(i=1; i<=n; ++i)
    {
       for(j=1; j<=m; ++j)
       {
          s = INT_MIN;
          if(s < b[j][i-1]) s = b[j][i-1];
          if(s < b[j-1][i-1]) s = b[j-1][i-1];
          if(s < b[j+1][i-1]) s = b[j+1][i-1];
          b[j][i] = a[j-1][i-1] + s;
       }
    }
    /*for(i=0; i<=m+1; ++i)
    {
       printf("\n");
       for(j=0; j<=n; ++j)
          printf("%-13d", b[i][j]);
    }*/
    s = INT_MIN;
    for(i=1; i<=m; ++i)
       if(s < b[i][n])
          s = b[i][n];
    printf("%d",s);
    return 0;
}
```

Tác giả: Vũ Đình Trung

PHẦN V. CÁC BÀI TOÁN KHÁC

Bài 1. Đấu giá - Chuyên tin 2010

Sở giao thông Hà Nội quyết định bán đấu giá các biển số xe đẹp để lấy tiền ủng hộ đồng bào lũ lụt miền Trung. Một biển số xe được gọi là đẹp nếu nó thỏa mãn các điều kiện sau:

- Là một số nguyên dương T mà $A \leq T \leq B$ trong đó A, B là hai số nguyên dương cho trước;
- T là một số nguyên tố;
- T là một số đối xứng (đọc T từ trái qua phải thu được kết quả giống như đọc T từ phải qua trái). Ví dụ 12321 là một số đối xứng.

Yêu cầu: Cho hai số nguyên dương A và B, hãy tìm số lượng các biển số xe đẹp.

Dữ liệu: Vào từ file văn bản AUCTION.INP gồm 1 dòng chứa hai số nguyên dương A và B($10^4 \leq A < B < 10^5$).

Kết quả: Đưa ra file văn bản AUCTION.OUT một số nguyên là số lượng biển số xe đẹp tìm được.

Ví dụ:

AUCTION.INP	AUCTION.OUT
11111 22222	23

Time : 2s

Gợi ý giải

Bài này trước tiên ta sẽ viết hàm kiểm tra tính đối xứng của một số có 5 chữ số. Hàm này ta có thể chuyển số nguyên a thành chuỗi số s[6] (dùng hàm sprintf(s , "%d", a)) hoặc sử dụng các phép toán chia và lấy phần dư để tách các chữ số của a, rồi kiểm tra nếu chữ số tại vị trí 0 và 4 giống nhau và chữ số tại vị trí 1 và 3 giống nhau thì trả về true, ngược lại trả về false.

Tiếp theo ta phải viết hàm kiểm tra tính nguyên tố của số nguyên a. Đơn giản chỉ cần kiểm tra nếu a chia hết cho bất kỳ số nguyên nào trong đoạn từ 2 đến căn bậc hai của a thì trả về false ngược lại trả về true.

Cuối cùng chỉ cần duyệt từ A đến B và đếm các số thỏa mãn đối xứng và là số nguyên tố.

Bạn đọc tự cài đặt.

Tác giả: Vũ Đình Trung

Bài 2. Trông xe - Chuyên tin 2010

Một bãi đỗ xe nhận trông xe trong vòng một tháng. Mỗi xe sẽ được gắn một số hiệu là một sốnguyên dương T ($10102010 \le T \le 10109999$). Hai xe khác nhau sẽ được gắn hai số hiệu khác nhau.

Một xe có thể ra vào bãi đỗ xe nhiều lần, mỗi lần vào bãi đỗ xe, người trông xe sẽ ghi vào sổ sách số hiệu của chiếc xe đó.

Cuối tháng dựa vào sổ ghi chép, người trông xe làm thống kê về số lần vào bãi đỗ xe của từng chiếcxe để tiến hành thu phí. Nếu một chiếc xe vào bãi đỗ xe **p** lần, cuối tháng chủ xe phải trả một lượng phí **C** được tính như sau:

- **$C = 100$ nếu $p \le 5$**
- **$C = 100 + (-5)$ nếu $p > 5$**

Yêu cầu: Tính tổng số phí người trông xe thu được vào cuối tháng.

Dữ liệu: Vào từ file văn bản PARK.INP có dạng:

- Dòng đầu chứa một số nguyên dương K ($0 < K \le 10^6$)
- K dòng tiếp theo, mỗi dòng chứa số hiệu một chiếc xe .

Kết quả: Đưa ra file văn bản PARK.OUT một số nguyên là tổng số phí thu được.

Ví dụ:

PARK.INP	PARK.OUT
7 10102010 10108888 10102010 10102010 10102010 10102010 10102010	201

Time : 2s

Gợi ý giải

Ta thấy giới hạn của bài toán = 10109999 - 10102010 + 1 = 7990 và k $<10^6$ nên hoàn toàn có thể áp dụng việc đếm theo chỉ số.

Trước tiên ta khai báo một mảng int dem[7991]. Đọc các đầu vào n tìm maxn và minn. Khi xét số hiệu của 1 chiếc xe nào đó ví dụ là n ta chỉ việc tăng dem[n-20102010] lên 1.

Cuối cùng duyệt lại mảng dem[] từ số hiệu nhỏ nhất (minn) đến cao nhất (maxn)để tính tổng chi phí thu được.

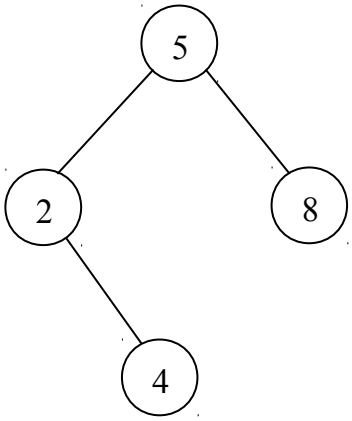
Tác giả: Vũ Đình Trung

Bài 3. 3480. Cây nhị phân tìm kiếm - http://vn.spoj.pl/problems/NKTREE/

Một trong những cấu trúc dữ liệu nổi tiếng để lưu trữ dữ liệu là cây nhị phân tìm kiếm. Mỗi nút trên cây có nhiều nhất là hai nút con và nhiều nhất là một nút cha. Các nút con được chia thành hai loại: nút con trái và nút con phải. Mỗi cây tìm kiếm có một nút không có nút cha gọi là nút gốc, và có ít nhất một nút không có nút con gọi là nút lá.

Mỗi một nút có gắn một giá trị nào đó thỏa điều kiện sau: Tại một nút v bất kỳ tất cả các giá trị thuộc cây con trái với gốc v nhỏ hơn giá trị tại nút v, và tất cả các giá trị ở các nút thuộc cây con bên phải với gốc v lớn hơn giá trị tại nút v.

Hình bên dưới mô tả một cây nhị phân tìm kiếm trong đó nút có giá trị 5 là gốc, các nút với giá trị 2, 4 và 8 là các nút lá.



Đường đi trên cây là dãy các giá trị tại các nút liên tiếp, trong đó mỗi nút sau là nút con trực tiếp của nút trước đó.

Yêu cầu: Cho một dãy gồm các giá trị đôi một khác nhau. Hãy cho biết tồn tại hay không cây tìm kiếm nhị phân, mà trên đó tồn tại một đường đi với dãy giá trị tương ứng là dãy đã cho.

Ví dụ, tồn tại cây nhị phân tìm kiếm với dãy 5 1 3 2, còn không tồn tại cây nhị phân tìm kiếm với dãy 5 2 3 1.

Dữ liệu

Lần lượt liệt kê các giá trị của dãy đã cho. Hai phân tử được ghi cách nhau bởi khoảng trắng hoặc dấu xuống dòng. Số lượng phần tử của dãy không vượt quá 50 000 và mỗi phần tử của dãy có giá trị tuyệt đối không vượt quá 2^{31}.

Kết quả

Ghi ra từ "YES", nếu tồn tại cây, tương ứng dãy đã cho hoặc từ "NO" trong trường hợp ngược lại.

Ví dụ

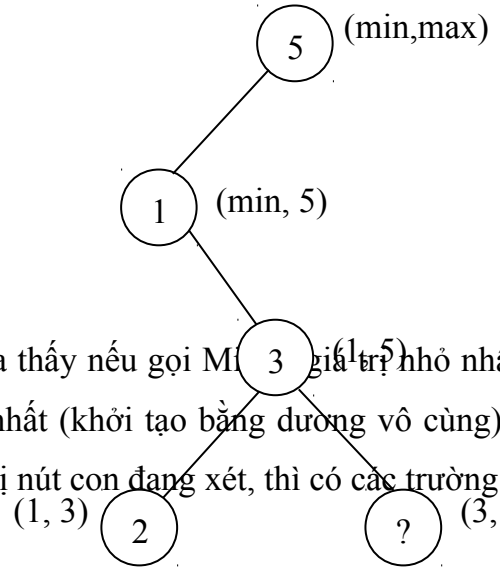
Dữ liệu

Dữ liệu	Kết quả
5 1 3 2	YES
5 2 3 1	NO

Time: 1s

Gợi ý giải

Với bài này chúng ta cần quan tâm đến các giá trị đó là giá trị min hiện tại, max hiện tại, giá trị nút cha trực tiếp và giá trị nút con đang xét. Có thể thấy rõ hơn ở hình dưới:



Nhìn vào cây này ta thấy nếu gọi Min là giá trị nhỏ nhất (khởi tạo bằng âm vô cùng), Max là giá trị lớn nhất (khởi tạo bằng dương vô cùng), b là giá trị trước đó (nút cha trực tiếp), c là giá trị nút con đang xét, thì có các trường hợp sau:

- Nếu c=b hoặc c<=Min hoặc c>=Max thì đáp án bài toán là "NO" dừng đọc dữ liệu.
- Nếu c thuộc khoảng (Min, Max) và c < b thì gán lại Max = b
- Nếu c thuộc khoảng (Min, Max) và c > b thì gán lại Min = b

Cuối cùng nếu tất cả các giá trị đọc được đều thỏa mãn thì in ra YES.

Chú ý: đọc dữ liệu bài này là đọc cho đến khi hết dữ liệu nên trong C ta có thể dùng vòng lặp ***while(scanf("%f", &c) >0){ }*** hoặc trong C++ dùng ***while((cin>>c)>0) { }***. Ở đây phải dùng kiểu ***float*** trong C hoặc ***long long*** trong C++ cho các biến vì khả năng bộ test của bài này đã bị sửa, có giá trị vượt quá kiểu số nguyên 32 bit.

Tác giả: Vũ Đình Trung

Bài 4. 2786. Cây khung nhỏ nhất (HEAP) - http://vn.spoj.pl/problems/QBMST/

Cho đơn đồ thị vô hướng liên thông G = (V, E) gồm n đỉnh và m cạnh, các đỉnh được đánh số từ 1 tới n và các cạnh được đánh số từ 1 tới m. Hãy tìm cây khung nhỏ nhất của đồ thị G

Input

Dòng 1: Chứa hai số n, m (1 <= n <= 10000; 1 <= m <= 15000)

M dòng tiếp theo, dòng thứ i có dạng ba số nguyên u, v, c. Trong đó (u, v) là chỉ số hai đỉnh đầu mút của cạnh thứ i và c trọng số của cạnh đó (1 <= u, v <= n; 0 <= c <= 10000).

Output

Gồm 1 dòng duy nhất: Ghi trọng số cây khung nhỏ nhất

Example

Input:

6 9

1 2 1

1 3 1

2 4 1

2 3 2

2 5 1

3 5 1

3 6 1

4 5 2

5 6 2

Output:

5

Time: 1s

Gợi ý giải

Tương tự bài **NKCITY** trong phần **ĐỒ THỊ**, chỉ có khác là kết quả trả về là tổng trọng số của cây khung nhỏ nhất.

Tác giả: Vũ Đình Trung

Bài 5. 4031. Mass of Molecule- http://vn.spoj.pl/problems/MMASS/

Hóa chất chỉ gồm các nguyên tố C, H, O có trọng lượng 12,1, 16 tương ứng.

Nó được biểu diễn dạng "nén", ví dụ COOHHH là CO2H3 hay CH (CO2H) (CO2H) (CO2H) là CH(CO2H)3. Nếu ở dạng nén thì số lần lặp >=2 và <=9.

Tính khối lượng hóa chất.

Input

Gồm một dòng mô tả hóa chất không quá 100 kí tự chỉ gồm C, H, O, (,), 2,..,9.

Output

Khối lượng của hóa chất, luôn <=10000.

Sample

MASS.IN	MASS.OUT
COOH	45
CH(CO2H)3	148
((CH)2(OH2H)(C(H))O)3	222

Time: 1s

Gợi ý giải

Cơ chế để giải bài này giống với kiểu ngăn xếp - stack, vào sau ra trước.

Đầu tiên ta sẽ đọc toàn bộ xâu: s, sau đó xét lần lượt các ký tự trong xâu: s[i], có các trường hợp sau:

- Nếu s[i] = ‘(‘ thì ta đưa vào stack giá trị 0
- Nếu s[i] ∈ [‘2’,’9’] thì lấy giá trị ở cuối stack ra nhân với giá trị tương ứng s[i] - ‘0’, rồi lấy kết quả đưa vào stack.
- Nếu s[i] ∈ {‘C’, ‘O’, ‘H’] thì đưa vào stack trọng lươngj của nguyên tố đó
- Nếu s[i] = ‘)’ thì ta lấy lần lượt các giá trị ở cuối stack ra đồng thời tính tổng của các giá trị này cho đến khi lấy ra giá trị 0, rồi lại đưa vào stack tổng vừa tìm được.

Duyệt xong ta sẽ tính tổng các giá trị còn lại trong stack => khối lượng của hóa chất.

Minh họa với ví dụ số 2 ở trên:

s[i]	Stack
C	{12}
H	{12, 1}
(	{12, 1, 0}
C	{12, 1, 0, 12}
O	{12, 1, 0, 12, 16}
2	{12, 1, 0, 12, 32}
H	{12, 1, 0, 12, 32, 1}
)	{12, 1, 45}
3	{12, 1, 135} = 148

Tác giả: Vũ Đình Trung
