MỘT SỐ BÀI TOÁN QUY HOẠCH ĐỘNG

Bài 1. Đề thi ACM 2016 Miền Bắc

Byteland là một xứ sở rất đẹp và yên bình. Ban đầu, vua Byteland đã chia vùng đất của mình thành m hàng và n cột, giao điểm của hàng thứ i và cột thứ j được gọi là tỉnh ij với dân số Pij. Sau đó, nhận thấy rằng chia quá nhiều tỉnh sẽ dẫn tới sự khác biệt về văn hóa, kinh tế nên nhà vua quyết định chia lại vương quốc thành 4 tỉnh. Nhà vua thực hiện việc này bằng cách chia đất theo một đường ngang và một đường dọc (chạy theo ranh giới của các tỉnh cũ).

Nhà vua không muốn có sự khác biệt dân số quá lớn nên ngài yêu cầu bạn tìm ra cách chia sao cho tỉnh đông dân nhất và tỉnh thưa dân nhất có sự chênh lệch dân số thấp nhất.

Dữ liệu đầu vào

– Dòng đầu tiên là 2 số nguyên m, n; (2<=m<=1000, 2<=m<=1000)

– m dòng tiếp theo, mỗi dòng gồm n số liệu là dân số các tỉnh trước khi chia (không quá 1000)

Dữ liệu đầu ra

Độ chênh lệch dân số ít nhất giữa tỉnh đông dân nhất và tỉnh thưa dân nhất

Ví dụ:

input	output
2 2 1 2 3 4	3
3 3 1 1 9 1 1 1 8 1 1	9

Code Tham khảo ACM 2016 Miền Bắc

```
#include <iostream>
#include <climits>
#include <cmath>
using namespace std;
const int nmax = 1010;
 long a[nmax][nmax], n, m, res = LONG_MAX, tmp, gtmin, gtmax;
 inline long get(int i, int j, int u, int v)
        {
           return a[u][v]-a[i-1][v]-a[u][j-1]+a[i-1][j-1];
        }
 int main()
        {
           cin >> m >> n;
```

```
    for (int i=0; i<=m+1; i++)
        a[i][0]=0;
    for (int j=0; j<=n+1; j++)
        a[0][j]=0;

    for (int i = 1; i<=m; i++)
        for (int j=1; j<=n; j++)
            cin >> a[i][j];

    for (int i = 1; i<=m; i++)
        for (int j=1; j<=n; j++)
            a[i][j]=a[i][j]+a[i-1][j]+a[i][j-1]-a[i-1][j-1];

    for (int i = 1; i<=m-1; i++)
        for (int j=1; j<=n-1; j++)
        {
            tmp = get(1,1,i,j); // tren trai
            gtmin = tmp;
            gtmax = tmp;
            tmp = get(1,j+1,i,n); // tren phai
            gtmin = min(gtmin, tmp);
            gtmax = max(gtmax, tmp);
            tmp = get(i+1,1,m,j); // duoi trai
            gtmin = min(gtmin, tmp);
            gtmax = max(gtmax, tmp);
            tmp = get(i+1,j+1,m,n); // duoi phai
            gtmin = min(gtmin, tmp);
            gtmax = max(gtmax, tmp);
            res = min(res,(long)abs(gtmin-gtmax));
        }
    cout << res;
    return 0;
}
```

Bài 2. Đề bài MPILOT spoj - PILOTS

Charlie sở hữu vài cái máy bay bà già và cần tối ưu chi phi để kiếm lời
Có N phi công (N chẵn) và cần có N/2 phi hành đoàn.Mỗi phi hành đoàn gồm 2 người- 1 lái chính, 1 trợ lí. Lái chính phải cao tuổi hơn trợ lý. Hợp đồng cho mỗi phi công có ghi mức lương nếu anh ta là lái chính hoặc là trợ lí. Với mỗi 1 hợp đồng thì lương lái chính > lương trợ lí.
Tìm cách ghép cặp sao cho tổng lương phải trả cho N người là ít nhất.

Input

Dòng đầu là N (N chẵn), số phi công, $2 \le N \le 10,000$.
N dòng tiếp theo, mỗi dòng là 2 số X,Y là lương phi công thứ i nếu làm lái chính hoặc trợ lí,$1 \le Y < X \le 100,000$.
Các phi công sắp tăng dần theo tuổi.

Output

Lương nhỏ nhất cần trả.

Sample

input	output
4 5000 3000 6000 2000 8000 1000 9000 6000	19000
6 10000 7000 9000 3000 6000 4000 5000 1000 9000 3000 8000 6000	32000

Gợi ý thuật toán MPILOT spoj

Sử dụng TT QHĐ
Gọi f[i][j] là số tiền min khi xét tới phi công thứ i khi còn dư j trợ lí.
Do các phi công tăng dần theo tuổi nên có thể tính f[i][j] qua f[j-1][…]

Code mẫu MPILOT spoj

```
#include <bits/stdc++.h>
#define fs first
#define sc second
using namespace std;
typedef pair<int,int> II;
   int n,x[10001],y[10001],f[10001][10001];
int main()
{
   //freopen("mpilot.inp","r",stdin);
   //freopen("mpilot.out","w",stdout);
   scanf("%d",&n);
   for(int i=1; i<=n; i++)
      scanf("%d%d",&x[i],&y[i]);
   f[1][1]=y[1];
   for(int i=2; i<=n; i++)
   {
      f[i][0]=f[i-1][1]+x[i];
```

```
            f[i][i]=f[i-1][i-1]+y[i];
            for(int j=1; j<=i-1; j++)
                f[i][j]=min(f[i-1][j-1]+y[i],f[i-1][j+1]+x[i]);
        }
        printf("%d",f[n][0]);
    }
```

Bài 3. COUNTPL - Đếm số Palindrome

Palindrome là xâu ký tự mà nếu đọc nó từ trái sang phải cũng như từ phải sang trái ta được cùng một xâu. Một xâu ký tự bất kỳ luôn có thể biểu diễn như là một dãy các palindrome nếu như ta coi xâu chỉ gồm một ký tự luôn là một palindrome. Ví dụ: Xâu 'bobseesanna' có thể biểu diễn dưới dạng dãy các palindrome theo nhiều cách, chẳng hạn:

'bobseesanna' = 'bob' + 'sees' + 'anna'

'bobseesanna' = 'bob' + 's' + 'ee' + 's' + 'anna'

'bobseesanna' = 'b' +'o' + 'b' + 'sees' + 'a' + 'n' + 'n' + 'a'

Yêu cầu

Cho xâu ký tự s, cần tìm cách biểu diễn xâu s dưới dạng một dãy gồm số ít nhất các palindrome. Ví dụ: Cho s='bobseesanna', do ta có 'bobseesanna' = 'bob' + 'sees' + 'anna' và không thể biểu diễn 'bobseesanna' bởi ít hơn là 3 palindrome nên biểu diễn này chính là biểu diễn cần tìm.

Input

Gồm một dòng chứa xâu ký tự s gồm không quá 255 ký tự.

Output

Gồm một dòng duy nhất ghi k là số lượng ít nhất các palindrome trong biểu diễn tìm được.

Ví dụ

Input	Output
bobseesanna	3

Gợi ý Đếm số Palindrome

Lập công thức:

-Gọi f[i] là số cách phân tích xâu từ s1 đến si

Ta có: f[i]= $\sum$ f[j] với 1<=j<=i

sj, sj+1,…, si là một palindrome.

Code Đếm số Palindrome

```
#include <bits/stdc++.h>
#define t 1000000007
using namespace std;
int n;
char a[1001];
long long f[1001];
bool ok(int i, int j)
```

```
{
   while (i <= j)
   {
      if (a[i] != a[j]) return false;
      i++;
      j--;
   }
   return true;
}
int main()
{
   scanf("%d\n", &n);
   for (int i = 1; i <= n; i++)
   {
      scanf("%c", &a[i]);
      f[i] = 0;
   }
   f[0] = 1;
   f[1] = 1;
   for (int i = 2; i <= n; i++)
   {
      for (int j = 1; j <= i; j++)
         if (ok(j, i)) f[i] = ((long long)f[i] + f[j - 1]) % t;
   }
   cout << f[n];
}
```

Bài 4. Đề bài NKCABLE Spoj- NỐI MẠNG

Các học sinh khi đến thực tập trong phòng máy tính thường hay chơi trò chơi điện tử trên mạng. Để ngăn ngừa, người trực phòng máy đã ngắt tất cả các máy tính ra khỏi mạng và xếp chúng thành một dãy trên một cái bàn dài và gắn chặt máy xuống mặt bàn rồi đánh số thứ tự các máy từ 1 đến N theo chiều từ trái sang phải. Các học sinh tinh nghịch không chịu thua, họ đã quyết định tìm cách nối các máy trên bàn bởi các đoạn dây nối sao cho mỗi máy được nối với ít nhất một máy khác. Để tiến hành công việc này, họ đã đo khoảng cách giữa hai máy liên tiếp. Bạn hãy giúp các học sinh này tìm cách nối mạng thoả mãn yêu cầu đặt ra sao cho tổng độ dài cáp nối phải sử dụng là ít nhất.

Dữ liệu

- Dòng đầu tiên chứa số lượng máy N ($1 \le N \le 25000$).
- Dòng thứ i trong số N-1 dòng tiếp theo chứa các khoảng cách từ máy i đến máy i+1 (i=1,2,…,N-1). Giả thiết rằng khoảng cách từ máy 1 đến máy N không vượt quá 10^6.

Kết quả
Ghi ra độ dài của cáp nối cần sử dụng.
Ví dụ

Input	Output
6 2 2 3 2 2	**7**

Gợi ý NKCABLE Spoj

-Các bạn dùng thuật toán QHĐ:
–Cơ sở: F[1]=A[1], F[2]=A[2], F[3]=F[1]+F[2].
–Công thức truy hồi: F[i]=Min(F[i-1]+A[i], F[i-2]+A[i])

Code Tham Khảo NKCABLE Spoj

```
#include<bits/stdc++.h>
using namespace std;
long i,n;
 long f[25000],a[25000];
 int main()
 {
   cin>>n;
    for(i=1;i<n;i++)
        cin>>a[i];
    f[1]=0;
    f[2]=a[1];
    f[3]=a[1]+a[2];
    for(i=4;i<=n;i++)
      f[i]=min(f[i-1]+a[i-1],f[i-2]+a[i-1]);
   cout<<f[n];
return 0;
}
```

Bài 5. Đề bài QBMAX spoj – đường đi có tổng lớn nhất 2

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

Ví dụ

Input	Output
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	41

Hướng dẫn QBMAX spoj – Đường đi có tổng lớn nhất 2

Bài này công thức QHĐ dễ suy ra được

– F[i,j]=max(F[i-1,j-1],F[i,j-1],F[i+1,j-1]) + a[i,j]; với j=2..n, i=1..m.

– Nghiệm bài toán là max(F[i,n]) (với i=1..m)

Code tham khảo QBMAX spoj

```
program QBMAX;
const   fi='';
        nmax=101;
        vc=maxint;
type    matran=array[0..nmax,0..nmax] of integer;

var
        a:matran;
        m,n:byte;
        f:text;

procedure docfile;
var i,j:byte;
begin

        assign(f,fi);
        reset(f);
        readln(f,m,n);

        for i:=0 to m+1 do
                for j:=0 to n+1 do
                        a[i,j]:=-vc;

        for i:=1 to m do
                for j:=1 to n do
```

```
                read(f,a[i,j]);
        close(f);
end;

function max(a,b,c:longint):longint;
begin
        max:=a;
        if max<b then
                max:=b;
        if max<c then
                exit(c);
end;

procedure bpa;
var i,j:byte;
        kq:longint;
begin
        for j:=2 to n do
                for i:=1 to m do
                        a[i,j]:=max(a[i-1,j-1],a[i,j-1],a[i+1,j-1]) + a[i,j];
        kq:=-maxlongint;
        for i:=1 to m do
                if kq<a[i,n] then
                        kq:=a[i,n];
        writeln(kq);
end;

begin
        docfile;
        bpa;
end.
```

Bài 6. Đề bài Quy hoạch động Đường đi có tổng lớn nhất 1

Cho ma trận A hình vuông có kích thước n*n. Đầu tiên bạn ở ô có tọa độ [1,1], bạn được phép đi sang phải và đi xuống dưới ô kề cạnh. Hãy tìm đường đi có tổng lớn nhất khi đi đến ô [n,n].

Input

-Dòng đầu là số nguyên dương N (n<= 1000).

– n dòng tiếp theo, mỗi dòng gồm n số nguyên dương biểu diễn ma trận A.

Output

– một dòng duy nhất là kết quả bài toán.

Bài này mình viết để các bạn tham khảo một dạng QHĐ cơ bản, nên về dữ liệu mình không yêu cầu A[i,j] có giới hạn bao nhiêu, và nhiều dữ kiện khác.

Ví dụ

Input	Output
4 **9** 2 3 4 **9** 0 2 2 **9 9 9** 2 2 2 **9 9**	63

Hướng dẫn giải bài QHĐ đường đi có tổng lớn nhất

– Gọi F[i,j] là tổng lớn nhất có được khi đi đến ô i,j.

– Khởi tạo F[i,0]=F[0,j]=0;

– Có 2 cách đi đến ô [i,j], một là đi từ trên xuống, hai là từ phải qua, như vậy ta có công thức lần lượt là F[i-1,j] và F[i,j-1]

– Công thức tổng quát F[i,j]=max(F[i-1,j],F[i,j-1])+a[i,j]; (i,j=1..n);

– Kết quả bài toán là F[n,n].

Code Quy hoạch động đường đi có tổng lớn nhất

```
#include <bits/stdc++.h>
using namespace std;
int n;
int a[100000][100000], f[100000][100000];
int main()
{
   cin>>n;
   for(int i=1; i<=n; i++)
      for(int j=1; j<=n; j++)
      cin>>a[i][j];
   for(int i=0; i<=n; i++)
      f[i][0]=f[0][i]=0;
   for(int i=1; i<=n; i++)
      for(int j=1; j<=n; j++)
   f[i][j]=max(f[i-1][j],f[i][j-1]) +a[i][j];
cout<<f[n][n];
}
```

Bài 7. Đề bài LIQ Dãy con tăng dài nhất

Cho một dãy số nguyên gồm N phần tử A[1], A[2], … A[N].

Biết rằng dãy con tăng đơn điệu là 1 dãy $A[i_1]$,… $A[i_k]$ thỏa mãn

$i_1 < i_2 < \ldots < i_k$ và $A[i_1] < A[i_2] < .. < A[i_k]$. Hãy cho biết dãy con tăng đơn điệu dài nhất của dãy này có bao nhiêu phần tử?

Input

- Dòng 1 gồm 1 số nguyên là số N ($1 \le N \le 1000$).

- Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] ($1 \le A[i] \le 10000$).

Output

Ghi ra độ dài của dãy con tăng đơn điệu dài nhất.

Ví dụ:

Input	Output
6 1 2 5 4 6 2	4

Giải thích test ví dụ: Dãy con dài nhất là dãy A[1] = 1 < A[2] = 2 < A[4] = 4 < A[5] = 6, độ dài dãy này là 4.

Gợi ý: Sử dụng phương pháp Quy Hoạch Động. F[i]: Độ dài dãy con đơn điệu tăng dài nhất mà phần tử cuối cùng là số A[i] này.

Hướng dẫn LIQ SPOJ Dãy con tăng dài nhất

– Gọi F[i] là độ dài dãy con tăng dần dài nhất kết thúc tại i

– khởi tạo F[i]=1; (với i=1..n)

– F[i]=max(F[i],F[j]+1) (A[i]>A[j] ; i=1..n, j=1..i-1).

– nghiệm bài toán Max(F[i]) (i=1..n)

Code tham khảo LIQ SPOJ Dãy con tăng dài nhất

```
#include <stdio.h>
#include <algorithm>
using namespace std;
#define nmax=1000
int n;
int a[1000], f[1000];
int main()
{
   scanf("%d",&n);
   int i,j;
   for (i=1; i<=n; i++)
   {
      scanf("%d",&a[i]);
      f[i]=1;
   }
   for (i=1; i<=n; i++)
   {
      for (j=i+1; j<=n; j++)
      {
         if (a[i]<a[j]) f[j]=max(f[i]+1,f[j]);
      }
   }
   int res=0;
   for (i=1; i<=n; i++) res=max(res,f[i]);
   printf("%d",res);
```

```
    return 0;
}
```

Code khác:

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

int main ()
{
	int n;
	cin>>n;
	long arr[1003];
	long F[1003];
	for (int i=1; i<=n; i++)
		cin>>arr[i];
	arr[0] = 0;
	F[0] = 0;
	for (int i=1; i<=n; i++)
	{
		F[i] = 1;
		for (int j=i-1; j>=1; j--)
		{
			if (arr[i]>arr[j])
			{
				F[i]=max(F[i], F[j]+1);
			}
		}
	}

	long dmax = 1;
	for (int i=1; i<=n; i++)
		if (F[i]>=dmax)
			dmax = F[i];
	cout<<dmax;
	return 0;
}
```

<u>Bài 8. Đề bài AMSSEQ spoj – dãy số</u>

Cho 1 dãy số gồm N phần tử ($N \leq 10000$), mỗi phần tử có 1 giá trị nằm trong khoảng [-1000, 1000]. Ban đầu, bạn sẽ ở vị trí ô số 0 với tổng điểm là 0. Mỗi nước đi, người chơi có thể di chuyển sang phải tối thiểu là 1 bước và tối đa là K bước ($K \leq 10$) . Khi dừng lại ở 1 ô nào đó thì giá trị của ô đó sẽ được cộng vào tổng điểm. Bạn có

thể dừng cuộc chơi bất cứ lúc nào. Hãy tìm cách chơi sao cho tổng điểm nhận được là nhiều nhất.

Dữ liệu vào

- Dòng đầu tiên chứa 2 số N, K.
- Dòng thứ 2 chứa N số của dãy, mỗi số cách nhau 1 dấu cách. Mỗi số nằm trong khoảng [-1000, 1000]

Dữ liệu ra

- Số điểm lớn nhất có thể đạt được.

Giới hạn:

- $N \leq 10000$.
- $K \leq 10$.
- Trong 20% số test có $N \leq 10$

Ví dụ

Input	Output
5 2 -2 3 -6 -4 5	4

Giải thích:

– Ta có thể đi theo thứ tự 0 -> 2 -> 4 -> 5. Số điểm đạt được là 0 + 3 – 4 + 5 = 4.

Thuật Toán AMSSEQ spoj

– Sử dụng QHĐ.

– gọi F[i] là số điểm lớn nhất đạt được khi đến ô vị trí i.

– khởi tạo F[1]=max(0,a[1]);

– F[i]=max(F[i],F[i-j]+a[i]) với (i=2..n ; j=1..k)

Kết quả bài toán là Max(F[i]) với (i=1..n).

Code tham khảo AMSSEQ spoj

```
uses math;
const   fi='';
        nmax=10000;
type    data=longint;
var
        F,A:array[-50..nmax+1] of data;
        n,k:data;

procedure docfile;
var
        f:text;
        i:data;
begin
        assign(f,fi); reset(f);
        readln(f,n,k);
        for i:=1 to n do
                read(f,a[i]);
```

```
        close(f);
end;

procedure QHD;
var     i,j,res:data;
begin
        res:=0;
        fillchar(F,sizeof(f),0);
        f[1]:=max(0,a[1]);
        for i:=2 to n do
                begin
                        F[i]:=low(data);
                        for j:=1 to k do
                                F[i]:=max(F[i],F[i-j]+A[i]);
                        res:=max(res,f[i]);
                end;
        writeln(res);
end;

begin
        docfile;
        QHD;
end.
```

Bài 9. Đề bài P152PROE spoj PTIT – đếm số cách

Cho 1 dãy số gồm n số nguyên a[1], a[2], …, a[n]. Đếm số cách chia dãy thành 3 phần bằng nhau, hay nói cách khác là đếm số cặp i, j thỏa mãn:

$$\sum_{k=1}^{i-1} a_k = \sum_{k=i}^{j} a_k = \sum_{k=j+1}^{n} a_k$$

Input

Dòng đầu tiên chứa số n (1 ≤ n ≤ 5*10^5).

Dòng thứ 2 gồm n số nguyên a[1], a[2], …, a[n] (0 <= |a[i]| <= 10^9) là các phần tử của dãy.

Output

In ra kết quả của bài toán.

Ví dụ:

Input	Output
5 1 2 3 0 3	2
2 4 1	0

Gợi ý P152PROE spoj PTIT

Problem E: Đếm số cách

Gọi sum là tổng của tất cả các phần tử trong dãy:

Nếu sum % 3 != 0, trường hợp này bỏ qua, in ra 0.

Tại vị trí i mà tại đó tổng từ 1 đến i = sum/3, tính tổng số lượng các phần tử vị trí j (i < j <= n)

sao cho tổng từ j đến n = sum/3. Cộng dồn tổng vừa tìm được vào kết quả.

Để xử lý phần tìm số lượng vị trí j, ta có thể chuẩn bị trước 1 mảng bằng QHĐ (Mảng tổng).

code tham khảo P152PROE spoj PTIT

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

long long sum[500010];

int main()
{
   int n;
   cin >> n;
   for (int i = 1; i <= n; i++)
   {
      int n1;
      cin >> n1;
      sum[i] = sum[i - 1] + n1;
   }
   long long tong = 0;
   if (sum[n] % 3 == 0)
   {
      int dem = 0;
      for (int i = n - 1; i>0; i--)
      {
         if (sum[i] == sum[n] / 3)
         {
            tong = tong + dem;
         }
         if (sum[i] == sum[n] / 3 * 2)
         {
            dem++;
         }
      }
   }
   cout << tong << endl;
```

}

Bài 10. Đề bài BCINCSEQ spoj PTIT – đoạn tăng

Cho dãy số nguyên A = (a_1, a_2, …, a_n). Hãy tìm một đoạn dài nhất gồm các phần tử **liên tiếp** trong dãy A có thứ tự không giảm

Quy ước: Đoạn chỉ gồm đúng 1 phần tử trong A cũng được coi là có thứ tự không giảm

Dữ liệu:

l Dòng 1 chứa số nguyên dương $n \leq 10^5$

l Dòng 2 chứa n số nguyên a_1, a_2, …, a_n (“i: $|a_i| \leq 10^9$) cách nhau ít nhất một dấu cách

Kết quả:một số nguyên duy nhất là số phần tử trong đoạn tìm được

Ví dụ

INPUT	OUTPUT
1188 99 11 22 22 33 11 66 33 44 77	4

2. Hướng dẫn BCINCSEQ spoj PTIT

– Gọi F[i] là độ dài xâu con liên tiếp dài nhất kết thúc tại phần tử có giá trị là A[i].

– Nếu a[i]>=A[i-1] thì F[i]:=F[i-1]+1

– Ngược lại F[i]:=1

Kết quả bài toán là F[n].

Khởi tạo F[0]=0; A[0]=dương vô cực.

code tham khảo BCINCSEQ spoj PTIT

```
CONST   FI='';
        nmax=100000;
type    data=longint;
var
        f:text;
        a:array[0..nmax+1] of data;
        n,i,j,max:data;
        C:array[0..nmax+1] of data;

procedure docfile;
var     i,j:data;
begin
        assign(f,fi); reset(f);
        readln(f,n);
        for i:=1 to n do
                read(f,a[i]);
        a[0]:=high(data);
        close(f);
end;
```

```
begin
    docfile;
    max:=0;
    for i:=1 to n do
        if a[i-1]<=a[i] then
            C[i]:=C[i-1]+1
        else
            C[i]:=1;
    for i:=1 to n do
        if c[i]>max then
            max:=c[i];
    writeln(max);
end.
```

Bài 11. Đề bài P145PROI PTIT spoj – Mật khẩu

Một xâu ký tự được gọi là mật khẩu *"an toàn"* nếu xâu có độ dài ít nhất bằng 6 và xâu chứa ít nhất một chữ cái in hoa , một chữ cái thường , một chữ số .
Ví dụ, 'a1B2C3', 'tinHoc6' là hai mật khẩu *"an toàn"*. Còn 'a1B2C', 'a1b2c3', 'A1B2C3', 'tinHoc' đều không phải là mật khẩu *"an toàn"*.
Một lần, Tí nhìn thấy một xâu S, chỉ gồm các loại ký tự: chữ cái in hoa, chữ cái thường và chữ số. Tí muốn tự kiểm tra khả năng đoán nhận mật khẩu bằng cách đếm xem có bao nhiêu cặp chỉ số (i, j) thỏa mãn điều kiện: $1 \le i < j \le length(S)$ và xâu con gồm các ký tự liên tiếp từ i đến j của S là mật khẩu "an toàn".
Cho xâu S, các hãy tính số lượng cặp chỉ số (i, j) thỏa mãn điều kiện nêu trên.

Input

Một dòng chứa xâu S có độ dài <= 10^6.

Output

In ra một số nguyên duy nhất là số cặp chỉ số (i, j) tìm được.

Ví dụ:

Input	Output
abc3456789PQ	6
abc123	0

Code tham khảo P145PROI PTIT spoj O(n)

```
program bt;
uses    math;
const   fi='';
    nmax=1000000;
var
    f:text;
    S:ansistring;
```

```
        n:longint;
        L:array[1..nmax+1,'1'..'3'] of longint;

procedure docfile;
var
        c:char;
begin
        assign(f,fi);
        reset(f);
        s:='';
        while not eoln(f) do
                begin
                        read(f,c);
                        case c of
                        'A'..'Z': s:=s+'3';
                        'a'..'z': s:=s+'2';
                        '0'..'9': s:=s+'1';
                        end;
                end;
        close(f);
        n:=length(s);
end;

procedure taoL;
var     i:longint;
        j:char;
begin
        for j:='1' to '3' do
                L[n+1,j]:=n+1;
        for i:=n downto 1 do
                begin
                        L[i,'1']:=L[i+1,'1'];
                        L[i,'2']:=L[i+1,'2'];
                        L[i,'3']:=L[i+1,'3'];

                        for j:='1' to '3' do
                                if s[i]=j then
                                        L[i,j]:=i;
                end;
end;
```

```
procedure xuli;
var     i:longint;
        dem:int64;
begin
        dem:=0;
        for i:=1 to n-5 do
                dem:=dem+n-max(max(i+5,L[i,'1']),max(L[i,'2'],L[i,'3']))+1;

        writeln(dem);
end;


begin
        docfile;
        taoL;
        xuli;   // readln;
end.
```

Code C++:

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

struct data
{
        int hv_1;
        int hv_A;
        int hv_a;
} typedef data;

data pass[1000006];

int check_safe (int begin, int end)
{
        int num_1=pass[end].hv_1-pass[begin-1].hv_1;
        int num_a=pass[end].hv_a-pass[begin-1].hv_a;
        int num_A=pass[end].hv_A-pass[begin-1].hv_A;
        if (num_1>0 && num_a>0 && num_A>0 && end-begin+1>=6) return
1;
        else return 0;
}

int BSearch (int begin, int front, int back)
```

```
{
	int vt=-1;
	while (front<=back && back-begin+1>=6)
	{
		int mid = (front+back)/2;
		if (check_safe (begin, mid)==1)
		{
			vt=mid;
			back=mid-1;
		}
		else front=mid+1;
	}
	return vt;
}

int main ()
{
	string xau;
	cin>>xau;
	pass[0].hv_1=0;
	pass[0].hv_a=0;
	pass[0].hv_A=0;
	int n=xau.length();
	for (int i=1; i<=n; i++)
	{
		if (xau[i-1]>='0' && xau[i-1]<='9')
		{
			pass[i] = pass[i-1];
			pass[i].hv_1++;
		}
		else if (xau[i-1]>='a' && xau[i-1]<='z')
		{
			pass[i] = pass[i-1];
			pass[i].hv_a++;
		}
		else if (xau[i-1]>='A' && xau[i-1]<='Z')
		{
			pass[i] = pass[i-1];
			pass[i].hv_A++;
		}
	}
	long long count=0;
```

```
        for (int i=1; i<=xau.length()-6+1; i++)
        {
                int VT = BSearch (i, i, n);
                if (VT!=-1)
                {
                        count+=n-VT+1;
                } else break;
        }
        cout<<count;
        return 0;
}
```

Bài 12. Đề bài QBSTR spoj – xâu con chung dài nhất

Xâu ký tự X được gọi là xâu con của xâu ký tự Y nếu ta có thể xoá đi một số ký tự trong xâu Y để được xâu X.

Cho biết hai xâu ký tự A và B, hãy tìm xâu ký tự C có độ dài lớn nhất và là con của cả A và B.

Input

Dòng 1: chứa xâu A

Dòng 2: chứa xâu B

Output

Chỉ gồm một dòng ghi độ dài xâu C tìm được

Ví dụ:

Input	Output
abc1def2ghi3 abcdefghi123	10

Hướng dẫn QBSTR spoj

Bài này phải sử dụng QHĐ.

+ Gọi F[i,j] là độ dài xâu con chung dài nhất tìm được khi xét i kí tự đầu tiên trong A và j kí tự đầu tiên trong B.

+ Nếu A[i]=B[j] thì F[i,j]:=F[i-1,j-1]+1.

+ ngược lại F[i,j]:=max(F[i-1,j],F[i,j-1]).

code tham khảo QBSTR spoj

```
program qbstr;
uses math;
const   fi='';
        nmax=1000;
var
        f:text;
        s1,s2:ansistring;
        KQ:array[0..nmax+10,0..nmax+10] of word;
```

```
procedure docfile;
begin
        assign(f,fi); reset(f);
        readln(f,s1);
        readln(f,s2);
        close(f);
end;

procedure bpa;
var     i,j:word;
begin
        for i:=0 to length(s1) do
                kq[i,0]:=0;

        for i:=0 to length(s2) do
                kq[0,i]:=0;

        for i:=1 to length(s1) do
                for j:=1 to length(s2) do
                        if s1[i]=s2[j] then
                                KQ[i,j]:=kq[i-1,j-1] + 1
                        else
                                KQ[i,j]:=max(kq[i,j-1],kq[i-1,j]);

        writeln(kq[length(s1),length(s2)]);
end;

begin
        docfile;
        bpa;
end.
```

Bài 14. Đề bài LINEGAME spoj – trò chơi với băng số

Trò chơi với băng số là trò chơi tham gia trúng thưởng được mô tả như sau: Có một băng hình chữ nhật được chia ra làm *n* ô vuông, đánh số từ trái qua phải bắt đầu từ 1. Trên ô vuông thứ *i* người ta ghi một số nguyên dương a_i, $i = 1, 2, \ldots, n$. Ở một lượt chơi, người tham gia trò chơi được quyền lựa chọn một số lượng tùy ý các ô trên băng số. Giả sử theo thứ tự từ trái qua phải, người chơi lựa chọn các ô $i_1, i_2, \ldots, i_k$. Khi đó điểm số mà người chơi đạt được sẽ là:

- $a_{i1} – a_{i2} + \ldots + (-1)^{k-1}a_{ik}$

Yêu cầu: Hãy tính số điểm lớn nhất có thể đạt được từ một lượt chơi.

Dữ liệu

- Dòng đầu tiên chứa số nguyên dương n ($n \leq 10^6$) là số lượng ô của băng số;
- Dòng thứ hai chứa n số nguyên dương $a_1, a_2, \ldots, a_n$ ($a_i \leq 10^4$, $i = 1, 2, \ldots, n$) ghi trên băng số. Các số liên tiếp trên cùng dòng được ghi cách nhau bởi ít nhất một dấu cách.

Kết quả

- Một số nguyên duy nhất là số điểm lớn nhất có thể đạt được từ một lượt chơi.

Ví dụ

Dữ liệu	Kết quả
7 4 9 2 4 1 3 7	17

Ràng buộc: 60% số tests ứng với 60% số điểm của bài có $1 \leq n \leq 20$.

```
#include<cstdio>
using namespace std;

typedef long long int ll;

int main()
{
   int n,tg;
   ll max=0,m1=0,m2=0,m3;
   scanf("%d",&n);
   for (int i=1;i<=n;i++)
   {
      scanf("%d",&tg);
      m3=m1;
      if(m1<m2+tg) m1=m2+tg;
      if(m2<m3-tg) m2=m3-tg;
   }
   if(m1>max) max=m1;
   if(m2>max) max=m2;
   printf("%lld",max);
}
```

Bài 15. Đề bài MAXARR1 spoj – help conan

Năm ngoái Conan chỉ mới bước vào học Tin học thật sự. Thế nhưng anh ta bị đàn em là Như Quỳnh thách đố bài toán sau:

Cho T ≤ 100000. Mỗi dòng của T có 1 số N (N ≤ 100000). Dãy số A được xây dựng như sau:

- A[0] = 0
- A[1] = 1

- A[2i] = A[i]
- A[2i+1] = A[i] + A[i+1]

Nhiệm vụ của bạn là tìm số lớn nhất của dãy A từ 1 với N.

Input

Dòng đầu tiên là số T.

T dòng sau, mỗi dòng là 1 số N.

Output

Có T dòng tương ứng với giá trị lớn nhất của các đoạn.

Ví dụ:

Input	Output
2 5 10	3 4

Thuật toán MAXARR1 spoj

Bài này không có gì để nói đến, dùng QHĐ bình thường thôi. Có thể vừa nhập vừa xử lí hoặc nhập xong rồi xử lí sau.

code tham khảo MAXARR1 spoj

```
#include <stdio.h>
#include <algorithm>
using namespace std;

int a[100002],amax[100002];

void xuli()
{
   a[0]=amax[0]=0;
   a[1]=amax[1]=1;
   int i;
   for (i=1; i<=50000; i++)
   {
      a[i*2]=a[i];
      amax[i*2]=max(a[i*2],amax[i*2-1]);
      a[2*i+1]=a[i]+a[i+1];
      amax[2*i+1]=max(a[i*2+1],amax[i*2]);
   }
}

int main()
{
   int test;
   xuli();
   scanf("%d",&test);
```

```
    int i,x;
    for (i=1; i<=test; i++ )
    {
       scanf("%d",&x);
       printf("%dn",amax[x]);
    }
    return 0;
}
```

Bài 16. Đề bài MTTRAVEL spoj – du lịch vòng quanh thế giới

Trên tuyến đường của xe chở khách du lịch vòng quanh thế giới xuất phát từ bến X có N khách sạn đánh số từ 1 đến N theo thứ tự xuất hiện trên tuyến đường, trong đó khách sạn N là địa điểm cuối cùng của hành trình mà tại đó tài xế bắt buộc phải dừng. Khách sạn i cách địa điểm xuất phát A_i Km (i=1, 2, …, N);
$A_1<A_2<…<A_N$.

Để đảm bảo sức khoẻ cho khách hàng, theo tính toán của các nhà chuyên môn, sau khi đã chạy được P (Km) xe nên dừng lại cho khách nghỉ ở khách sạn. Vì thế, nếu xe dừng lại cho khách nghỉ ở khách sạn sau khi đã đi được Q Km thì lái xe phải trả một lượng tiền phạt là : $(Q-P)^2$. Để đảm bảo lịch trình tài xế không được dừng khi chưa chạy đủ P Km và phải dừng tại một khách sạn nào đó.

Ví Dụ : Với N=4, P=300, A1=250, A2=310, A3=550, A4=590. Xe bắt buộc phải dừng lại ở khách sạn 4 là địa điểm cuối cùng của hành trình. Nếu trên đường đi lái xe chỉ dừng lại tại khách sạn thứ 2 thì lượng phạt phải trả là : $(310-300)^2+((590-310)-300)^2=500$

Yêu Cầu: Hãy xác định xem trên tuyến đường đến khách sạn N, xe cần dừng lại nghỉ ở những khách sạn nào để tổng lượng phạt mà lái xe phải trả là nhỏ nhất.

Dữ Liệu

Dòng đầu tiên chứa số nguyên dương N (N<=10000);
Dòng thứ hai chứa số nguyên dương P (P<=500);
Dòng thứ ba chứa các số nguyên dương A_1, A_2, A_3, …, An. (Ai<=2000000, i=1,2,..N)

Kết Quả

Dòng đầu tiên ghi Z là lượng phạt mà lái xe phải trả ;
Dòng thứ hai ghi K là số khách sạn mà lái xe cần dừng lại cho khách nghỉ;
Dòng thứ ba chỉ chứa chỉ số của K khách sạn mà xe dừng lại cho khách nghỉ.(Trong đó nhất thiết phải có khách sạn thứ N)

Ví Dụ

Input	Output
4 300 250 310 550 590	500 2 2 4

Hướng Dẫn MTTRAVEL spoj

- Hàm mục tiêu:

Gọi F[i] là lượng phạt ít nhất nếu người lái xe dừng lại địa điểm i.

- Bài toán cơ sở: F[0]=oo
- Công thức truy hồi:

F[i]:=Min{F[j]+sqr(A[i]-A[j]-P)}; với mọi j=0,..i-1

Code tham khảo MTTRAVEL spoj

```
#include<bits/stdc++.h>
using namespace std;
#define N 10005
#define ll long long
#define sqr(a) a*a
const ll oo=LLONG_MAX;
int n,a[N],p,tr[N],dem=0,kq[N];
ll d[N];
void vet(int u)
{
   while(u)
   {
      dem++;
      kq[dem]=u;
      u=tr[u];
   }
   cout<<dem<<"\n";
   for(int i=dem;i;i--) cout<<kq[i]<<" ";
}
int main()
{
  // freopen("TOURISM.INP","r",stdin);
   //freopen("TOURISM.OUT","w",stdout);
   ios_base::sync_with_stdio(0);cin.tie(0);
   cin>>n>>p;
   for(int i=1;i<=n;i++) cin>>a[i];
   for(int i=1;i<=n;i++)
   {
      ll dmin=oo;
      int jmin=0;
      for(int j=0;j<i;j++)
         if(dmin>=d[j]+(ll)(a[i]-a[j]-p)*(a[i]-a[j]-p))
         {
            dmin=d[j]+(ll)(a[i]-a[j]-p)*(a[i]-a[j]-p);
            jmin=j;
```

```
        }
      d[i]=dmin;
      tr[i]=jmin;
    }
    cout<<d[n]<<"\n";
    vet(n);
}
```

Bài 17. LCS2X - VOI 2014 - Dãy con chung bội hai dài nhất

Dãy C = c1, c2, .. ck được gọi là dãy con của dãy A = a1, a2, .., an nếu C có thể nhận được bằng cách xóa bớt một số phần tử của dãy A và giữ nguyên thứ tự của các phần tử còn lại, nghĩa là tìm được dãy các chỉ số $1 \le l1 < l2 < \ldots < lk \le n$ sao cho c1 = a_l1, c2 = a_l2, …, ck = a_lk. Ta gọi độ dài của dãy là số phần tử của dãy.
Cho hai dãy A = a1, a2, …, am và B = b1, b2, …, bn Dãy C = c1, c2, …, ck được gọi là dãy con chung bội hai của dãy A và B nếu C vừa là dãy con của dãy A, vừa là dãy con của dãy B và thỏa mãn điều kiện $2 \times ci \le c(i+1)$ (i = 1, 2, …, k – 1).

Yêu cầu

Cho hai dãy A và B. Hãy tìm độ dài dãy con chung bội hai có độ dài lớn nhất của hai dãy A và B.

Input

Dòng đầu tiên chứa T là số lượng bộ dữ liệu. Tiếp đến là T nhóm dòng, mỗi nhóm cho thông tin về một bộ dữ liệu theo khuôn dạng sau:

- Dòng đầu chứa 2 số nguyên dương m và n.
- Dòng thứ hai chứa m số nguyên không âm a1, a2, ..., am mỗi số không vượt quá 10^9.
- Dòng thứ ba chứa n số nguyên không âm b1, b2, ..., bn mỗi số không vượt quá 10^9.
- Các số trên cùng một dòng được ghi cách nhau ít nhất một dấu cách.

Giới hạn

- 30% số test có m, n <= 15.
- 30% số test khác có m, n <= 150.
- có 40% số test còn lại có m, n <= 1500.

Output

Ghi ra T dòng, mỗi dòng ghi một số nguyên là độ dài dãy con chung bội hai dài nhất của dãy A và B tương ứng với bộ dữ liệu vào.

Example

Input	Output
1 5 5 5 1 6 10 20 1 8 6 10 20	3

```
#include <bits/stdc++.h>

using namespace std;
   int m,n,a[1001],b[1001],f[1001][1001];
int main()
{
   cin>>m>>n;
   for(int i=1; i<=m; i++)
      cin>>a[i];
   for(int j=1; j<=n; j++)
      cin>>b[j];
   f[0][0]=0;
   for(int i=1; i<=m; i++)
      f[i][1]=(a[i]==b[1])?1:f[i-1][1];
   for(int j=1; j<=n; j++)
      f[1][j]=(a[1]==b[j])?1:f[1][j-1];
   for(int i=2; i<=m; i++)
      for(int j=2; j<=n; j++)
   {
      f[i][j]=max(f[i][j-1],max(f[i-1][j],f[i-1][j-1]));
      if(a[i]==b[j])f[i][j]=max(f[i][j],f[i-2][j-2]+1);
   }
   cout<<f[m][n];
}
```

Bài 18. Đề bài NKPALIN spoj – chuỗi đối xứng

`Một chuỗi được gọi là đối xứng (palindrome) nếu như khi đọc chuỗi này từ phải sang trái cũng thu được chuỗi ban đầu.

Yêu cầu: tìm một chuỗi con đối xứng dài nhất của một chuỗi s cho trước. Chuỗi con là chuỗi thu được khi xóa đi một số ký tự từ chuỗi ban đầu.

Dữ liệu vào

Gồm một dòng duy nhất chứa chuỗi s, chỉ gồm những chữ cái in thường.

Kết qủa

Gồm một dòng duy nhất là một xâu con đối xứng dài nhất của xâu s. Nếu có nhiều kết quả, chỉ cần in ra một kết quả bất kỳ.

Giới hạn

Chuỗi s có độ dài không vượt quá 2000.

Ví dụ

Input	Output
lmevxeyzl	level

Hướng dẫn NKPALIN spoj

Gọi F[i,j] là độ dài xâu đối xứng dài nhất thu được trong xâu S[1..i] và S[j..n] (i=1..n; j=n..1) n là độ dài xâu S,

Nghiệm bài toán F[n,1]

Code tham khảo NKPALIN spoj

```
program bt;
uses    math;
const   fi='';
        nmax=2000;
var
        f:text;
        S:ansistring;
        L:array[0..nmax+1,0..nmax+1] of word;
        n:word;

procedure docfile;
begin
        assign(f,fi); reset(f);
        readln(f,s);
        close(f);
end;

procedure bpa;
var     i,j:word;
        kq:ansistring;
begin
        n:=length(s);
        for i:=0 to n do
                begin
                        L[i,n+1]:=0;
                        L[0,i]:=0;
                end;
        for i:=1 to n do
                for j:=n downto 1 do
                        if s[i]=s[j] then
                                L[i,j]:=L[i-1,j+1] + 1
                        else
                                L[i,j]:=max(L[i-1,j],L[i,j+1]);
        i:=n; j:=1;
        kq:='';
        while (i>0) and (j<n+1) do
                if s[i]=s[j] then
```

```
                begin
                    kq:=kq+s[i];
                    dec(i); inc(j);
                end
            else
                if l[i,j]=l[i-1,j] then
                    dec(i)
                else
                    inc(j);
        writeln(kq);
    end;

    begin
        docfile;
        bpa;
    end.
```

Bài 19. Đề bài LATGACH spoj – lát gạch

Cho một hình chữ nhật kích thước 2xN (1<=N<=100). Hãy đếm số cách lát các viên gạch nhỏ kích thước 1×2 và 2×1 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	Output
3 1 2 3	1 2 3

Hướng dẫn LATGACH spoj

thực chất đây là bây toán Fibonacci. F[i]:=F[i-1]+F[i-2]. việc còn lại là xử lí số lớn.

Code tham khảo LATGACH spoj

```
    const   fi='';
    var
        f:text;
        T:byte;
        A:array[0..100] of ansistring;
```

```
function cong(a,b:ansistring):ansistring;
var     c:ansistring;
        i,du,carry:word;
begin
        while length(a)<length(b) do
                a:='0'+a;
        while length(b)<length(a) do
                b:='0'+b;

        du:=0; c:='';
        for i:=length(a) downto 1 do
                begin
                        carry:=ord(a[i])+ord(b[i])-48*2+du;
                        if carry>9 then
                                begin
                                        du:=1;
                                        carry:=carry-10;
                                end
                        else
                                du:=0;
                        c:=chr(carry+48) + c;
                end;
        if du=1 then c:='1'+c;
        exit(c);
end;

procedure fibo;
var     i:byte;
begin
        a[0]:='1';
        a[1]:='1';
        for i:=2 to 100 do
                a[i]:=cong(a[i-1],a[i-2]);
end;

begin
        assign(f,fi); reset(f);
        readln(f,t);
        fibo;
        while not eof(f) do
                begin
                        readln(f,t);
```

```
            writeln(a[t]);
        end;
    close(f);
end.
```

Bài 20. Đề bài LATGACH4 spoj – lat gach 4

Cho một hình chữ nhật kích thước 2xN (1<=N<10^9). Hãy đếm số cách lát các viên gạch nhỏ kích thước 1×2 và 2×1 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 lấy phần dư cho 111539786.

Example

Input	Output
3 1 2 3	1 2 3

Code tham khảo LATGACH4 spoj

```
const   fi='';
    sodu = 111539786;
    nmax = 1000000;

var     f:text;
    A:array[0..nmax] of longint;
    T,ti:word;
    i,tam:longint;

function fibo(n:int64):int64;
var     k1,k2:int64;
begin
    if n<=nmax then
        exit(a[n] mod sodu);

    if n mod 2 = 0 then
        begin
            k1:=fibo(n div 2) mod sodu;
            k2:=fibo(n div 2 - 1) mod sodu;
            fibo:=(((k1*k1) mod sodu )+ ((k2*k2) mod sodu)) mod sodu;
        end
```

```
        else
            fibo:= (fibo(n div 2)*((fibo(n div 2 + 1) + fibo(n div 2 - 1)) mod
    sodu)) mod sodu;
    end;

    begin
        A[1]:=1;
        a[2]:=2;
        for i:=3 to nmax do
            a[i]:=(a[i-1]+a[i-2]) mod sodu;

        assign(f,fi); reset(f);
        readln(f,t);
        for ti:=1 to t do
            begin
                readln(f,tam);
                writeln(fibo(tam));
            end;
        close(f);
    end.
```

Bài 21. Đề bài YB_KT2B2 spoj – phần thưởng 2

TIMBERSAW oai hùng ngày nào nay đã trở thành bác bảo vệ của rừng Radiant (có dạng hình chữ nhật kích thước MxN). Để đền đáp công lao to lớn của TIMBERSAW, Chúa đất cho phép anh chọn một khu rừng hình chữ nhật có kích thước k*k, có các cạnh song song với các bìa rừng và Chúa đất chỉ cho phép anh khai thác cây trên đường biên của khu rừng đó. Biết giá trị của mỗi cây trong khu rừng Radiant là a[i,j] nguyên.

Bạn hãy giúp TIMBERSAW chọn một khu rừng có giá trị cao nhất mà vẫn thỏa mãn yêu cầu của Chúa đất.

Input

- Dòng 1: Gồm 3 số m, n , k ($0< k \leq \min(m,n)$; $m,n \leq 1000$).
- m dòng sau, mỗi dòng n số nguyên là giá trị của mỗi cây trong khu rừng.

Output

- Một số duy nhất là giá trị cao nhất mà TIMBERSAW có thể nhận được.

Example

Input	Output
4 4 3 3 4 5 -5 -7 0 7 2 3 -7 6 -4 1 -6 -3 -5	14

Hướng dẫn giải YB_KT2B2 spoj

Gợi ý: cách làm tương tự như bài BONUS

Code:

```
const   fi='';
        nmax=1000;
type    data=longint;
var
        f:text;
        A:array[0..nmax+1, 0..nmax+1] of int64;
        n,m,k:data;

procedure docfile;
var     i,j:data;
begin
        assign(f,fi); reset(f);
        readln(f,m,n,k);
        for i:=0 to m+1 do
                a[i,0]:=0;
        for j:=0 to n+1 do
                a[0,j]:=0;

        for i:=1 to m do
                for j:=1 to n do
                        begin
                                read(f,a[i,j]);
                                a[i,j]:=a[i-1,j]+a[i,j-1]-a[i-1,j-1]+a[i,j];
                        end;
        close(f);
end;


function max(a,b:int64):int64;
begin
        if a>b then exit(a);
        exit(b);
end;

procedure QHD;
var     i,j,bt:data;
        res:int64;
begin
        res:=low(int64);
        for i:=k to m do
```

```
            for j:=k to n do
                begin
                    bt:=A[i,j]-a[i-k,j]-a[i,j-k]+a[i-k,j-k];
                    if k>2 then
                    bt:=bt-(A[i-1,j-1]-a[i-k+1,j-1]-a[i-1,j-k+1]+a[i-k+1,j-k+1]);
                    res:=max(res,bt);
                end;
        writeln(res);
end;

begin
        docfile;
        QHD;
end.
```

Bài 22. BCCHIANHOM - Chia nhóm

Cho dãy số A gồm N số (N <= 10) a1, a2, .., an. và một số nguyên dương K(1 < K < N). Hãy đưa ra số cách để chặt dãy số thành K nhóm (các phần tử trong nhóm là liên tiếp) mà các nhóm có tổng bằng nhau.

Input

Dòng đầu tiên bao gồm 2 số nguyên n và k (0 < n <= 12, 0 < k <= n)
Dòng tiếp theo bao gồm n số nguyên a1, a2, …, an (-10000 <= ai <= 10000)

Output

In ra số cách thỏa mãn.

Ví dụ:

Input	Output
3 2 -2 0 -2	2
3 2 1 2 3	1

Hướng dẫn:

- Sử dụng QHĐ cơ bản: lưu tổng giá trị của mảng trước đó tại mỗi phần tử (arr[i]=arr[i-1]+value) VD: i:1-3 {-2 0 -2} -> arr[]={0 -2 -2 -4} (i:0->3) - Đệ quy - quay lui: + Vì dãy chặt tạo thành các nhóm và tổng số phần tử các nhóm luôn = n -> Nên với mỗi arr[i] sẽ là trường hợp tổng giả sử có thể chặt được. và ta có vt=i; + Với mỗi tổng giả sử có được, ta lặp lại hành động với dãy arr[vt+1->n]; (p/s tổng có từ quy hoạch trên ta có SUM_[i->j] = arr[j]-arr[i-1]).

Code:

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

```
int n, k;
int arr[20];
void read ()
{
	cin>>n>>k;
	arr[0]=0;
	for (int i=1; i<=n; i++)
	{
		int tmp;
		cin>>tmp;
		arr[i]=arr[i-1]+tmp;
	}
}

int count=0;
int tim (int x, int bg, int ed, int K)
{
	if (bg > ed) return 0;
	if (K==1)
	{
		if (x==arr[ed]-arr[bg-1])
		{
			count++;
			return 1;
		}
		else return 0;
	}
	else if (K==0) return 0;
	else
	{
		int BG = bg;
		for (int i=bg; i<=ed; i++)
		{
			if (arr[i]-arr[BG-1]==x)
			{
				tim (x, i+1, ed, K-1);
			}
		}
	}
}

int main ()
```

```
{
        read ();
        if (k==1)
        {
                count=1;
        }
        else
        {
                for (int i=1; i<=n; i++)
                {
                        int label = arr[i];
                        tim (label, i+1, n, k-1);
                }
        }
        cout<<count;
        return 0;
}
```

Bài 23. BCPARTI - Partition thuận nghịch đệ quy

Một biểu diễn của một số nguyên dương N dưới dạng tổng các số nguyên dương nhỏ hơn hoặc bằng N được gọi là partition của số N. Ví dụ với N =15 ta có thể biểu diễn: 15 = 1+2+3+4+5 = 1+2+1+7+1+2+1.

Một partition được gọi là thuận nghịch nếu đọc theo cả hai chiều đều được kết quả giống nhau. Ví dụ {1+2+1+7+1+2+1} là một partition của 15 thoả mãn tính chất thuận nghịch.

Một partition được gọi là thuận nghịch đệ quy nếu nửa bên trái của nó cũng thuận nghịch đệ quy hoặc rỗng. Với định nghĩa này, mỗi số N sẽ hiển nhiên sẽ có hai partition thuận nghịch đệ quy là dãy N số 1 và dãy chỉ gồm duy nhất số N.

Ví dụ, các partition thuận nghịch đệ quy của 7 là:

7, 1+5+1, 2+3+2, 1+1+3+1+1, 3+1+3, 1+1+1+1+1+1+1

Viết chương trình nhập vào số tự nhiên N và đưa ra số partition thuận nghịch đệ quy của N.

Dữ liệu vào

Dòng đầu tiên chứa một số nguyên $1 \le t \le 1000$ là số lượng bộ test. Mỗi bộ test sẽ viết trên một dòng duy nhất một số nguyên N(N<=1000)

Dữ liệu ra

Với mỗi bộ dữ liệu vào, đưa ra một dòng gồm 2 số: số thứ tự bộ test và số lượng partition thuận nghịch đệ quy tương ứng. Hai số cách nhau bởi một dấu cách.

Input	Output
3 4 7 20	1 4 2 6 3 60

Xét dễ hiểu thì thế này:

4 có 2 2 là thuận nghich đề quy nhưng con 2 có hai thuận ngịch đệ quy -> khi 2 phân tích thì 4 sẽ có thêm 2 thuận nghịch đệ quy nữa.

Vậy với [1] có 1 thuận nghịch đệ quy và [2] có 2 thuận ngịch đệ quy thì có thể suy ra các số tiếp theo.

Code:

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

int main ()
{
int arr[1003];
arr[1]=1;
arr[2]=2;
int t;
cin>>t;
int n;
for (int i=1; i<=t; i++)
{
cin>>n;
for (int j=3; j<=n; j++)
{
int S=1;
for (int k=1; k<=j/2; k++)
{
S+=arr[k];
}
arr[j]=S;
}
cout<<i<<" "<<arr[n]<<endl;
}
```

```
return 0;
}
```

Bài 24. Hình chữ nhật lớn nhất

Cho một bảng kích thước MxN, được chia thành lưới ô vuông đơn vị M dòng N cột (1 <= M, N <= 1000)
Trên các ô của bảng ghi số 0 hoặc 1. Các dòng của bảng được đánh số 1, 2… M theo thứ tự từ trên xuống dưới và các cột của bảng được đánh số 1, 2…, N theo thứ tự từ trái qua phải
Yêu cầu:
Hãy tìm một hình chữ nhật gồm các ô của bảng thoả mãn các điều kiện sau:
1 – Hình chữ nhật đó chỉ gồm các số 1
2 – Cạnh hình chữ nhật song song với cạnh bảng
3 – Diện tích hình chữ nhật là lớn nhất có thể
Input
Dòng 1: Ghi hai số M, N
M dòng tiếp theo, dòng thứ i ghi N số mà số thứ j là số ghi trên ô (i, j) của bảng
Output
Gồm 1 dòng duy nhất ghi diện tích của hình chữ nhật tìm được
Example
Input:
11 13
0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 1 1 1 1 1 1 1 1 1 0 0 0
1 1 1 1 1 1 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 1 1
0 0 0 0 0 1 0 0 0 0 0 1 1

Output:
49
THUẬT TOÁN :
Gọi h[j] là chiều cao của hình chữ nhật tính từ hàng i đang xét đi lên tại cột j chứa liên tiếp các số 1
Gọi l[j] là vị trí trái nhất của cột j có h[j] -> h[l[j]] lớn hơn hoặc bằng h[j]
Gọi r[j] là vị trí phải nhất của cột j có h[j] -> h[r[j]] lớn hơn hoặc bằng h[j]

Vậy khi đó tồn tại một hình chữ nhật có chiều cao là h[j] và chiều dài max là r[j]-l[j]+1
Duyệt từng hàng :
+ khởi tạo h
+ tạo l và r bằng QHĐ
+ tìm max cho res
Cách tạo l – r :
+ Đối với l thì duyệt từ 1 -> n còn r thì ngược lại
+ Tại i thì ban đầu l[i] = i. Xét các vị trí trước nó, bắt đầu là i-1. Nếu như h[l[i]-1] >= h[i], thỏa điều ta cần thì gán l[i] = l[l[i]-1]] vì trước đó đã tính l[l[i]-1] rồi. Tương tự với r thôi
CODE :

```
#include <iostream>
#include <fstream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <algorithm>
 using namespace std;
 #define maxn        1010

int     n , m , a[maxn][maxn];
int     h[maxn] , l[maxn] , r[maxn] , res;

int main()
{
    scanf("%d%d",&m,&n);
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++) scanf("%d",&a[i][j]);
    h[0] = -1; h[n+1] = -1;
    for (int i=1;i<=m;i++) {
        for (int j=1;j<=n;j++)
            h[j] = a[i][j] * (h[j] + 1);
        for (int j=1;j<=n;j++) {
            l[j] = j;
            while (h[l[j]-1] >= h[j]) l[j] = l[l[j]-1];
        }
        for (int j=n;j>0;j--) {
            r[j] = j;
            while (h[r[j]+1] >= h[j]) r[j] = r[r[j]+1];
```

```
        }
        for (int j=1;j<=n;j++)
            res = max(res,h[j]*(r[j]-l[j]+1));
    }
    printf("%d",res);
}
```