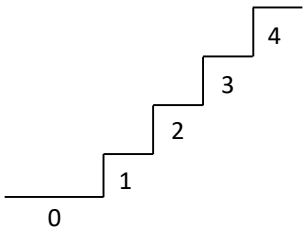
MỤC LỤC

Bài 1: Cầu thang 1 **Tên chương trình: GOSTAIRS.***

Một cầu thang có n bậc thang được đánh số từ 1 đến n. Một người đứng ở bậc thứ 0 (ở nền nhà) muốn lên đến bậc thứ n mà chỉ có thể bước với số bước là 1 bậc hoặc 2 bậc.



(ví dụ n=4)

Yêu cầu: Hãy cho biết có bao nhiêu cách đi từ bậc 0 đến bậc thứ n.

Dữ liệu vào từ file 'GOSTAIRS.INP' chứa số nguyên dương n (n≤50).

Kết quả ghi vào file 'GOSTAIRS.OUT' các kết quả tìm được.

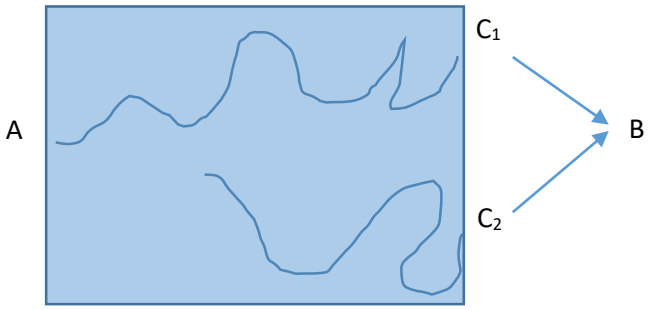
Ví dụ:

GOSTAIRS.INP
8

GOSTAIRS.OUT
34

HƯỚNG DẪN:

Nói về quy tắc cộng (cơ bản):



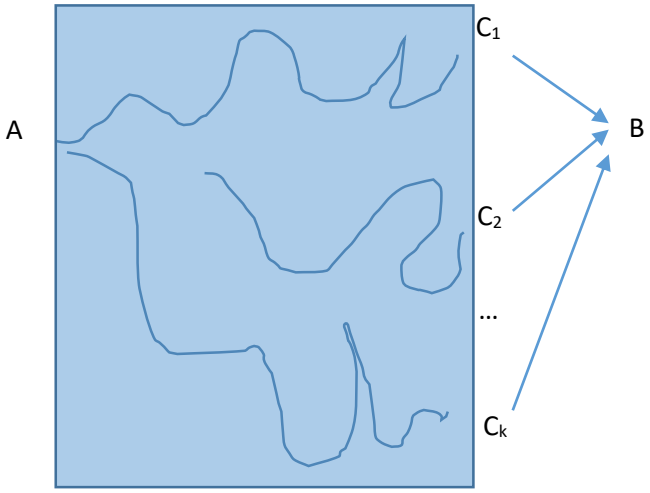
Gọi ***d[B] là số cách đi từ A đến B***

d[C_1] là số cách đi từ A đến C_1

d[C_2] là số cách đi từ A đến C_2

➔ $d[B]=d[C_1]+d[C_2]$

Mở rộng:



Gọi ***d[B] là số cách đi từ A đến B***

d[C_1] là số cách đi từ A đến C_1

d[C_2] là số cách đi từ A đến C_2

...

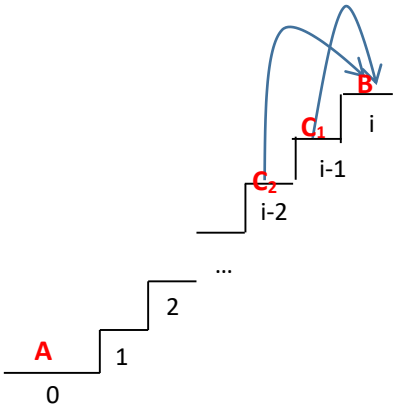
d[C_k] là số cách đi từ A đến C_k

➔ $d[B]=d[C_1]+d[C_2]+...+d[C_k]$

❖ Hướng dẫn bài GOSTAIRS

Gọi d[i] là số cách đi từ bậc 0 (nền nhà) đến bậc thứ i

Hay d[i] là số cách đi được i bậc.



*Như vậy ta thấy: để đi **đến bậc thứ i** ta chỉ có thể **đi từ bậc thứ i-1** hoặc **từ bậc thứ i-2.***

(Ta có thể xem như bậc 0 (nền nhà) là điểm A, bậc thứ i-1 là điểm C_1, bậc thứ i-2 là điểm C_2, bậc thứ i là điểm B)

Ta có: số cách đi đến bậc thứ i-1 là d[i-1]

và số cách đi đến bậc thứ i-2 là d[i-2]

Theo quy tắc cộng: số cách đi đến bậc thứ i là: d[i]=d[i-1]+d[i-2]

❖ Cài đặt:

```
d[1]=1;
d[2]=2;
for(i=3;i<=n;i++)
    d[i]=d[i-1]+d[i-2];
```

Đáp án là d[n]

Bài 2: Đường đi trên lưới **Tên chương trình: ROADSQUA.***

Cho một lưới ô vuông đơn vị kích thước n*m. Ô giao với hàng i và cột j được gọi là ô (i, j). Một bước đi là sự di chuyển từ 1 ô sang ô kề cạnh ở bên phải hay ở bên dưới ô đó.

Yêu cầu: Cho biết số cách đi từ ô (1, 1) đến ô (n, m).

Dữ liệu vào từ file 'ROADSQUA.INP' ghi 2 số nguyên dương n và m ($1 \le n, m \le 100$)

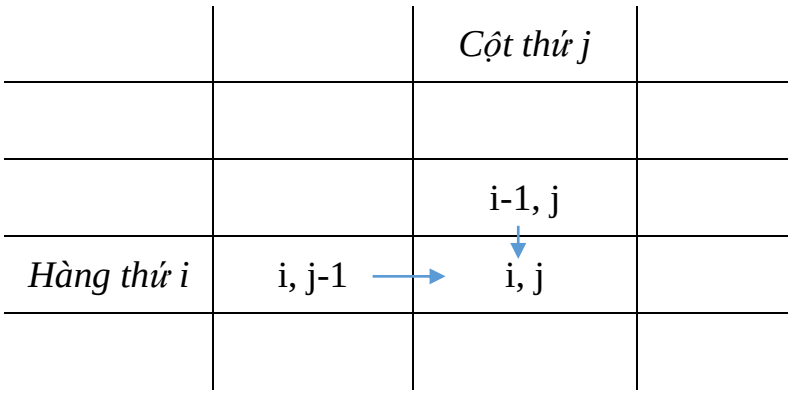
Kết quả ghi vào file 'ROADSQUA.OUT' kết quả tìm được.

Ví dụ:

ROADSQUA.INP
2 3

ROADSQUA.OUT
3

❖ **Hướng dẫn bài ROADSQUA**



Gọi d[i,j] là số cách đi từ ô (1,1) đến ô (i,j)

Để đến được ô (i,j) thì ta chỉ có thể từ ô (i-1,j) và ô (i,j-1) đến.

Áp dụng quy tắc cộng:

➔ *d[i,j]=d[i-1,j]+d[i,j-1]*

❖ **Cài đặt:**

Khởi gán hàng 1 và cột 1 của mảng d

```
for(j=1; j<=m; j++)
        d[1][j]=1;
for(i=1; i<=n; i++)
        d[i][1]=1;
```

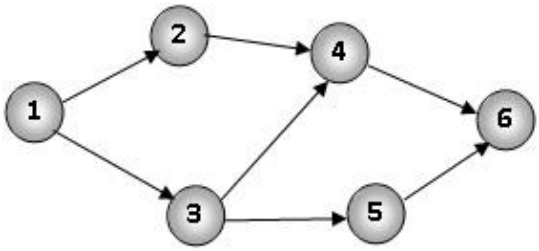
Tính tiếp mảng d

```
for(i=2; i<=n; i++)
        for(j=2; j<=m; j++)
                d[i][j]=d[i-1][j]+d[i][j-1];
```

Đáp án là: d[n,m]

Bài 3: Những con đường **Tên chương trình: WAYSHUGO.***

Trong chúng ta, hầu hết ai cũng biết hoặc nghe qua trò chơi Hugo. Có một lần, Hugo bị mụ phù thủy Sylar truy đuổi. Trước mặt Hugo là một mê cung, Hugo muốn thoát được thì anh ta phải vượt qua được mê cung này. Mê cung này có thể xem như một hệ thống gồm n địa điểm được đánh số từ 1 đến n, trong đó gồm toàn các con đường một chiều (nếu Hugo đi ngược chiều thì sẽ bị mụ phù thủy Sylar phát hiện). Biết rằng các con đường một chiều nối từ địa điểm i đến địa điểm j (có hướng từ i đến j) phải thỏa điều kiện i<j. Như vậy, địa điểm 1 sẽ không có bất cứ con đường nào có hướng về nó và địa điểm n sẽ không có bất cứ con đường nào hướng ra từ nó. Hugo đang đứng ở địa điểm 1 và muốn đến địa điểm thứ n.



Từ địa điểm 1 đi đến địa điểm 6 có 3 cách đi:
+ Cách 1: 1 → 2 → 4 → 6
+ Cách 2: 1 → 3 → 4 → 6
+ Cách 3: 1 → 3 → 5 → 6

Yêu cầu: Hãy cho biết có bao nhiêu con đường khác nhau để Hugo có thể đi từ địa điểm 1 đến địa điểm thứ n.

Dữ liệu vào từ file 'WAYSHUGO.INP'

- Dòng đầu chứa số nguyên dương n.
- Các dòng tiếp theo, mỗi dòng chứa hai số i và j cho biết có đường đi một chiều trực tiếp từ i đến j.

Kết quả ghi vào file 'WAYSHUGO.OUT' ghi số lượng con đường đi từ địa điểm 1 đến địa điểm thứ n.

Ví dụ:

WAYSHUGO.INP
6 1 2 1 3 2 4 3 4 3 5 4 6 5 6

WAYSHUGO.OUT
3

Giới hạn kỹ thuật:

- n≤10000
- Có 60% test với n≤20.

❖ **Hướng dẫn bài WAYSHUGO**

Quy ước: a[j][i]=1 → có con đường trực tiếp từ j đến i

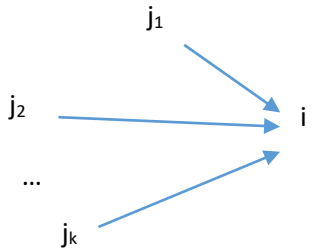
***Đọc file:**

```
while (cin >>j >>i)
        a[j][i]=1;
```

***Thuật toán:**

Gọi d[i] là số cách đi từ địa điểm 1 đến địa điểm i.

Giả sử để đến được địa điểm i thì chỉ có thể từ địa điểm $j_1, j_2, ..., j_k$ đi đến nó (như hình dưới)



Để tính d[i] thì ta thấy:

- *$d[j_1]$ chính là số cách đi từ điểm 1 đến điểm j_1*
- *$d[j_2]$ chính là số cách đi từ điểm 1 đến điểm j_2*
- ...
- *$d[j_k]$ chính là số cách đi từ điểm 1 đến điểm j_k*

Như vậy (theo quy tắc cộng): $d[i] = d[j_1] + d[j_2] + ... + d[j_k]$

Từ đó, dễ thấy hệ thức tổng quát để tính d[i] như sau:

$$d[i] = \sum_{j=1}^{i-1} d[j]$$

với a[j][i]=1

❖ **Cài đặt:**

```
d[1]=1;
for(i=2; i<=n; i++)
        {       d[i]=0;
                for(j=1; j<i; j++)
                        d[i]+=d[j];
        }
```

Đáp án là: d[n]

Bài 4: Cái túi thần kỳ **Tên chương trình: MAGICBAG.***

Sau nhiều lần vào hang động và lấy đi những món đồ của bọn cướp phân phát cho người dân, Một lần nọ Alibaba đã bị bọn cướp bắt. Biết được Alibaba là người tốt, tên tướng cướp quyết định thả cho Alibaba ra về và đồng thời cũng cho mang theo một số cổ vật trong hang động đi. Trong hang động có n cổ vật mà tên tướng cướp muốn cho Alibaba, các cổ vật được xếp trên một hàng (có thể xem như một đường thẳng) và được đánh số thứ tự lần lượt từ 1 đến n, cổ vật thứ i có giá trị là a_i (là một số nguyên; i=1, 2,…n). Tên tướng cướp chỉ cho phép Alibaba được mang đi các cổ vật mà không được chọn 2 cổ vật đứng kế nhau. Alibaba sau một hồi suy nghĩ, trong tay lại có chiếc túi thần kỳ nên Alibaba thoải mái chọn các cổ vật bỏ vào cái túi thần kỳ mà không sợ không mang đi được. Tất nhiên việc lấy các cổ vật cũng phải tuân thủ nguyên tắc của tên tướng cướp đưa ra.

Yêu cầu: Hãy cho biết tổng giá trị tối đa của các cổ vật mà Alibaba mang đi là bao nhiêu?

Ví dụ: *có 4 cổ vật với trị giá tương ứng là: 6, 2, 1, 5*

➔ *Chọn được các cổ vật có giá trị tương ứng là 6 5 thỏa mãn điều kiện có tổng giá trị lớn nhất là 11*

Dữ liệu vào từ file MAGICBAG.INP:

- Dòng đầu tiên chứa số nguyên dương n ($n \le 10^5$)
- n dòng tiếp theo, dòng thứ i chứa số nguyên dương a_i ($a_i \le 10^9$)

Kết quả ghi vào file MAGICBAG.OUT tổng giá trị lớn nhất của các cổ vật thỏa mãn yêu cầu.

Ví dụ:

MAGICBAG.INP
4 6 2 1 5

MAGICBAG.OUT
11

❖ Hướng dẫn bài MAGICBAG

a_1 a_2 a_3 a_4 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i … a_{n-2} a_{n-1} a_n

Gọi t[i] là tổng lớn nhất khi ta đã chọn trong dãy xét từ vị trí 1 đến vị trí i (thoả đk) mà ***phần tử a[i] bắt buộc được chọn***

t[1]=a[1]

t[2]=a[2]

t[3]=a[1]+a[3]

Tính t[i]?

- Trường hợp 1: ***không chọn a[i-1]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= **t[i-2]** + a[i]

- Trường hợp 2: ***không chọn a[i-2] a[i-1]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= **t[i-3]** + a[i]

- Trường hợp 3: ***không chọn a[i-3] a[i-2] a[i-1]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= **t[i-4]** + a[i]

  TH này không khả thi ➔ bỏ vì các giá trị là dương (việc bỏ 3 phần tử chắc chắn sẽ không mang lại hiệu quả!)

→ Hệ thức: *t[i]=**max(**t[i-2],t[i-3]**)**+a[i]*

❖ Cài đặt:

```
t[1]=a[1];
t[2]=a[2];
t[3]=a[1]+a[3];
for(i=4; i<=n; i++)
    t[i]=max(t[i-2],t[i-3])+a[i];
```

Đáp án là: *max(t[n],t[n-1])*

Bài 5: Các món quà của bà **Tên chương trình: GIFTS.***

Nhân dịp về quê ăn tết, Nam rất vui mừng gặp được ông bà và người thân của em. Bà của Nam rất yêu quý Nam và biết được Nam rất giỏi trong việc tính toán cũng như rất giỏi bộ môn Tin học nên bà của Nam đã đưa ra một hình thức rất thú vị để thưởng cho thành tích học tập của Nam và đó cũng là phần lì xì tết cho Nam.

Bà sắp ra cho Nam một dãy n món quà, được xếp trên một đường thẳng, được đánh số lần lượt từ 1 đến n. Món thứ i có giá trị v_i (i=1, 2,…n). Nam được chọn các món quà đó sao thỏa mãn cả 2 điều kiện sau:

- Không được chọn ba món quà liên tiếp.
- Trong ba món quà liên tiếp phải chọn ít nhất một món.

Yêu cầu: Hãy cho biết với các yêu cầu trên thì tổng giá trị các món quà Nam nhận tối đa được là bao nhiêu?

Ví dụ: *có 6 món quà với trị giá tương ứng là: 2, 6, 5, 1, 7, 3*

➔ *Ta chọn được các món quà có giá trị tương ứng là 6, 5, 7, 3 thỏa mãn tất cả các điều kiện trên có tổng giá trị lớn nhất là 21*

Dữ liệu vào từ file GIFTS.INP:

- Dòng đầu tiên chứa số nguyên dương n ($n \leq 1000$)
- n dòng tiếp theo, dòng thứ i chứa số nguyên dương v_i ($v_i \leq 30000$)

Kết quả ghi vào file GIFTS.OUT tổng giá trị lớn nhất thỏa mãn yêu cầu.

Ví dụ:

GIFTS.INP
6 2 6 5 1 7 3

GIFTS.OUT
21

❖ Hướng dẫn bài GIFTS

a_1 a_2 a_3 a_4 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i … a_{n-2} a_{n-1} a_n

Gọi t[i] là tổng lớn nhất khi ta đã chọn trong dãy xét từ vị trí 1 đến vị trí i (thoả đk) mà ***phần tử a[i] bắt buộc được chọn***

t[1]=a[1]

t[2]=a[1]+a[2]

*t[3]=**max(**a[1]+a[3] **,** a[2]+a[3]**)***

*t[4]=**max(**a[1]+a[2]+a[4] **,** a[1]+a[3]+a[4]**)***

Tính t[i]?

- Trường hợp 1: ***không chọn a[i-1]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= t[i-2] + a[i]

- Trường hợp 2: ***không chọn a[i-2] a[i-1]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= t[i-3] + a[i]

- Trường hợp 3: ***không chọn a[i-2]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= t[i-3] + a[i-1]+a[i]

- Trường hợp 4: ***không chọn a[i-3] a[i-2]***

  a_1 a_2 … a_{i-4} a_{i-3} a_{i-2} a_{i-1} a_i

  t[i]= t[i-4] + a[i-1]+a[i]

Ta thấy ở TH 2 và TH 3 thì TH 3 chắc chắn tốt hơn TH 2 nên ta bỏ TH 2

→ Hệ thức: t[i]=**max(** t[i-2] +a[i] **,** t[i-3]+a[i-1]+a[i] **,** t[i-4]+a[i-1]+a[i] **)**

❖ Cài đặt:

```
t[1]=a[1];
t[2]=a[1]+a[2];
t[3]=max(a[1]+a[3] , a[2]+a[3]);
t[4]=max(a[1]+a[2]+a[4] , a[1]+a[3]+a[4]);
for(i=5; i<=n; i++)
    t[i]=max(t[i-2]+a[i], max(t[i-3]+a[i-1]+a[i] , t[i-4] + a[i-1]+a[i]));
```

Đáp án là: *max(t[n],t[n-1])*

Bài 6: Dãy con **Tên chương trình: INCSUBSE.***

Cho một dãy số gồm n phần tử a_1, a_2…, a_n.

Yêu cầu: hãy tìm một dãy con tăng có nhiều phần tử nhất.

Dữ liệu vào từ file INCSUBSE.INP gồm

- Dòng đầu là n ($n \leq 7000$)
- Dòng thứ hai là các phần tử của dãy ($-32000 \leq a_i \leq 32000$)

Kết quả ghi vào file INCSUBSE.OUT'số lượng phần tử dãy con tăng nhiều nhất.

Ví dụ:

INCSUBSE.INP
12 6 12 8 11 3 4 1 7 5 9 10 2

INCSUBSE.OUT
5

❖ Hướng dẫn bài INCSUBSE

Dãy con???

Cho dãy ban đầu: $a_1, a_2, \dots, a_n$

Dãy con được khái niệm như sau:

- Xoá đi 1 hoặc 1 số phần tử trong dãy --> những pt còn lại tạo thành 1 dãy --> gọi là dãy con của dãy ban đầu!
- Không xoá pt nào cả! --> Nó cũng là con của nó!
- Xoá hết! --> Dãy rỗng (không có pt nào) --> Dãy này là con của mọi dãy!

Ví dụ: 3 7 2 9

--> Dãy con:

{}

{3} {7} {2} {9}

{3 7} {3 2} {3 9} {7 2} {7 9} {2 9}

{3 7 2} {3 7 9} {3 2 9} {7 2 9}

{3 7 2 9}

Những khái niệm:

Dãy tăng < < < ...

Dãy không giảm ≤ ≤ ≤ ...

Dãy giảm > > > ...

Dãy không tăng ≥ ≥ ≥ ...

Thuật toán bài INCSUBSE:

Gọi ***d[i]*** là số lượng phần tử của dãy con tăng dài nhất (nhiều phần tử nhất) khi ta xét dãy chỉ bao gồm i phần tử đầu tiên ***mà chắc chắn a[i] phải được chọn*** vào trong dãy con tăng dài nhất đó!

a[1] a[2] a[3] .... a[i-1] a[i]

i	1	2	3	4	5	6	7	8	9	10	11	12
a[i]	6	12	8	11	3	4	1	7	5	9	10	2
d[i]	1	2	2	3	1	2	1	3	3	4	4+1=5	2

Hệ thức:

d[1]=1;

d[i]=max(d[j])+1 với j=1, 2,.. i-1 và a[j] <=a[i]

❖ **Cài đặt:**

```
d[1]=1;
for(i=2; i<=n; i++)
{
    //Tìm giá trị lớn nhất của d[j]
    maxdj=0;
    for(j=1; j<=i-1; j++)
        if((d[j]>maxdj) && (a[j]<=a[i]))
            maxdj=d[j];
    d[i]=maxdj+1;
}
//Tìm giá trị lớn nhất của mảng d --> đó đáp án
maxd=d[1];
for(i=2; i<=n; i++)
    maxd=max(maxd,d[i]);
```

Đáp án là: maxd

Bài 7: Hàng cây 2 **Tên chương trình: LINETREEK.***

Ở một ngôi làng nọ, trên một con đường (được xem như một đường thẳng), có n cây gỗ quý được đánh số theo thứ tự lần lượt từ 1 đến n và có giá trị lần lượt là a_1, a_2,… a_n. Sau khi tính toán, trưởng làng đã quyết định khai thác (lấy gỗ) các cây gỗ đó. Tuy nhiên, sau khi khai thác trưởng làng muốn giữ lại một số cây để làm bóng mát cho con đường thỏa mãn các điều kiện sau:

- Cây thứ k phải được giữ lại.
- Các cây có số thứ tự lớn hơn phải có giá trị lớn hơn.
- Số lượng cây giữ lại là nhiều nhất có thể.

Ví dụ: *7 cây có giá trị tương ứng là 3 7 2 8 6 9 5 thì ta giữ lại các cây có giá trị là 3 7 8 9 (với k=1)*

Yêu cầu: Hãy giúp trưởng làng thực hiện điều đó.

Dữ liệu vào từ file LINETREEK.INP:

- Dòng thứ nhất chứa số nguyên dương n và k ($k \le n \le 10^4$).
- n dòng tiếp theo, dòng thứ i chứa số nguyên dương a_i ($a_i \le 10^9$; i=1,2,…n).

Kết quả ghi vào file LINETREEK.OUT một số nguyên dương duy nhất là số lượng cây giữ lại nhiều nhất thỏa mãn các yêu cầu trên.

Ví dụ:

LINETREEK.INP
7 1 3 7 2 8 6 9 5

LINETREEK.OUT
4

❖ Hướng dẫn bài LINETREEK

➢ Đưa tất cả các phần tử phù hợp sang mảng b có m phần tử

Dãy a: **a[1] a[2] a[3] ... a[k-1] a[k] a[k+1] ... a[n]**

Dãy b: **<a[k] a[k] >a[k]**

- Từ vị trí thứ 1 đến vị trí thứ k-1 của mảng a các phần tử có giá trị <a[k]

```
m=0;
for(i=1; i<=k-1; i++)
        if(a[i]<a[k])
        {       m++;
                b[m]=a[i];
        }
```

- Đưa a[k] sang mảng b

```
m++;
b[m]=a[k];
```

- Từ vị trí thứ k+1 đến vị trí thứ n của mảng a các phần tử có giá trị >a[k]

```
for(i=k+1; i<=n; i++)
        if(a[i]>a[k])
        {       m++;
                b[m]=a[i];
        }
```

➢ Tìm dãy con tăng dài nhất của mảng b có m phần tử

```
d[1]=1;
for(i=2; i<=m; i++)
{       maxdj=0;
        for(j=1; j<=i-1; j++)
                if((d[j]>maxdj) && (b[j]<b[i]))
                        maxdj=d[j];
        d[i]=maxdj+1;
}
maxd=d[1];
for(i=2; i<=m; i++)
        maxd=max(maxd,d[i]);
```

Đáp án là: maxd

Bài 8: Ba lô đặc biệt (Bài 3 - HSGK10_2021-2022) Tên chương trình: SPECPACK.*

Tại một kho báu X có n vật quý hiếm được sắp xếp thành một hàng dài, các vật được đánh số thứ tự từ 1 đến n từ đầu hàng đến cuối hàng. Vật thứ i có giá trị là số nguyên dương a_i $(0 < a_i \leq 10^6)$. Alibaba đi lạc vào kho báu X, sau đó anh ta may mắn được người canh giữ kho báu giúp đỡ bằng cách cho phép mang theo một số vật quý hiếm nhưng các đồ vật này muốn đưa ra được khỏi kho báu cần phải bỏ vào một cái "ba lô đặc biệt", ba lô này có những tính chất như sau:

- *Chỉ đựng được những vật có trong n đồ vật của kho báu; vật bỏ vào sau phải có số thứ tự lớn hơn số thứ tự của vật bỏ vào trước;*
- *Giá trị của vật bỏ vào trước lớn hơn giá trị của vật bỏ vào sau;*
- *Chỉ chứa những vật có giá trị là một số nguyên tố.*

Yêu cầu: Hãy giúp Alibaba tìm số nguyên dương ***D*** là *số lượng vật được chọn nhiều nhất có thể bỏ vào* "*ba lô đặc biệt*" .

Dữ liệu vào: File **SPECPACK.INP** chứa:

- Dòng thứ nhất chứa duy nhất số nguyên dương n $(1 < n \leq 10^5)$;
- Dòng thứ i trong n dòng tiếp theo chứa số nguyên dương a_i $(1 \leq a_i \leq 10^6)$.

Kết quả ra: Ghi vào file **SPECPACK.OUT** duy nhất số nguyên dương ***D*** thỏa mãn yêu cầu bài toán.

Ví dụ:

SPECPACK.INP	SPECPACK.OUT	Giải thích
7 13 8 7 11 7 6 2	4	***4*** *là số lượng vật nhiều nhất có thể chọn thỏa mãn điều kiện bài toán. Cụ thể: thứ tự các món vật được chọn: 1, 4, 5,7 có giá trị tương ứng là: 13, 11, 7, 2*

Ràng buộc dữ liệu:

- 20% tests ứng với: $\mathbf{1 < n \leq 20}$; $\mathbf{1 \leq a_i \leq 10^6}$;
- 50% tests ứng với: $\mathbf{20 < n \leq 10^3}$; $\mathbf{1 \leq a_i \leq 10^6}$;
- 30% tests ứng với: $\mathbf{10^3 < n \leq 10^5}$; $\mathbf{1 \leq a_i \leq 10^6}$.

❖ **Hướng dẫn bài SPECPACK (gần giống bài LINETREEK)**

➢ **Đưa tất cả các phần tử là số nguyên tố sang mảng b có m phần tử**

```
m=0;
for(i=1; i<=n; i++)
        if(a[i] là số nguyên tố)
        {       m++;
                b[m]=a[i];
        }
```

➢ **Tìm dãy con giảm dài nhất của dãy b có m phần tử**

```
d[1]=1;
for(i=2; i<=m; i++)
{       maxdj=0;
        for(j=1; j<=i-1; j++)
                if((d[j]>maxdj) && (b[j]>b[i]))
                        maxdj=d[j];
        d[i]=maxdj+1;
}
maxd=d[1];
for(i=2; i<=m; i++)
        maxd=max(maxd,d[i]);
```

Bài 9: Dãy hình chữ V **Tên chương trình: ROWFIGUV.***

Một dãy số $b_1, b_2,\ldots b_{i-1}, b_i, b_{i+1}\ldots b_k$ được gọi là dãy hình chữ V khi dãy đó thỏa mãn:

$$b_1>b_2>\ldots>b_{i-1}> b_i< b_{i+1}<\ldots <b_k \text{ (với } i=2, 3\ldots k-1)$$

Yêu cầu: Cho một dãy số nguyên $a_1, a_2,\ldots a_n$. Hãy cho biết dãy con hình chữ V dài nhất của nó có bao nhiêu phần tử.

Dữ liệu vào từ file 'ROWFIGUV.INP':

- Dòng đầu ghi số nguyên dương n ($n\leq 10^4$).
- n dòng tiếp theo mỗi dòng là một phần tử của dãy số đã cho (phạm vi mỗi phần tử có trị tuyệt đối không quá 10^9).

Kết quả ghi vào file 'ROWFIGUV.OUT' số lượng phần tử của dãy con hình chữ V dài nhất tìm được.

Ví dụ:

ROWFIGUV.INP
8 6 2 5 9 4 7 4 8

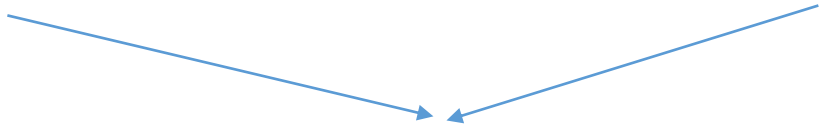
ROWFIGUV.OUT
5

(Giải thích: dãy con hình chữ V dài nhất là 6 5 4 7 8)

Ràng buộc: có 60% test với $n\leq 20$

❖ Hướng dẫn bài ROWFIGUV

a_1 a_2 a_3 . . . . a_i . . . . . a_{n-1} a_n



➢ **Tìm dãy con giảm dài nhất tính từ đầu về cuối dãy: d[i]**

```
d[1]=1;
for(i=2; i<=n; i++)
{       maxdj=0;
        for(j=1; j<=i-1; j++)
                if((d[j]>maxdj) && (a[j]>a[i]))
                        maxdj=d[j];
        d[i]=maxdj+1;
}
```

➢ **Tìm dãy con giảm dài nhất tính từ cuối lên đầu dãy: t[i]**

```
t[n]=1;
for(i=n-1; i>=1; i--)
{       maxtj=0;
        for(j=n; j>=i+1; j--)
                if((t[j]>maxtj) && (a[j]>a[i]))
                        maxtj=t[j];
        t[i]=maxtj+1;
}
```

➢ **Kết hợp thông tin của 2 dãy con trên ta tìm: max(d[i]+t[i])-1**

```
kq=0;
for(i=1; i<=n; i++)
        kq=max(kq, d[i]+t[i]-1);
```

Đáp án là: **kq**

Bài 10: Ba lô 1 **Tên chương trình: BALO1.***

Cho n đồ vật, vật thứ i có khối lượng W_i

Yêu cầu: Hãy tìm cách chọn các đồ vật trong n đồ vật (mỗi loại đồ vật chỉ chọn 1), xếp vào ba lô sao cho tổng khối lượng của chúng lớn nhất nhưng không vượt quá giới hạn qui định là S.

Dữ liệu vào từ file 'BALO1.INP':

- Dòng đầu chứa 2 số nguyên dương n và S ($n \leq 100$; $S \leq 10^5$).
- Trong n dòng sau, dòng thứ i là số nguyên dương W_i ($W_i \leq 10^9$).

Kết quả ghi vào file 'BALO1.OUT' tổng khối lượng lớn nhất lấy được.

Ví dụ:

BALO1.INP
4 17 3 5 6 7

BALO1.OUT
16

❖ **Hướng dẫn bài BALO1**

Gọi t[i,j] là tổng *khối lượng* ***lớn nhất*** được bỏ vào balo khi ta xét tất cả các vật từ 1 đến i: **a_1 a_2 … a_{i-1} a_i** ***không được vượt quá*** khối lượng **j (khối lượng cho phép bỏ vào balo là j)**

➔ t[i,j] <= j

i=1, 2, … n

j=0, 2, … s

➢ Nếu chỉ có 1 vật: **a_1**

t[1,j]=a[1] nếu a[1]<=j

t[1,j]=0 nếu a[1]>j

```
for(j=0; j<=s; j++)
        if(a[1]<=j)
                t[1][j]=a[1];
        else
                t[1][j]=0;
```

➢ Nếu có nhiều vật: a_1 a_2 … a_{i-1} a_i (Đang xét đến a_i có bỏ vào balo hay không?)

- Nếu a[i] không thể bỏ vào balo: a[i]>j

  t[i,j]=t[i-1,j]
- Nếu a[i] có thể bỏ vào balo: a[i]<=j

  Nếu bỏ vào balo: t[i,j]=t[i-1,**j-a[i]**] + a[i]

  Nếu không bỏ vào: t[i,j]=t[i-1,j]

  ⇨ t[i,j]=max(t[i-1,j], t[i-1,j-a[i]] +a[i])

```
for(i=2; i<=n; i++)
        for(j=1; j<=s; j++)
                if(a[i]>j)
                        t[i][j]=t[i-1,j];
                else
                        t[i][j]=max(t[i-1][j], t[i-1][j-a[i]] + a[i]);
```

Đáp án là: t[n,s]

Bài 11: Câu chuyện Alibaba **Tên chương trình: CALIBABA.***

Câu chuyện về Alibaba và 40 tên cướp có lẽ trong chúng ta ai cũng từng nghe biết tới. Có nhiều tình tiết khá ly kỳ rất hấp dẫn, trong đó có một số người kể về một tình tiết như sau:

Trong hang động của 40 tên cướp có rất nhiều đồ quý hiếm, Alibaba một lần nọ đi vào trong hang tính lấy đi một số đồ vật thì bị 40 tên cướp phát hiện. Tuy nhiên, chúng lại không giết Alibaba mà chúng lại đưa ra một trò chơi thử tài trí tuệ tính toán của Alibaba. Tất nhiên, nếu Alibaba mà tính toán được thì chúng cho Alibaba về cùng với những phần quà rất giá trị. Chúng sắp ra trên một đường thẳng n đồ vật, vật thứ i có giá trị là a_i (với i=1, 2,…, n), chúng yêu cầu Alibaba tính xem nếu chúng cho phép Alibaba mang đi một hoặc một số đồ vật bất kỳ, nếu gọi S là tổng giá trị của các đồ vật Alibaba có thể mang đi thì S có thể có những giá trị nào?

Ví dụ: *Có 4 đồ vật có giá trị tương ứng 3 7 2 8*

➔ *Alibaba có thể mang đi các đồ vật ứng với giá trị trong các trường hợp sau:*

3; 7; 2; 8; tổng các giá trị tương ứng là 3; 7; 2; 8

3 7; 3 2; 3 8; 7 2; 7 8; 2 8; tổng các giá trị tương ứng là 10; 5; 11; 9; 15; 10

3 7 2; 3 7 8; 3 2 8; 7 2 8; tổng các giá trị tương ứng là 12; 18; 13; 17

3 7 2 8 tổng các giá trị tương ứng là 20

Tóm lại: *có các tổng là: 2; 3; 5; 7; 8; 9; 10; 11; 12; 13; 15; 17; 18; 20*

Như vậy S có thể nhận 14 giá trị.

Yêu cầu: Hãy cho biết S có thể nhận bao nhiêu giá trị?

Dữ liệu vào từ file **CALIBABA.INP**

- Dòng đầu là số nguyên dương n ($n \le 100$)
- Trong n dòng sau, dòng thứ i chứa số nguyên dương a_i ($a_i \le 100$ với i=1, 2,…, n)

Kết quả ghi vào file **CALIBABA.OUT** một số nguyên duy nhất là số lượng những giá trị S có thể được nhận.

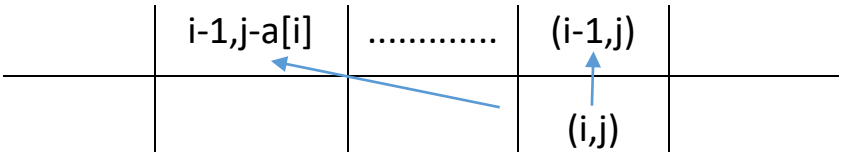
Ví dụ:

CALIBABA.INP
4 3 7 2 8

CALIBABA.OUT
14

❖ **Hướng dẫn bài CALIBABA**

➢ **Gọi dd[i][j]=1 nếu xét từ vật thứ 1 đến vật thứ i có thể tạo được tổng giá trị là j**



---> Dễ thấy: dd[i][j]=1 khi dd[i-1][j]=1 hoặc dd[i-1][j-a[i]]=1

	j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
i	a[i]																					
1	3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	7	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
3	2	1	0	1	1	0	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
4	8	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1	0	1

➢ **Tổng** s=a[1]+a[2]+...+a[n]

➢ **Khởi gán mọi vật đều tạo được tổng giá trị là 0**

```
for (i=1; i<=n; i++)
    dd[i][0]=1;
```

➢ **Xét 1 vật thì chỉ tạo được tổng giá trị là a[1]**

```
dd[1][a[1]]=1;
```

➢ **Xét vật thứ i (giả sử đã xét đến vật thứ i-1)**

```
for (i=2; i<=n; i++)
    for (j=1; j<=s; j++)
      {     if(j>=a[i])
                  if ((dd[i-1][j-a[i]]==1)
                        dd[i][j]=1;
            if (dd[i-1][j]==1)
                  dd[i][j]=1;
      }
```

➢ **Kiểm tra xem với n vật sẽ tạo ra được các tổng giá trị nào?**

```
d=0;
for (j=1; j<=s; j++)
    if (dd[n][j]==1)
        d++;
```

Đáp án là: d

Bài 12: Con đường hoa (Bài 4 - HSGK10_2022-2023) Tên chương trình: PAFLOWER.*

Ở thành phố nọ, để trang trí cho con đường tham quan nhân dịp các ngày lễ, lãnh đạo thành phố đã chỉ đạo trồng những cây hoa ở hai bên lề đường (có thể xem là lề đường A và lề đường B). Sau một thời gian, các cây hoa này đã trưởng thành và có thể phục vụ cho du khách tham quan. Trước dịp tết vừa qua, người ta thấy trong các cây hoa được trồng thì cũng có khá nhiều cây hoa không được đẹp nên lãnh đạo thành phố quyết định đưa ra phương án bỏ đi một số cây hoa và sắp xếp lại sao cho cảnh quan được hài hoà hơn. Người chịu trách nhiệm công việc đó đã đánh dấu các cây hoa được đánh giá là đẹp có số 1, còn các cây hoa coi là xấu được đánh dấu là số -1. Việc bỏ đi các cây hoa xấu có thể làm cho con đường tham quan không còn nhiều cây hoa nữa nên công việc ở đây cần làm phải đảm bảo tất cả các điều kiện sau:

- Không được di chuyển cây hoa ở lề đường A sang lề đường B và ngược lại;
- Các cây hoa trên một lề đường không được thay đổi thứ tự vị trí với nhau;
- Một cây hoa của lề đường A và một cây hoa của lề đường B sẽ tạo thành một cặp;
- Với một cặp cây hoa được giữ lại phải luôn luôn không được hai cây hoa cùng xấu;
- Số cặp cây hoa được giữ lại là nhiều nhất.

Mỗi một lề đường đều có n cây hoa, với lề đường A cây thứ i được đánh dấu là giá trị a_i, với lề đường B cây thứ i được đánh dấu là giá trị b_i (a_i và b_i có giá trị là -1 hoặc 1; $i = 1, 2, \ldots, n$)

Yêu cầu: Hãy cho biết số cặp cây hoa được giữ lại nhiều nhất thoả mãn tất cả các điều kiện nêu trên là bao nhiêu?

Dữ liệu: Vào từ file PAFLOWER.INP:

- Dòng đầu chứa số nguyên dương n ($n \leq 1000$);
- Trong n dòng tiếp theo, dòng thứ i chứa cặp số nguyên a_i và b_i cách nhau ít nhất một khoảng trắng ($i = 1, 2, \ldots, n$).

Kết quả: Ghi vào file PAFLOWER.OUT một số nguyên duy nhất là số lượng cặp cây hoa nhiều nhất được giữ lại thoả mãn tất cả các điều kiện.

Ví dụ:

PAFLOWER.INP
5 -1 -1 1 1 1 -1 -1 -1 -1 1

PAFLOWER.OUT
4

Giải thích: Chọn được nhiều nhất là 4 cặp cây hoa

- *Lề đường A: gồm các cây ở vị trí 1, 2, 3, 4;*
- *Lề đường B: gồm các cây ở vị trị 2, 3, 4, 5.*

❖ Hướng dẫn bài PAFLOWER

➢ *Khai báo:*

long long a[1001], b[1001], d[1001][1001];

➢ *Đọc file:*

```
cin >>n;
for(i=1; i<=n; i++)
        cin >>a[i] >>b[i];
```

➢ *Xử lý:*

Vị trí i: 1 2 3 4 5

Hàng A: -1 1 1 -1 -1

Hàng B: -1 1 -1 -1 1

Gọi d[i][j] là số cặp cây (thoả mãn đề) nhiều nhất khi ta xét

Hàng A: a[1] a[2] ... a[i-1] a[i]

Hàng B: b[1] b[2] ... b[j-1] b[j]

Đáp án: d[n][n]

Bảng d gồm 5 hàng, 5 cột

		j=0	j=1	j=2	j=3	j=4	j=5
			-1	1	-1	-1	1
i=0		0	0	0	0	0	0
i=1	-1	0	0	1	1	1	1
i=2	1	0	1	1	2	2	2
i=3	1	0	1	2	2	3	3
i=4	-1	0	1	2	2	3	4
i=5	-1	0	1	2	2	3	4

i-1, j-1	i-1, j
i, j-1	i, j

*TH1: Hàng A không có cây nào --> tính d[0][j]=0

Hàng A:

Hàng B: b[1] b[2] ... b[j-1] b[j]

```
for(j=0; j<=n; j++)
    d[0][j]=0;
```

*TH2: Hàng B không có cây nào --> tính d[i][0]=0

Hàng A: a[1] a[2] ... a[i-1] a[i]

Hàng B:

```
for(i=0; i<=n; i++)
    d[i][0]=0;
```

*TH tổng quát:

- Nếu 2 cây a[i] và b[j] cùng xấu thì:
  + Nếu vứt đi cây b[j]:
    Hàng A: a[1] a[2] ... a[i-1] a[i]
    Hàng B: b[1] b[2] ... b[j-1]
    ---> d[i][j]=d[i][j-1]
  + Nếu vứt đi cây a[i]:
    Hàng A: a[1] a[2] ... a[i-1]
    Hàng B: b[1] b[2] ... b[j-1] b[j]
    ---> d[i][j]=d[i-1][j]
  Tóm lại: d[i][j]=max(d[i][j-1], d[i-1][j])
- Nếu 2 cây a[i] và b[j] có ít nhất 1 cây đẹp:
  Hàng A: a[1] a[2] ... a[i-1] a[i]
  Hàng B: b[1] b[2] ... b[j-1] b[j]
  ---> *d[i][j]=d[i-1][j-1] + 1*

Cài đặt:

```
for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
        if(a[i]==-1 && b[j]==-1)
            d[i][j]=max(d[i][j-1], d[i-1][j]);
        else
            d[i][j]=d[i-1][j-1] + 1;
```

Đáp án là: d[n][n]

Bài 13: Di chuyển từ tây sang đông **Tên chương trình: WTOE.***

Cho hình chữ nhật m*n ô vuông (các hàng đánh số từ 1 đến m từ trên xuống và các cột đánh số từ 1 đến n từ trái sang phải), mỗi ô chứa một số nguyên. Có thể di chuyển từ một ô sang ô thuộc cột bên phải cùng dòng hoặc chênh lệch một dòng.

Yêu cầu: Hãy tìm cách di chuyển từ một ô nào đó thuộc cột 1 đến một ô nào đó thuộc cột n sao cho tổng các số của các ô đi qua là nhỏ nhất.

Dữ liệu vào từ file 'WTOE.INP':

- Dòng đầu là 2 số nguyên m và n (m, n≤100)
- Tiếp theo là m dòng, mỗi dòng ghi n số nguyên là các số nằm trên các ô vuông của hàng tương ứng bắt đầu từ cột 1 đến cột n. Các số nguyên này có giá trị tuyệt đối không vượt quá 30000

Kết quả ghi vào file 'WTOE.OUT' tổng các số của các ô đi qua.

Ví dụ:

WTOE.INP
3 3 3 3 1 1 2 2 2 1 6

WTOE.OUT
4

❖ **Hướng dẫn bài WTOE**

MẢNG A

	Cột 1	Cột 2	Cột 3
Hàng 0			
Hàng 1	3	3	1
Hàng 2	1	2	2
Hàng 3	2	1	6
Hàng 4			

Gọi t[i,j] là tổng lớn nhất khi ta đi từ cột 1 đến ô (i,j)

i-1, j-1		
i, j-1	i, j	
i+1, j-1		

Để đến được ô (i, j) thì ta chỉ có thể đi từ 1 trong các ô (i-1, j-1), (i, j-1), (i+1, j-1) đến nó.

t[i,j]=min(t[i-1,j-1], t[i,j-1], t[i+1,j-1]) + a[i,j]

MẢNG T

	Cột 1	Cột 2	Cột 3
Hàng 0	1e	1e	1e
Hàng 1	**3**	4	4
Hàng 2	**1**	3	4
Hàng 3	**2**	2	8
Hàng 4	1e	1e	1e

- Thực hiện khởi gán hàng thứ 0, hàng thứ m+1 giá trị dương vô cùng

  t[0,j]=1e9 với j=1, 2, ..., n

  t[m+1,j]=1e9 với j=1, 2, ..., n
- Thực hiện khởi gán cột 1 (từ vị trí của hàng 1 đến hàng thứ m) có giá trị t[i,1]=a[i,1] với i=1, 2, ..., m
- Tính phần còn lại của mảng t

  t[i,j]=min(t[i-1,j-1], t[i,j-1], t[i+1,j-1]) + a[i,j]

  với j=2, 3, ..., n (với mỗi giá trị của j thì i=1, 2, ..., m)

Đáp án là: giá trị nhỏ nhất của cột thứ n của mảng t

❖ Cài đặt:

- Đọc mảng 2 chiều a

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

- Khởi gán hàng 0 và hàng m+1

```
for(j=1; j<=n; j++)
{       t[0][j]=1e9;
        t[m+1][j]=1e9;
}
```

- Khởi gán cột 1 ứng với hàng từ thứ 1 đến m

```
for(i=1; i<=m; i++)
        t[i][1]=a[i][1];
```

- Tạo các phần tử còn lại của mảng t

```
for(j=2; j<=n; j++)
        for(i=1; i<=m; i++)
                t[i][j]=min(min(t[i-1][j-1],t[i][j-1]),t[i+1][j-1]) + a[i][j];
```

- Tìm giá trị nhỏ nhất của cột thứ n của mảng t

```
kq=1e9;
for(i=1; i<=m; i++)
        kq=min(kq,t[i][n]);
```

Khai báo biến: a[101][101], t[101][101]

Bài 14: Cuộc gặp gỡ thú vị **Tên chương trình: MEET.***

Sau khi thất lạc nhau, vợ chồng nhà kiến đã tìm kiếm nhau rất vất vả và cuối cùng họ cũng bắt được tín hiệu với nhau. Tuy nhiên, trước mặt chúng là một thách thức khá thú vị, đó là một lưới ô vuông kích thước n hàng m cột. Có thể mô tả lưới đó được đánh chỉ số hàng từ 1 đến n (trên xuống dưới), chỉ số cột từ 1 đến m (trái sang phải), giao giữa hàng i và cột j là ô vuông (i,j), trên ô vuông (i,j) có lượng thức ăn là a_{ij}. Một con kiến xuất phát từ phía trên (ngoài lưới ô vuông) và một con kiến còn lại xuất phát từ phía dưới (cũng ngoài lưới ô vuông). Vợ chồng nhà kiến quyết định sẽ gặp nhau ở một ô vuông nào đó trong lưới, khi đi vào ô vuông nào thì sẽ lấy đi lượng thức ăn trong ô vuông đó và chỉ có thể đi từ ô vuông có lượng thức ăn ít đến ô vuông có lượng thức ăn nhiều hơn. Hơn nữa, với con kiến đi từ trên xuống thì chỉ có thể đi từ một ô vuông đang đứng xuống được 3 ô ngay ở hàng dưới nó (từ ô (i,j) chỉ có thể hướng đến được 3 ô (i+1,j-1), (i+1,j) và (i+1,j+1)), với con kiến đi từ dưới lên thì chỉ có thể đi từ một ô vuông đang đứng lên được 3 ô ngay ở hàng trên nó (từ ô (i,j) chỉ có thể hướng đến được 3 ô (i-1,j-1), (i-1,j) và (i-1,j+1))

Yêu cầu: Hãy cho biết khi gặp nhau tại một ô nào đó thì lượng thức ăn nhiều nhất của cả vợ chồng nhà kiến có thể kiếm được là bao nhiêu? (tại ô vuông gặp nhau thì lượng thức ăn chỉ tính cho một con kiến)

Dữ liệu vào từ file 'MEET.INP':

- Dòng đầu là số nguyên dương n và m (n, m≤1000).
- n dòng tiếp theo, mỗi dòng là m số nguyên dương. Trong dòng thứ i, số nguyên dương thứ j là a_{ij} ($a_{ij}≤10^5$; i=1,2…n; j=1,2..m).

Kết quả ghi vào file 'MEET.OUT' một số nguyên duy nhất là lượng thức ăn nhiều nhất mà vợ chồng nhà kiến có thể lấy được.

Ví dụ:

MEET.INP
3 4 2 **4** 3 1 **6** 2 4 6 4 **7** 2 4

MEET.OUT
17

❖ Hướng dẫn bài MEET

➢ Đi từ trên xuống:

Gọi t[i,j] tổng lớn nhất đi từ trên xuống đến ô (i,j)

i-1, j-1	i-1, j	i-1, j+1
	i, j	

--> t[i,j]=max(t[i-1,j-1],t[i-1,j],t[i-1,j+1])+a[i,j]

với điều kiện 3 ô đó có thể đi đến được ô i, j

➢ Đi từ dưới lên:

d[i,j] tổng lớn nhất đi từ dưới lên đến ô (i,j)

	i, j	
i+1, j-1	i+1, j	i+1, j+1

--> d[i,j]=max(d[i+1,j-1],d[i+1,j],d[i+1,j+1])+a[i,j]

với điều kiện 3 ô đó có thể đi đến được ô i, j

Kết quả bài toán: max(t[i,j]+d[i,j])-a[i,j] với i=1, 2, ..., n và j=1, 2, ... , m

❖ **Cài đặt:**

- Đọc mảng 2 chiều a

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

➢ **Đi từ trên xuống:**

➢ ***MẢNG T***

	Cột 0	Cột 1	Cột 2	Cột 3	Cột 4	Cột 5
Hàng 1	-1e	**2**	**4**	**3**	**1**	-1e
Hàng 2	-1e					-1e
Hàng 3	-1e					-1e

- Khởi gán cột 0 và cột m+1

```
for(i=1; i<=n; i++)
{       t[i][0]=-1e9;
        t[i][m+1]=-1e9;
}
```

- Khởi gán hàng 1 ứng với cột từ thứ 1 đến m

```
for(j=1; j<=m; j++)
        t[1][j]=a[1][j];
```

- Tạo các phần tử còn lại của mảng t

```
for(i=2; i<=n; i++)
        for(j=1; j<=m; j++)
        {       maxtt=-1e9;
                if(a[i-1][j-1]<a[i][j])
                        maxtt=max(maxtt,t[i-1][j-1]);
                if(a[i-1][j]<a[i][j])
                        maxtt=max(maxtt,t[i-1][j]);
                if(a[i-1][j+1]<a[i][j])
                        maxtt=max(maxtt,t[i-1][j+1]);
                t[i][j]=maxtt+a[i][j];
        }
```

➢ **Đi từ dưới lên:**

➢ ***MẢNG D***

	Cột 0	Cột 1	Cột 2	Cột 3	Cột 4	Cột 5
Hàng 1	-1e					-1e
Hàng 2	-1e					-1e
Hàng 3	-1e	**4**	**7**	**2**	**4**	-1e

- Khởi gán cột 0 và cột m+1

```
for(i=1; i<=n; i++)
{       d[i][0]=-1e9;
        d[i][m+1]=-1e9;
}
```

- Khởi gán hàng n ứng với cột từ thứ 1 đến m

```
for(j=1; j<=m; j++)
        d[n][j]=a[n][j];
```

- Tạo các phần tử còn lại của mảng d

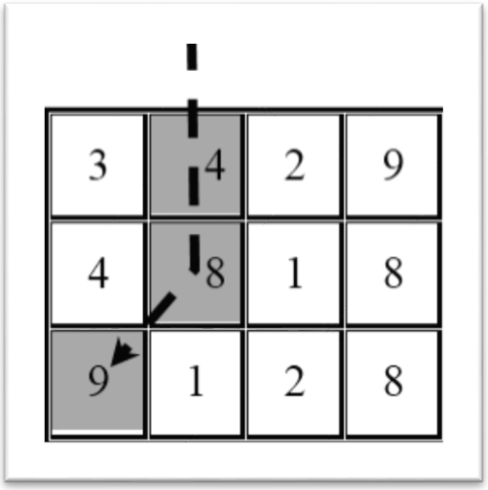
```
for(i=n-1; i>=1; i--)
        for(j=1; j<=m; j++)
        {       maxdd=-1e9;
                if(a[i+1][j-1]<a[i][j])
                        maxdd=max(maxdd,d[i+1][j-1]);
                if(a[i+1][j]<a[i][j])
                        maxdd=max(maxdd,d[i+1][j]);
                if(a[i+1][j+1]<a[i][j])
                        maxdd=max(maxdd,d[i+1][j+1]);
                d[i][j]=maxdd+a[i][j];
        }
```

➢ **Tìm giá trị lớn nhất: max(t[i,j]+d[i,j])-a[i,j]** với i=1,2,...,n và j=1,2,..., m

```
kq=-1e9;
for(i=1; i<=n; i++)
        for(j=1; j<=m; j++)
                kq=max(kq,t[i][j]+d[i][j])-a[i][j];
```

Đáp án là: kq

Bài 15: Thu lượm lương thực (Bài 4 - HSGK12_2022-2023) Tên chương trình: YELLANTS.*



Mỗi năm khi mùa mưa sắp đến, những con kiến thường phải đi kiếm lương thực để tích trữ. Công việc tìm kiếm và thu lượm lương thực của chúng thường theo những kế hoạch đã vạch sẵn. Những chú kiến vàng rất chăm chỉ và luôn tuân theo các quy tắc do họ nhà kiến đặt ra. Khu vực chú kiến vàng kiếm thức ăn thường được mô tả là một lưới ô vuông gồm *m* hàng và *n* cột, hàng được đánh số thứ tự từ 1 đến *m* từ trên xuống và cột được đánh số thứ tự từ 1 đến *n* từ trái qua phải. Ô ở hàng *i*, cột *j* được gọi là ô (*i*, *j*), trên ô (*i*, *j*) của lưới có lượng thức ăn tương ứng với giá trị là a_{ij}. Chú kiến vàng sẽ đi theo lộ trình từ một ô nào đó ở hàng 1 đến một ô nào đó ở hàng thứ *m* của lưới để lấy những thức ăn trong lưới theo nguyên tắc: nếu chú kiến vàng đang đứng ở ô (*i*, *j*) thì chỉ có thể đến được một trong ba ô ở hàng ngay phía dưới là ô (*i*+1, *j*-1); (*i*+1, *j*) và ô (*i*+1, *j*+1). Tuy nhiên, lộ trình đặt ra của chú kiến vàng cũng phải tuân thủ việc lượng thức ăn thu lượm luôn phải tăng dần theo từng ô đã đi qua *(lượng thức ăn của ô chú kiến vàng đến sau sẽ có giá trị lớn hơn lượng thức ăn của ô chú kiến vàng đã đến ngay trước ô đó).*

Yêu cầu: Hãy cho biết *tổng lượng thức ăn nhiều nhất* mà chú kiến vàng có thể thu lượm được là bao nhiêu?

Dữ liệu: vào từ file YELLANTS.INP gồm:

- Dòng đầu là số nguyên dương *m* và *n* (*m*, *n* ≤ 1000);
- Trong *m* dòng tiếp theo, dòng thứ *i* chứa n số nguyên dương a_{ij} ($0 < a_{ij} \le 10^9$; *i* = 1, 2, …, *m; j = 1, 2, …, n*), hai số kế nhau cách nhau ít nhất một khoảng trắng.

Kết quả: ghi vào file YELLANTS.OUT một số nguyên duy nhất là tổng số lượng thức ăn nhiều nhất chú kiến vàng có thể thu lượm được.

(Dữ liệu đầu vào luôn đảm bảo có đường đi)

Ví dụ:

YELLANTS.INP
3 4 3 4 2 9 4 8 1 8 9 1 2 8

YELLANTS.OUT
21

Giải thích: Chú kiến vàng sẽ đi qua các ô có giá trị lần lượt là 4, 8, 9
→ Tổng là: 4+8+9=21)

❖ Hướng dẫn bài YELLANTS (coi bài MEET)