Giải các bài toán tin bằng phương pháp QUY HOẠCH ĐỘNG

Có thể tóm lược nguyên lý QHĐ do Bellman phát biểu như sau: *Quy hoạch động là lớp các bài toán mà quyết định ở bước thứ i phụ thuộc vào quyết định ở các bước đã xử lý trước đó.*

Nhận dạng các bài toán có thể giải bằng phương pháp quy hoạch động.

Một bài toán P muốn giải bằng phương pháp quy hoạch động cần có 2 đặc điểm sau:

- Bài toán P thỏa mãn nguyên lý tối ưu Bellman, nghĩa là có thể sử dụng lời giải tối ưu của các bài toán con từ mức thấp nhất để tìm dần lời giải tối ưu cho bài toán con ở mức cao hơn và cuối cùng là lời giải tối ưu cho bài toán P.

- Bài toán P có các bài toán con phủ chồng lên nhau, nghĩa là không gian bài toán con "hẹp" không tạo dạng hình cây.

Các bước giải quyết bài toán bằng phương pháp quy hoạch động.

Bước 1:Xây dựng hàm mục tiêu

Áp dụng nguyên lý tối ưu của Bellman ta phân rã bài toán ban đầu thành các bài toán con có cùng cấu trúc sao cho việc giải quyết bài toán con cấp i phụ thuộc vào kết quả của các bài toán con trước đó. Cụ thể hóa bước này là ta phải xây dựng được hàm mục tiêu F(i) là nghiệm của bài toán con cấp i.

Bước 2: Xác định các bài toán cơ sở.

Bài toán cơ sở là các bài toán con nhỏ nhất mà ta có thể biết ngay kết quả hoặc tính được kết quả dễ dàng. Đây chính là cơ sở để tính nghiệm cho các bài toán cấp lớn hơn.

Bước 3: Xây dựng công thức truy hồi

Căn cứ vào ý nghĩa của hàm mục tiêu, tìm mối quan hệ giữa các bài toán con các cấp, ta tiến hành xây dựng công thức tính kết quả của bài toán cấp i dựa vào kết quả của các bài toán con cấp trước đó.

Bước 4: Lập bảng phương án

Sử dụng công thức truy hồi và nghiệm các bài toán cơ sở tính nghiệm tất cả các bài toán con và lưu trữ chúng vào bảng phương án.

Bước 5: Kết luận nghiệm của bài toán.

Dựa vào bảng phương án chỉ ra nghiệm của bài toán.

Các bước bước giải quyết trên tuy rất cụ thể nhưng vẫn trừu tượng đối với học sinh. Để các em bước đầu làm quen ta cùng nhau giải quyết các bài toán đơn giản sau:

Bài toán 1: Tìm số Fibonaci thứ N?

Bước 1: Hàm mục tiêu

F(i) là số fibonaci thứ i.

Bước 2: Các bài toán cơ sở

F(1) = 1; F(2) = 1

Bước 3: Công thức truy hồi: F(i) = F(i-1) + F(i-2)

Bước 4: Bảng phương án

i	1	2	3	4	5	6	7	…	…
F(i)	1	1	2	3	5	8	13		

Bước 5: Nghiệm F(N)

Bài toán 2: Di chuyển quân tốt.

Cho một bàn cờ kích thước NxN. Các dòng từ trên xuống dưới, các cột từ trái qua phải được đánh số từ một đến N. Ô nằm ở hàng i, cột j gọi là Ô(i, j). Người ta đặt một con tốt trắng tại Ô(1,1) và M con tốt đen trên các ô còn lại của bàn cờ sao cho không có 2 con tốt nào nằm trên cùng một ô. Ta có thể di chuyển con tốt trắng sang ô bên phải, hoặc xuống dưới ô đang chứa nó nếu như ô đó không chứa tốt đen. Bạn hãy tính xem có bao nhiêu cách di chuyển tốt trắng đến Ô(N, N).

Bước 1: Hàm mục tiêu

F(i, j) là số cách di chuyển quân tốt trắng đến O(i, j).

Bước 2: Các bài toán cơ sở

F(0,j) = 0 ∀j:1..N ;

F(i,0) = 0 ∀i:1..N

F(1,1) = 0;

F(u, v) =0 ∀(u, v) là tọa độ các quân tốt đen

Bước 3: Công thức truy hồi

Ta thấy chỉ đứng ở O(i-1, j) và O(i, j-1) mới có thể để đến được O(i, j). Do đó số cách đến được O(i, j) bằng số cách đến O(i-1, j) cộng với số cách đến O(i,j-1)

Ta được: F(i, j) = F (i-1, j) + F(i, j-1)

Bước 4: Bảng phương án

0	1	2	3	4	5	6
1	T					
2			D			
3						
4		D			D	
5						
6			D			

Minh họa hiện trạng bàn cờ

0	0	0	0	0	0	0
0	1	1	1	1	1	1
0	1	2	0	1	2	3
0	1	3	3	4	6	9
0	1	0	3	7	0	9
0	1	1	4	11	11	20
0	1	2	0	11	22	42

Bảng phương án

Bước 5: Nghiệm F(N, N) = 42

◆**Dạng 1:** ***Tìm dãy các phần tử là dãy con dài nhất thỏa mãn điều kiện bài toán.***

Phương pháp chung

+ Hàm mục tiêu: F(i) là độ dài dãy con
+ Bài toán cơ sở: F(1) = 1
+ Công thức truy hồi: F(i) = Max {1, F(j)+1 } với A_j và A_i thỏa điều kiện bài toán
+ Bảng phương án: Dùng mảng 1 chiều lưu trữ.
+ Nghiệm bài toán: F(N)

Các bài tập áp dụng:

Bài 1: Xếp Tháp *Tên chương trình:* ***Tower.Pas***

Một lần nữa, Bờm lại thể hiện được mình không chỉ là người học giỏi mà chơi cũng rất giỏi. Ở lớp Bờm, các bạn đang rất thích chơi xếp tháp và cậu đang thể hiện mình là người vô địch trong trò chơi này khi thắng rất nhiều bạn cùng lớp. Liệu bạn có thể thắng được Bờm?

Ở trò chơi này, người chơi được cho N hình trụ đứng với nhiều kích thước khác nhau và yêu cầu người chơi xếp được tòa tháp cao nhất từ các hình trụ theo đúng thứ tự từ 1 đến N sao cho khối ở trên phải được xếp khít với khối ở dưới, hay đường kính đáy của hình trên không vượt quá đường kính đáy hình dưới. Một khối trụ có thể dùng hoặc không dùng nhưng phải đúng theo thứ tự đã cho.

Dữ liệu: Cho từ file **Tower.Inp**

Dòng đầu ghi số N (N≤ 1000000)

N dòng tiếp theo mỗi dòng ghi 2 số R_i và H_i là bán kính đáy và chiều cao hình trụ thứ i.

Kết quả: Ghi ra file **Tower.Out**

Một số nguyên duy nhất là chiều cao lớn nhất của tòa tháp xếp được.

Ví dụ:

Tower.Inp	Tower.Out
4 4 2 2 5 1 3 3 1	10

Nhận xét: Đây là bài toán tìm dãy con không tăng theo R_i có tổng chiều cao lớn nhất.

Hướng dẫn

Gọi F[i] là độ cao lớn nhất của tòa tháp xếp được với khối i là đỉnh.

F[i] = H[i] ∀i:1..N;

Công thức truy hồi:

Nếu khối i xếp được trên khối j (1≤ j < i) hay R[i] ≤ R[j].Ta được:

F[i] = Max{F[j]+H[i]} ∀j:1..i-1.

Nghiệm bài toán: Max{F[i]} ∀i:1..N

Bài 2: Dãy số WAVIO:

Dãy số Wavio là dãy số nguyên thỏa mãn các tính chất : các phần tử đầu sắp xếp thành 1 dãy tăng dần đến 1 phần tử đỉnh sau đó giảm dần. Ví dụ dãy số *1 2 3 4* **5** *2 1* là 1 dãy Wavio độ dài 7. Cho 1 dãy gồm N số nguyên, hãy chỉ ra một dãy con Wavio có độ dài lớn nhất trích ra từ dãy đó .

Hướng dẫn:

- L1[i] là độ dài lớn nhất của 1 dãy con tăng dần của dãy A_1, A_2,...A_i .
- L2[i] là độ dài lớn nhất của 1 dãy con giảm dần của dãy A_i, A_2,...A_N.
- Công thức truy hồi giống như bài toán dãy con tăng.
- Nghiệm bài toán: Tìm phần tử j trong 2 mảng L1, L2 thỏa mãn L1[j]+L2[j] lớn nhất.

Bài 3: Xâu con đối xứng *Tên chương trình:* ***Palin.Pas***

Xâu đối xứng là xâu mà khi đọc từ bên trái sang phải hay ngược lại đều cho một kết quả. Bạn hãy tìm một xâu con đối xứng dài nhất của một xâu cho trước. Xâu con của một xâu S có được bằng cách bỏ đi một số kí tự của S.

Dữ liệu: **Palin.Inp** gồm một dòng duy nhất ghi xâu S có độ dài không quá 10000 kí tự.

Kết quả: **Palin.Out** gồm một số duy nhất là độ dài lớn nhất tìm được

Ví dụ:

Palin.Inp	**Palin.Out**
banana	5

Hướng dẫn

Gọi F[i,len] là chỉ số j lớn nhất sao cho j ≤ i và trong đoạn xâu kí tự j đến kí tự thứ i có tồn tại xâu con đối xứng có độ dài len. Nếu không tìm được j thì F[j, len]=0.

F[i, 1] = i ∀i:1..N

F[i, 2] bằng Max của F[i-1, 2] và j với j lớn nhất nhỏ hơn i và kí tự thứ j bằng kí tự thứ i trong xâu ban đầu.

Công thức truy hồi:

- F[i, len] = F[i-1, len] nếu không chọn kí tự i vào xâu đối xứng.
- F[i, len] = giá tri j lớn nhất nhỏ hơn F[i-1, len-2] và kí tự thứ j bằng kí tự thứ i.

◆***Dạng 2: Tìm dãy các phần tử là dãy con chung dài nhất***

Phương pháp chung

Giả sử cho 2 dãy A_1, A_2,…, A_M; B_1, B_2,…, B_N (Các phần tử của dãy có thể là số hoặc kí tự). Yêu cầu tìm dãy con chung dài nhất của hai dãy đã cho.

+Hàm mục tiêu: F(i, j) là độ dài dãy con chung dài nhất của hai dãy A_1, A_2,…, A_i; B_1, B_2,…, B_j.

+ Các bài toán cơ sở: F(0, j) = 0 = F(i, 0) ∀i:1..M; ∀j:1..N

+ Công thức truy hồi: 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)

+ Bảng phương án: Dùng mảng hai chiều để lưu trữ.

+ Nghiệm bài toán: F(M, N)

Các bài tập áp dụng:

Bài 1: Xâu con chung dài nhất

Cho 2 xâu X,Y. Hãy tìm xâu con của X và của Y có độ dài lớn nhất.

Hướng dẫn

- Gọi L(i,j) là *độ dài xâu con chung dài nhất của xâu X(i) gồm i kí tự phần đầu của X* (X(i)=X[1..i]) và *xâu Y(j) gồm j kí tự phần đầu của Y* (Y(j) =Y[1..j]).

- Bài toán cơ sở

  L(0,j)=L(i,0)=0.

- Công thức truy hồi

  L(i,j) = L(i 1,j 1)+1 nếu X[i] = Y[j].

  L(i,j) = max(L(i 1,j), L(i,j 1)) nếu X[i]≠Y[j].

Bài 2: Dãy con chung bội hai dài nhất *Tên chương trình:* **Lcs2x.*Pas***

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 (Lcs2x.Inp)

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 (Lcs2x.Out)

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

Lcs2x.Inp

```
1
5 5
5 1 6 10 20
1 8 6 10 20
```

Lcs2x.Out

```
3
```

Hướng dẫn:

Gọi F[i, j] là độ dài dãy con chung dài nhất của dãy A[1..i] và dãy B[1..j] thỏa điều kiện đề bài và A[i]=B[j].

Công thức truy hồi:

- Nếu A[i]<>B[j] thì F[i, j]=0

- F[i, j]= Max{F[u, v]/ u<i, v<j}+1 và thỏa điều kiện đề bài. Có nghĩa là tồn tại 2 chỉ số k, t sao cho Max{F[u, v]/ u<i, v<j}=F[k,t] và A[k]*2<=A[i]; B[t]*2<=B[j]

◆***Dạng 3: Tìm dãy gồm các phần tử thỏa mãn điều kiện nào đó***

Phương pháp chung

Giả sử có N loại phần tử có "khối lượng" là A_1, A_2,…, A_N và "giá trị" là B_1, B_2,…, B_N . Yêu cầu chọ ra một số lượng các loại phần tử có "khối lượng" phụ thuộc M để giá trị đạt được là nhỏ nhất hoặc lớn nhất có thể.

+Hàm mục tiêu: F(i) là tổng giá trị các phần tử được chọn để tổng "khối lượng" các phần tử là i thỏa điều kiện bài toán.

+ Bài toán cơ sở: F(0) = 0

+ Công thức truy hồi:

$F(k) = Max\{F(k-A_i)\}+B_i$ với k:1..S

(Tùy thuộc vào yêu cầu của bài toán, có thể là Min)

+ Bảng phương án: Dùng mảng một chiều để lưu trữ.

+ Nghiệm bài toán: F(k) với k thỏa điều kiện bài toán.

Các bài tập áp dụng:

Bài 1: MONSTER *Tên chương trình*: ***Monster.Pas***

Bờm phải đi qua một thung lũng đầy cướp. Mỗi tên cướp có một chỉ số thể hiện sự hung dữ. Cụ thể tên cướp thứ i có mức độ hung dữ là một số nguyên H[i]. Bờm đứng trước nguy cơ bị bọn cướp tấn công, tuy nhiên Bờm ta có thể sống sót bằng những đồng vàng của mình để mua chuộc một số tên cướp. Tên cướp thứ i sẽ đòi T[i] đồng vàng để bảo vệ cho Bờm. Khi gặp một tên cướp, nếu nó có độ hung dữ lớn hơn tổng độ hung dữ của số tên cướp mà Bờm đã mua chuộc thì tên cướp đó sẽ tấn công Bờm. Nói các khác để tồn tại Bờm bắt buộc phải mua chuộc tên cướp này. Trong trường hợp ngược lại khi gặp một tên cướpcó độ hung dữ nhỏ hơn hoặc bằng tổng độ hung dữ của số tên cướp mà Bờm đã mua chuộc thì nó không thể tấn công Bờm và Bờm có thể mua chuộc nó hoặc không. Bạn hãy tính số vàng ít nhất mà Bờm phải dùng để vượt qua được thung lũng.

***Input*: Monster.inp**

Dòng đầu ghi số nguyên dương N (N<1000) và V (V<10000) là số vàng Bờm có.
Dòng thứ 2 ghi N số nguyên H_1, H_2, …, H_N. ($H_i<10^9$)
Dòng thứ 3 ghi N số nguyên dương T_1, T_2, …, T_N. ($T_i<10$)

***Output*: Monster.out**

Ghi ra số vàng ít nhất Bờm sử dụng nếu vượt qua được thung lũng,ngược lại ghi -1

***Ví dụ*:**

Monster.inp	Monster.out
3 5 8 5 10 1 1 2	2

Hướng dẫn giải bài MONSTER

- Hàm mục tiêu: Gọi F[i, j] là tổng độ hung dữ lớn nhất của các tên cướp mà Bờm mua chuộc được khi đi qua i tên cướp đầu tiên và dùng j đồng tiền vàng. Nếu Bờm không thể đi qua i tên cướp đầu tiên với j đồng vàng thì F[i, j] = -1.

- Các bài toán cơ sở:
  F[1, j] = H[1] nếu j ≥ T[1] ngược lại F[1, j] = -1.
- Công thức truy hồi: (i > 1)
  Bờm mua chuộc tên cướp thứ i thì
  F[i, j] = F[i-1, j-T[i]]+H[i] (điều kiện: F[i-1, j-T[i]] ≠ -1)
  Bờm không mua chuộc tên cướp thứ i thì
  F[i, j] = F[i-1, j] (điều kiện: F[i-1, j] ≥ H[i])
- Nghiệm của bài toán là j nhỏ nhất sao cho F[n, j] <> -1 hoặc F[n,v] nếu F[n,v]= -1.

Bài 2: Hệ thống tiền tệ *Tên chương trình:* ***Money.Pas***

Chúng ta hãy cùng theo dõi một hệ thống tiền tệ rất riêng. Đó là giá trị của những đồng xu. Thông thường các đồng xu có các giá trị như, 5, 10, 20 đơn vị, đôi khi đồng xu 2 đơn vị cũng được dùng để đo lường tốt hơn.

Mọi người đều muốn biết có bao nhiêu cách khác nhau để có thể lấy ra một lượng tiền cho trước bằng cách dùng các hệ thống tiền xu khác nhau. Ví dụ dùng hệ thống [1, 2,…,18] có thể lấy ra 18 đồng bằng nhiều cách khác nhau như: 1 xu 18 đơn vị, 2 xu 9 đơn vị, 2 xu 8 đơn vị và 2 xu 1 đơn vị,… và nhiều cách khác.

Yêu cầu: Với hệ thống tiền xu cho trước và một lượng tiền S hãy cho biết có bao nhiêu cách lấy được số tiền S từ hệ thống tiền xu đã có.

Dữ liệu: Cho từ file **Money.Inp**
- Dòng đầu ghi hai số N (1<N<25) là số loại tiền trong hệ thống và S (S<10000)
- Dòng thứ hai ghi N số nguyên dương là các mệnh giá trong hệ thống tiền tệ.

Kết quả: Ghi ra file **Money.Out**
Ghi ra một số duy nhất là số cách tìm được.

Ví dụ:

Money.Inp	Money.Out
3 10 1 2 5	10

Hướng dẫn
- Gọi F[i, j] là số cách tạo được số tiền là j từ i đồng xu đầu tiên.
- F[0, j]=0 ∀j:1..S; F[i, 0]=1 ∀i:1..N; F[0,0]=0

Công thức truy hồi:
F[i, j] = F[i-1, j] Nếu không sử dụng đồng xu thứ i.
F[i, j] = F[i-1, j] + F[i-1, j-A[i]] Nếu sử dụng ít nhất 1 đồng xu thứ i. (j>A[i])

Bài 3: Du lịch vòng quanh thế giới *Tên chương trình:* ***Travel.Pas***

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: Vào từ File **Travel.Inp**

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: Ghi ra File **Travel.Out**

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:

Travel.Inp	**Travel.Out**
4	500
300	2
250 310 550 590	2 4

Hướng Dẫn

- 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)}; ∀j=0,..i-1

◆***Dạng 4: Ghép cặp***

Phương pháp chung

Giả sử cho hai dãy phần tử, dãy 1 được đánh số từ một đến K, dãy 2 được đánh số từ 1 đến N. Khi ghép phần tử i dãy 1 với phần tử j của dãy 2 được giá trị V[i, j]. Tìm cách ghép sau cho tổng giá trị thu được là lớn nhất hoặc nhỏ nhất.

+ Hàm mục tiêu: F(i, j) là tổng giá trị thu được khi ghép từ i phần tử đầu của dãy 1 với j phần tử đầu của dãy 2 thỏa điều kiện bài toán.

+ Các bài toán cơ sở: F(0, j) = 0 = F(i, 0)

+ Công thức truy hồi:

F(i, j) = Max{F(i-1, j-1)+v[i, j]; F(i, j-1; F(i-1, j)}

+ Bảng phương án: Dùng mảng hai chiều để lưu trữ.

+ Nghiệm bài toán: F(K,N) với k thỏa điều kiện bài toán.

Các bài tập áp dụng:

Bài 1: Nối Điểm (Wires) *Tên chương trình:* ***Wires.Pas***

Trên Hai đường thẳng song song L1 và L2, người ta đánh dấu trên mỗi đường N điểm. Các điểm trên đường thẳng L1 được đánh số từ 1 đến N, từ trái qua phải, còn các điểm trên đường thẳng L2 được đánh số bởi P[1], P[2],..., P[N] cũng từ trái qua phải, trong đó P[1], P[2],.., P[N] là một hoán vị của các số 1, 2,..., N.

Ta gọi các số gán cho các điểm là số hiệu của chúng. Cho phép nối hai điểm trên 2 đường thẳng có cùng số hiệu.

Yêu Cầu: Tìm cách nối được nhiều cặp điểm nhất với điều kiện các đoạn nối không được cắt nhau .

Dữ Liệu: Vào từ File **Wires.Inp**

- Dòng đầu tiên chứa số nguyên dương N (N<=10000)
- Dòng thứ hai chứa các số P[1], P[2],....,P[N]

Kết Quả: Ghi Ra File **Wires.Out**

- Dòng đầu chứa số K là số lượng đoạn nối tìm được.
- Dòng tiếp theo chứa K số hiệu của các đầu mút của các đoạn nối được ghi theo thứ tự tăng dần.

Ví Dụ :

WIRES.INP	**WIRES.OUT**
9	5
2 5 3 8 7 4 6 9 1	2 3 4 6 9

Hướng dẫn:

- Hàm mục tiêu: Gọi F(i) là số đoạn thẳng tối đa của các cặp nối của các điểm từ 1 đến i.
- Các bài toán cơ sở: F[0]=0;
- Công thức truy hồi:

  F[i] = Max{F[P[j]]+1} ∀j:=1..ViTri(i) trong dãy P[1], P[2],.., P[N].
- Nghiệm bài toán: F[N].

Bài 2 Công trình *Tên chương trình:* ***Congtrinh.Pas***

Có N công trình cần vật liệu thi công. Công trình i cần cung cấp D[i] đơn vị hàng. Hàng được cung cấp từ hai kho A và B. Cước vận chuyển một đơn vị hàng từ kho A đến công trường i là A[i]. Cước vận chuyển một đơn vị hàng từ kho B đến công trường i là B[i]. Biết kho A có R đơn vị hàng và tổng số hàng của hai kho vừa đủ cung cấp cho N công trường. Hãy phân phối hàng từ hai kho đến các công trường sao cho tổng cước phí vận chuyển là ít nhất.

Dữ liệu cho từ file "congtrinh.inp"

Dòng đầu chứa 2 số N R (N,R <1000)
Dòng 2 chứa N số D[1] D[2]…D[N]
Dòng 3 chứa N số A[1] A[2]…A[N]
Dòng 4 chứa N số B[1] B[2]…B[N]

Kết quả xuất ra file "Congtrinh.out"

Ghi tổng chi phí vận chuyển ít nhất

Ví dụ:

Congtrinh.inp	**Congtrinh.out**
5 100 30 80 80 50 40 3 5 10 4 23 4 6 2 7 4	1070

Hướng dẫn

- Hàm mục tiêu: S(i ,j) là cước phí vận chuyển nhỏ nhất trong trường hợp kho A cung cấp j đơn vị hàng đến các công trường 1..i. (j ≤ R)
- Công thức truy hồi:

Nhận xét: S(1,j) = Min {A[1]*x + B[1]*(D[1]-x) } với x là số đơn vị hàng cung cấp từ kho A ($x \le D[1]$ và $x \le j$)

Do đó S(i, j) = Min {A[i]*x + B[i]*(D[i]-x) + S(i-1, j-x)}

Bài toán 3. Cắm hoa

Cần cắm k bó hoa khác loại nhau vào n lọ xếp thẳng hàng sao cho hoa có số hiệu nhỏ được đặt trước hoa có số hiệu lớn. Với mỗi loại hoa i ta biết giá trị thẩm mỹ khi cắm hoa đó vào lọ j, v[i,j]. Các số liệu đều là số tự nhiên và được ghi cách nhau bởi dấu cách trên mỗi dòng.

Dữ liệu vào ghi trong tệp văn bản HOA.INP: dòng đầu tiên là hai trị k và n. Từ dòng thứ hai trở đi là các giá trị v[i,j] với i:=1..k và j:=1..n; 1 <= k <= n <= 100.

Dữ liệu ra ghi trong tệp văn bản HOA.OUT gồm hai dòng: dòng đầu tiên là tổng giá trị thẩm mỹ của phương án cắm hoa tối ưu. Từ dòng thứ hai là dãy k số hiệu lọ được chọn cho mỗi bó hoa.

Thí dụ:

```
HOA.INP
  4  6
  1  1  6  4  3 10
  9  1  4  7  2  7
  7  2  6 10  2  3
  6 10  7  1  3  9
```

```
HOA.OUT
2 4
2 3 4 6
```

◆***Dạng 5: Xác định đường đi ngắn nhất***

Phương pháp chung

Giả sử có N thành phố, độ dài đường đi giữa hai thành phố i, j là A[i, j]. Tìm đường đi ngắn nhất giữa hai thành phố bất kỳ.

+ Hàm mục tiêu: F(i, j) là độ dài đường đi ngắn nhất từ i đến j.
+ Bài toán cơ sở: F(i, i) = 0
+ Công thức truy hồi: F(i, j) = Min{F(i, k)+F(k, j)} ∀k:1..N
+ Bảng phương án: Dùng mảng hai chiều để lưu trữ
+ Nghiệm bài toán: F(u,v) là đường đi ngắn nhất giữa 2 tphố u, v

Các bài tập áp dụng:

Bài 1: Di chuyển trên các hình tròn

Cho N hình tròn (đánh số từ 1 đến N). Một người muốn đi từ hình tròn này sang hình tròn khác cần tuân theo qui ước:

- Nếu khoảng cách giữa 2 điểm gần nhất của 2 hình tròn không quá 50 cm thì có thể bước sang.

- Nếu khoảng cách này hơn 50cm và không quá 80cm thì có thể nhảy sang.

- Các trường hợp khác không thể sang được.

Một đường đi từ hình tròn này sang hình tròn khác được gọi là càng "tốt" nếu số lần phải nhảy là càng ít. Hai đường đi có số lần nhảy bằng nhau thì đường đi nào có số hình tròn đi qua ít hơn thì đường đi đó "tốt" hơn.

Các hình tròn được cho trong một file văn bản, trong đó dòng thứ i mô tả hình tròn số hiệu i (i = 1, 2,..., N) bao gồm 3 số thực: hoành độ tâm, tung độ tâm, độ lớn bán kính (đơn vị đo bằng mét).

Lập trình đọc các hình tròn từ một file văn bản (tên file vào từ bàn phím), sau đó cứ mỗi lần đọc số hiệu hình tròn xuất phát S và hình tròn kết thúc T từ bàn phím, chương trình sẽ đưa ra đường đi từ S đến T là "tốt nhất" theo nghĩa đã nêu (hoặc thông báo là không có).

Yêu cầu đường đi được viết dưới dạng một dãy các số hiệu hình tròn lần lượt cần được đi qua trong đó nói rõ tổng số các bước nhảy, tổng số các hình tròn đi qua và những bước nào cần phải nhảy.

Bài 2: **Mua vé**

Có N người xếp hàng mua vé. Ta đánh số họ từ 1 đến N theo thứ tự trong hàng. Thời gian phục vụ bán vé cho người thứ i là T[i]. Mỗi người cần mua 1 vé nhưng được quyền mua tối đa 2 vé, vì thế một số người có thể nhờ người đứng ngay trước mình mua hộ. Người thứ i nhận mua vé cho người thứ i+1 thì thời gian mua vé cho 2 người là R[i], tìm phương án sao cho N người đều có vé với thời gian ít nhất.

Dữ liệu vào từ file "Tick.inp"

Dòng thứ nhất ghi số N (1<N<2000)

Dòng thứ 2 ghi N số nguyên T[1] T[2] …T[n]

Dòng thứ 3 ghi N-1 số nguyên R[1] R[2] …R[N-1]

Kết quả ghi ra file "Tick.out"

Dòng đầu ghi tổng thời gian phục vụ bán vé.

Dòng tiếp theo ghi chỉ số của các khách hàng cần rời khỏi hàng (Nếu không có ai rời khỏi hàng thì ghi số 0).

Hướng dẫn:

- Hàm mục tiêu: F(k) là thời gian ít nhất để phục vụ vé cho k người từ người 1 đến người k.
- Công thức truy hồi:

  Ta thấy nếu người k tự mua vé thì thời gian ít nhất phục vụ k người là T[k]+F(k-1), nếu người thứ k nhờ người thứ k-1 mua hộ thì thời gian ít nhất phục vụ k người là R[k-1]+F(k-2)

  Do đó: F(k) = Min(T[k]+F(k-1); R[k-1]+F(k-2))
- Tổ chức dữ liệu: Mảng L[1..N], L[i] lưu trữ thời gian ít nhất phục vụ vé cho i người.
- Sau khi xây dựng xong mảng L, dựa vào 3 mảng L, T, R ta tìm ra kết quả những người nào cần mua vé ghi vào mảng KQ theo nhận xét:

  Nếu T[k]+L[k-1]≤R[k-1]+L[k-2] thì người k mua vé (KQ[k]=1) ngược lại thì người k-1 mua vé (KQ[k-1]=1).

◆*Dạng 6: Di chuyển*

Phương pháp chung

Đối với các bài toán có dạng di chuyển thì việc xác định hàm mục tiêu cần căn cứ vào quy luật di chuyển cho phép và cần phải chú ý vào giá trị biên. Đây là dạng toán thường gặp tuy nhiên việc xây dựng hàm mục tiêu và công thức truy hồi là khá dễ dàng.

Các bài tập áp dụng:

Bài 1: KHU VUI CHƠI *Tên chương trình* XYZ.PAS

Bản đồ khu vui chơi là một hình chữ nhật có kích thước MxN ô vuông. Khu vui chơi có một cổng vào đặt tại ô (1, 1) và một cổng ra đặt tại ô (M, N). Mỗi ô (i, j) được bố trí một trò chơi, giá vé vào ô (i, j) là C[i, j]. Tại mỗi ô khách có thể di chuyển sang các ô chung cạnh bên phải hoặc phía dưới. Vào ngày nghỉ, Bờm quyết định tham quan khu vui chơi. Trong khu vui chơi có một trò chơi Bờm rất thích đặt tại ô (U, V) và nhất định Bờm phải tham gia trò chơi này. Nhưng số tiền có hạn Bờm không biết phải đi theo lộ trình nào để chơi được trò chơi mình thích và ít tốn tiền nhất. Bạn hãy giúp Bờm nhé.

Dữ liệu cho từ file XYZ.INP

- Dòng đầu là 4 số M, N, U, V ($3 \le M, N \le 10000$; $1 \le U \le M$; $1 \le V \le N$)
- M dòng tiếp theo, mỗi dòng ghi N số thể hiện ma trận chi phí C, với C[i, j] là giá vé khi vào ô (i, j).

Kết quả ghi ra file XYZ.OUT một số duy nhất là chi phí ít nhất tìm được.

Ví dụ:

XYZ.INP	XYZ.OUT
4 5 2 3 5 6 4 4 1 1 2 9 1 5 1 1 1 2 3 4 6 5 1 4	25

Hướng dẫn

- Gọi F(i, j) là chi phí ít nhất khi đến Ô(i, j)
- Công thức truy hồi: F(i, j) = Min{ F(i, j-1), F(i-1, j)}+A[i]
- Do bắt buộc phải qua Ô(U,V) nên kết quả bài toán chính là tổng chi phí thấp nhất đi từ Ô(1,1) đến Ô(u,v) và chi phí thấp nhất đi từ Ô(u,v) đến Ô(m,n).

Bài 2: NHẢY *Tên chương trình:* ***Jump.Pas***

Trên trục tọa độ Ox có N vị trí được đánh dấu sẵn, vị trí i có tọa độ X_i . Xét một trò chơi như sau:

Nhiệm vụ chính của bạn sẽ thực hiện các bước nhảy giữa các điểm đã đánh dấu. Trước khi bắt đầu trò chơi bạn phải chọn điểm bắt đầu (là một trong N vị trí đã đánh dấu) và chiều nhảy thuận theo chiều tia Ox hoặc ngược lại. Bạn được thực hiện số bước nhảy tùy ý với điều kiện chiều nhảy phải giống nhau và độ dài bước nhảy phải là một số dương và lớn hơn hoặc bằng bước nhảy liền trước. Tất nhiên với những điều kiện này bạn không thể thực hiện quá N bước nhảy. Mỗi vị trí i có một số điểm tương ứng là P_i, điểm của bạn là tổng số điểm các vị trí bạn đến được (bao gồm cả điểm xuất phát).

Yêu Cầu: Tìm cách nhảy để đạt được số điểm lớn nhất.

Dữ Liệu: Vào từ File **Jump.Inp**

- Dòng đầu ghi số nguyên dương N ($1<N<1000$)
- N dòng tiếp theo mỗi dòng thứ i ghi 2 số nguyên tương ứng là X_i, P_i. ($1<X_i, P_i<10^6$)

Kết Quả: Ghi ra File **Jump.Out**

Một số duy nhất là số điểm lớn nhất đạt được.

Ví Dụ:

Jump.Inp	Jump.Out
6 5 6 1 1 10 5 7 6 4 8 8 10	25

Giải thích: Các vị trí nhảy qua: 4 5 7 10 (tương ứng điểm nhận được: 8+6+6+5=25)

Hướng Dẫn

Để đơn giản ta có thể xem như các vị trí đã được sắp xếp tăng dần theo X_i. Ngoài ra ta chỉ xét việc nhảy theo chiều X_i tăng, việc nhảy ngược lại có thể được làm tương tự bằng cách đảo dấu các tọa độ X_i và thứ tự các vị trí.

- Gọi F[i, j] là số điểm đạt được lớn nhất có thể nếu vị trí cuối cùng là vị trí j và vị trí liền trước là i.
- Gọi k là vị trí liền trước i, ta có X[i]-X[k]<=X[j]-X[i].

  Do đó F[i, j]= Max{F[k,i] / ∀k liền trước i}+P[j]