Chuyên đề

PHƯƠNG PHÁP QUY HOẠCH ĐỘNG

I. GIỚI THIỆU

Trong chương trình Tin học chuyên ngành ở các trường đại học sư phạm, phương pháp quy hoạch động hầu như không được nhắc đến hoặc chỉ giới thiệu sơ qua nên nhiều sinh viên mới ra trường tỏ ra lúng túng khi gặp những bài toán dạng này trong việc giảng dạy môn Tin học dành cho các lớp chuyên Tin ở phổ thông. Hơn nữa, tài liệu, sách báo viết về phương pháp quy hoạch động rất ít, hầu hết giáo viên đều tự tìm tòi, nghiên cứu, tự biên soạn chương trình để giảng dạy. Mấy năm gần đây Bộ Giáo dục và Đào tạo mới chính thức đưa ra chương trình giảng dạy dành cho các lớp chuyên Tin ở các trường phổ thông, trong đó quy hoạch động là một chuyên đề bắt buộc trong chương trình tin học chuyên của lớp 11 chiếm một thời lượng lớn (15 tiết). Những bài toán áp dụng phương pháp quy hoạch động cũng thường xuyên xuất hiện trong các đề thi học sinh giỏi Tin học cấp tỉnh, cấp quốc gia, Olympic cho nên việc nắm được kiến thức, kỹ năng giải bài toán quy hoạch động của giáo viên dạy chuyên Tin cũng như học sinh trong các đội tuyển Tin học là vấn đề rất cần thiết.

Bản thân tôi là một giáo viên giảng dạy môn Tin học trong trường THPT chuyên Lê Quý Đôn – Khánh Hoà, đã qua nhiều năm tham gia bồi dưỡng đội tuyển học sinh giỏi môn Tin học ở cấp trường, cấp tỉnh và qua hai năm giảng dạy lớp chuyên Tin, tôi đã rút ra được một số kinh nghiệm trong việc giảng dạy chuyên đề quy hoạch động. Sau đây tôi xin được giới thiệu một số kinh nghiệm trong việc sử dụng phương pháp quy hoạch động để giải một số bài toán trong tin học.

Quy hoạch động là một phương pháp nhằm giảm thời gian chạy của các thuật toán thể hiện các tính chất của các bài toán con gối đầu nhau (*overlapping subproblem*) và cấu trúc con tối ưu (*optimal substructure*). Phương pháp quy hoạch động do nhà toán học Richard Bellman pháp minh vào năm 1953.

Phương pháp quy hoạch động (*dynamic programming*) là một kỹ thuật được áp dụng để giải nhiều lớp bài toán, đặc biệt là các bài toán tối ưu. Đây là phương pháp có ý nghĩa trong việc giải quyết các bài toán thực tế.

Phương pháp này dựa trên nguyên lý tối ưu của Bellman: *Nếu một dãy các lựa chọn là tối ưu thì mọi dãy con của nó cũng là tối ưu.*

Ý tưởng của phương pháp: Để không phải tính lại kết quả bài toán con ta lưu các kết quả đã tính vào một bảng và chỉ việc sử dụng lại giá trị của nó khi cần thiết.

Khác với phương pháp chia để trị là kỹ thuật tiếp cận TOP - DOWN (đi từ trên xuống), phương pháp quy hoạch động dùng kỹ thuật BOTTOM - UP (đi từ dưới lên):

Xuất phát từ các trường hợp riêng đơn giản nhất, có thể tìm ngay ra nghiệm. Bằng cách kết hợp nghiệm của chúng, ta nhận được nghiệm của bài toán cỡ lớn hơn. Cứ thế tiếp tục, chúng ta sẽ nhận được nghiệm của bài toán ban đầu.

Trong quá trình tìm nghiệm của bài toán từ dưới lên chúng ta sẽ sử dụng bảng để lưu giữ kết quả của các bài toán con đã giải trước đó. Kết quả của những bài toán con này có thể là cơ sở cho việc giải bài toán lớn hơn.

Khi giải một bài toán con, cần đến nghiệm của bài toán con nhỏ hơn, ta chỉ cần lấy kết quả từ bảng, không cần phải giải lại. Chính vì thế mà giải thuật nhận được bằng phương pháp này rất có hiệu quả.

Ưu điểm của phương pháp quy hoạch động: chương trình chạy nhanh.

Phạm vi áp dụng của phương pháp quy hoạch động:

+ Các bài toán tối ưu: như tìm xâu con chung dài nhất, bài toán ba lô, tìm đường đi ngắn nhất, bài toán Ôtômat với số phép biến đổi ít nhất, …

+ Các bài toán có công thức truy hồi.

Hạn chế của phương pháp quy hoạch động:

Phương pháp quy hoạch động không đem lại hiệu quả trong các trường hợp sau:

+ Sự kết hợp lời giải của các bài toán con chưa chắc cho ta lời giải của bài toán lớn.

+ Số lượng các bài toán con cần giải quyết và lưu trữ kết quả có thể rất lớn, không thể chấp nhận được.

+ Không tìm được công thức truy hồi.

II. CẤU TRÚC CHUNG CỦA CHƯƠNG TRÌNH CHÍNH:

```
BEGIN {Chương trình chính}
    Chuẩn bị: đọc dữ liệu và khởi gán một số giá trị ban đầu;
    Tạo bảng;
    Tra bảng và ghi kết quả;
END.
```

III. CÁC BƯỚC ĐỂ GIẢI BÀI TOÁN QUY HOẠCH ĐỘNG:

1. Mô tả bài toán dưới dạng các bài toán con;

2. Tính nghiệm tối ưu của bài toán con trong trường hợp riêng đơn giản nhất;

3. Tìm các công thức đệ quy biểu diễn nghiệm tối ưu của bài toán lớn thông qua nghiệm tối ưu của các bài toán con.

4. Tính nghiệm tối ưu từ dưới lên (bottom up) và ghi lại các nghiệm tối ưu của các bài toán vào bảng.

5. Dựa vào bảng lưu kết quả của các bài toán đã tìm được để tìm nghiệm cho bài toán ban đầu.

IV. MỘT SỐ VÍ DỤ

1. Dãy Fibonaci:

Đề bài: In ra màn hình 20 số hạng đầu của dãy Fibonaci.

Biết: $F_1 = 1$

$F_2 = 1$

$F_i = F_{i-1} + F_{i-2}$ với $i > 2$

Giải thuật:

+ Tính nghiệm của bài toán trong trường hợp riêng đơn giản nhất.

F1 = F2 = 1

+ Tìm các công thức đệ quy biểu diễn nghiệm tối ưu của bài toán lớn thông qua nghiệm tối ưu của các bài toán con.

$F_i = F_{i-1} + F_{i-2}$ với $i > 2$

Mười số hạng đầu của dãy Fibonaci được biểu diễn trong bảng sau:

i	1	2	3	4	5	6	7	8	9	10
F[i]	1	1	2	3	5	8	13	21	34	55

2. Tổ hợp chập k của n phần tử:

Đề bài: Tính các phần tử của mảng C[n, k] = C_n^k = số tổ hợp chập k của n phần tử, với $0 \le k \le n \le 20$.

Biết $C_n^0 = C_n^n = 1$

$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$

Giải thuật:

+ Tính nghiệm của bài toán trong trường hợp riêng đơn giản nhất:

```
For i := 1 To n Do
Begin
        C[0, i] := 1;
        C[i, i] := 1;
End;
```

+ Tìm các công thức đệ quy biểu diễn nghiệm tối ưu của bài toán lớn thông qua nghiệm tối ưu của các bài toán con.

```
For  i  :=  2  To  n  Do
```

For j := 1 To i-1 Do C[i, j] := C[i-1,j-1] + C[i-1,j];

n\k	0	1	2	3	4	5
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

3. Dãy con không giảm liên tục dài nhất

Đề bài: Cho dãy số nguyên A_1, A_2, ... , A_N. Hãy tìm dãy con B gồm một số phần tử liên tục trong A không giảm dài nhất.

Ví dụ:

Input

9

3 2 1 4 5 8 9 6 7

Output

1 4 5 8 9

Giải thuật:

+ Gọi T[i] (i=1..N) là độ dài lớn nhất của dãy con không giảm liên tục khi có i phần tử từ 1 đến i trong A để chọn, trong đó phần tử thứ i phải được chọn.

+ Trường hợp đơn giản nhất giả sử dãy A chỉ có 1 phần tử thì T[1]=1;

+ Công thức đệ quy:

Với i ≥ 2 ta có:

Nếu A[i] ≥ A[i-1] thì T[i] = T[i-1] + 1, ngược lại T[i] = 1.

For i:=2 to N do

If A[i] >= a[i-1] then T[i]:=T[i-1] + 1 Else T[i]:=1;

+ Nghiệm của bài toán:

Tìm v là vị trí phần tử lớn nhất trong mảng T

v:=1;

For i:=1 to N do if T[i] ≥ T[v] then v:=i;

Như vậy dãy con tìm được sẽ có T[v] phần tử và phần tử cuối cùng là phần tử thứ v trong A. Cho nên dãy con tìm được sẽ là dãy gồm các phần tử từ v – T[v] + 1 đến phần tử thứ v của dãy A.

In dãy con tìm được

For i:= v-T[v]+1 to v do write(A[i], ' ');

4. Tìm dãy con không giảm dài nhất:

Đề bài: Cho một dãy n số nguyên. Hãy loại bỏ khỏi dãy một số phần tử để được một dãy con không giảm dài nhất. In ra dãy con đó.

Ví dụ:

Input:

10

2 6 -7 5 8 1 -3 5 15 9

Kết quả tìm được dãy con không giảm dài nhất có 4 phần tử:

-7 -3 5 9

Giải thuật:

+ Tổ chức dữ liệu:

Gọi A là dãy ban đầu.

Gọi B[i] là số phần tử của dãy con dài nhất trong dãy có i phần tử đầu tiên A[1] .. A[i] và A[i] được chọn làm phần tử cuối. (i [1, n])

C là dãy con không giảm dài nhất tìm được.

Truoc[i] là chỉ số của phần tử trước phần tử i (các phần tử giữ lại C).

+ Giải thuật tạo bảng: (Tính mảng B và mảng Truoc)

Trường hợp đơn giản nhất: dãy chỉ có 1 phần tử, thì B[1] := 1;

For i = 2 to n Do

◆ Với mọi (j < i) và (A[j] <= A[i]), tìm B[j] lớn nhất (gọi là BMax).

◆ B[i] := Bmax + 1;

◆ Trước[i] := j; {j là chỉ số ứng với BMax tìm được}

Trong ví dụ trên ta có bảng sau:

i	1	2	3	4	5	6	7	8	9	10
A[i]	2	6	-7	5	8	1	-3	5	15	9
B[i]	1	2	1	2	3	2	2	3	4	4
Truoc[i]	0	1	0	3	4	3	3	7	8	8

Dãy con không giảm dài nhất có 4 phần tử: -7 -3 5 9

```
Procedure  TaoBang;
Var i, j, BMax, chiSo :byte;
Begin
        B[1]  :=  1; truoc[1]:=0;
        For i := 2 to n do
        begin
                BMax :=  0; chiso:=0;
                For  j  :=  i-1  DownTo  1  Do
                If (A[j] <= A[i]) and (B[j] > BMax) then
                begin
                        BMax := B[j];
                        chiSo := j;
                end;
                B[i] := BMax + 1; Truoc[i] := chiSo;
        end;
End;
```

+ Tra bảng: để tìm các phần tử của dãy C:

Tìm phần tử lớn nhất của mảng B. (ứng với chỉ số ChiSoMax). Phần tử lớn nhất của mảng B chính là số phần tử của dãy C.

A[ChiSoMax] là phần tử cuối của dãy C. Dựa vào mảng Truoc, ta tìm các phần tử còn lại trong dãy C: tìm ngược từ cuối dãy lên đầu dãy.

```
Procedure  TraBang;
Var chiSo, ChiSoMax, i : byte;
Begin
        ChiSoMax  :=  n;
        for  i:=  n-1  downto  1  do
                if B[i] > B[ChiSoMax] then ChiSoMax := i;
        chiSo  :=  ChiSoMax;
        for i := B[ChiSoMax] downto 1 do
        begin
                C[i]:= A[chiSo];
                chiSo := Truoc[chiSo];
        end;
End;
```

5. Bài toán balô 1:

Đề bài: Cho n món hàng (n ≤ 50). Món thứ i có khối lượng là A[i] (số nguyên). Cần chọn những món hàng nào để bỏ vào một ba lô sao tổng khối lượng của các món hàng đã chọn là lớn nhất nhưng không vượt quá khối lượng W cho trước. (W ≤ 100). Mỗi món chỉ chọn 1 hoặc không chọn.

Input:

n W
A[1] A[2] … A[n]

Ví dụ:

4 10
5 2 4 3

OutPut:

Tổng khối lượng của các món hàng bỏ vào ba lô.

Khối lượng của các món hàng đã chọn.

Trong ví dụ trên:

Tổng khối lượng của các món hàng bỏ vào ba lô là 10

Khối lượng các món hàng được chọn: 5 2 3

Hướng giải:

+ Tổ chức dữ liệu:

Fx[k, v] là tổng khối lượng của các món hàng bỏ vào ba lô khi có k món hàng đầu tiên để chọn và khối lượng tối đa của ba lô là v.

Với k ϵ [1, n], v ϵ [1, W].

Nói cách khác: Khi có k món để chọn, Fx[k, v] là khối lượng tối ưu khi khối lượng tối đa của ba lô là v.

Khối lượng tối ưu luôn nhỏ hơn hoặc bằng khối lượng tối đa:

Fx[k,v] ≤ v.

Ví dụ: Fx[4, 10] = 8 Nghĩa là trong trường hợp tối ưu, tổng khối lượng của các món hàng được chọn là 8, khi có 4 món đầu tiên để chọn (từ món thứ 1 đến món thứ 4) và khối lượng tối đa của ba lô là 10. Không nhất thiết cả 4 món đều được chọn.

+ Giải thuật tạo bảng:

Trường hợp đơn giản chỉ có 1 món để chọn: Ta tính Fx[1, v] với mọi v.

Nếu có thể chọn (nghĩa là khối lượng tối đa của ba lô >= khối lượng của các món hàng thứ 1), thì chọn: Fx[1, v] := A[1];

Ngược lại (v < A[1]), không thể chọn, nghĩa là Fx[1, v] := 0;

* Giả sử ta đã tính được Fx[k–1 , v] đến dòng k–1, ∀ v ∈ [1, W]. Khi có thêm món thứ k để chọn, ta cần tính Fx[k , v] ở dòng k, ∀ v ∈ [1,W].

Nếu có thể chọn món hàng thứ k (v >= A[k]), thì có 2 trường hợp:

– Trường hợp 1: Nếu chọn thêm món thứ k bỏ vào ba lô thì:

Fx[k, v] := Fx[k–1, u] + A[k]; Với u là khối lượng còn lại sau khi chọn món thứ k. u = v – A[k]

– Trường hợp 2: Ngược lại, không chọn món thứ k, thì Fx[k, v] := Fx[k–1, v];

Trong 2 trường hợp trên ta chọn trường hợp nào có Fx[k, v] lớn hơn. Ngược lại (v < A[k]), thì không thể chọn, nghĩa là Fx[k, v] := Fx[k–1, v];

+ Công thức đệ quy là:

If v >= A[k] Then Fx[k,v] := Max(Fx[k-1, v - A[k]] + A[k] , Fx[k-1,v]) Else Fx[k,v] := Fx[k-1, v];

Dưới đây là bảng Fx[k,v] tính được trong ví dụ trên:

A	k \ v	1	2	3	4	5	6	7	8	9	10
5	1	0	0	0	0	5	5	5	5	5	5
2	2	0	2	2	2	5	5	7	7	7	7
4	3	0	2	2	4	5	6	7	7	9	9
3	4	0	2	3	4	5	6	7	8	9	10

```
Procedure TaoBang;
Var k ,v : integer;
Begin
        For  v:=1  to  W  do
        If v >= A[1] then Fx[1, v] := A[1] Else Fx[1, v] := 0;
        For k:= 2 to n do for v:=1 to W do
                If  v  >=  A[k]  then
                Fx[k,v]:= Max(Fx[k-1,v-A[k]]+ A[k], Fx[k-1,v])
                Else Fx[k,v]:=Fx[k-1,v];
End;
```

+ Giải thuật tra bảng để tìm các món hàng được chọn:

Chú ý: Nếu Fx[k, v] = Fx[k–1, v] thì món thứ k không được chọn.

Fx[n, W] là tổng khối lượng tối ưu của các món hàng bỏ vào ba lô.

Bước 1: Bắt đầu từ k = n, v = W.

Bước 2: Tìm trong cột v, ngược từ dưới lên, ta tìm dòng k sao cho Fx[k,v] > Fx[k–1, v].

Đánh dấu món thứ k được chọn:

Chọn[k] := true;

Bước 3: v := Fx[k, v] – A[k].

Nếu v > 0 thì thực hiện bước 2, ngược lại thực hiện bước 4

Bước 4: Dựa vào mảng Chọn để in ra các món hàng được chọn.

```
Procedure TraBang;
var k, v: Integer;
Begin
        k := n; v := w;
        FillChar(chon,SizeOf(chon),false);
        Repeat
                While Fx[k,v] = Fx[k-1,v] do Dec(k);
                chon[k]:= True;
                v := Fx[k,v] - A[k];
        Until v = 0;
        For  k  :=  1  to  n  do
                If chon[k] then Write(A[k]:5); Writeln;
End;
```

6. Bài toán ba lô 2

Đề bài: Cho n món hàng (n ≤ 50). Món thứ i có khối lượng là A[i] và giá trị C[i] (số nguyên). Cần chọn những món hàng nào để bỏ vào một ba lô sao tổng giá trị của các món hàng đã chọn là lớn nhất nhưng tổng khối lượng của chúng không vượt quá khối lượng W cho trước (W ≤ 100). Mỗi món chỉ chọn 1 hoặc không chọn.

Input:

n W

A[1] C[1]

A[2] C[2]

…

A[n] C[n]

Ví dụ:

5 13

3 4

4 5

5 6

2 3

1 1

OutPut:

Tổng giá trị của các món hàng bỏ vào ba lô.

Khối lượng và giá trị của các món hàng đã chọn.

Trong ví dụ trên:

Tổng giá trị của các món hàng bỏ vào ba : 16

Các món được chọn:

1(3, 4) 2(4, 5) 3(5, 6) 5(1, 1)

Hướng dẫn giải:

Tương tự bài ba lô 1, nhưng Fx[k, v] là giá trị lớn nhất của ba lô khi có k món hàng đầu tiên để chọn và khối lượng tối đa của ba lô là v.

Công thức đệ quy là:

If v >= A[k] Then

Fx[k,v]:= Max(Fx[k-1, v-A[k]] + C[k], Fx[k-1,v])

Else

Fx[k,v]:=Fx[k–1,v];

Chú ý: chỉ khác bài balô 1 ở chỗ dùng C[k] thay cho A[k]

Dưới đây là bảng Fx[k,v] tính được trong ví dụ trên:

k/v	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	4	4	4	4	4	4	4	4	4	4	4
2	0	0	4	5	5	5	9	9	9	9	9	9	9
3	0	0	4	5	6	6	9	10	11	11	11	15	15
4	0	3	4	5	7	8	9	10	12	13	14	15	15
5	1	3	4	5	7	8	9	10	12	13	14	15	16

7. Bài toán ba lô 3

Đề bài: Cho n loại hàng (n ≤ 50). Mỗi món hàng thuộc loại thứ i có khối lượng là A[i] và giá trị C[i] (số nguyên). Số lượng các món hàng của mỗi loại không hạn chế. Cần chọn những món hàng của những loại hàng nào để bỏ vào một ba lô sao tổng giá trị của các món hàng đã chọn là lớn nhất nhưng tổng khối lượng của chúng không lượt quá khối lượng W cho trước (W ≤ 100). Mỗi loại hàng có thể hoặc không chọn món nào, hoặc chọn 1 món, hoặc chọn nhiều món.

Input:

n W

A[1] C[1]

A[2] C[2]

…

A[n] C[n]

Ví dụ:

5 13

3 4

4 5

5 6

2 3

1 1

OutPut:

Tổng giá trị của các món hàng bỏ vào ba lô.

Số lượng của các loại hàng đã chọn.

Trong ví dụ trên:

Tổng giá trị của các món hàng bỏ vào ba lô: 19

Các món được chọn:

Chọn 1 món hàng loại 1, mỗi món có khối lượng là 3 và giá trị là 4

Chọn 5 món hàng loại 4, mỗi món có khối lượng là 2 và giá trị là 3

Hướng dẫn giải:

+ Tổ chức dữ liệu:

Fx[k, v] là tổng giá trị của các món hàng bỏ vào ba lô khi có k loại hàng

đầu tiên để chọn và khối lượng tối đa của ba lô là v.

Với k [1, n], v ∈ [1, W].

X[k, v] là số lượng các món hàng loại k được chọn khi khối lượng tối đa của

ba lô là v.

+ Giải thuật tạo bảng:

* Trường hợp đơn giản chỉ có 1 món để chọn: Ta tính Fx[1, v] với mọi v:

X[1, v] = v div A[1]

Fx[1, v] = X[1, v] * C[1]

* Giả sử ta đã tính được Fx[k–1 , v] đến dòng k–1, với mọi v ∈ [1,W].

Khi có thêm loại thứ k để chọn, ta cần tính Fx[k , v] ở dòng k, với mọi v ∈ [1,W]

Nếu ta chọn xk món hàng loại k, thì khối lượng còn lại của ba lô dành cho các loại hàng từ loại 1 đến loại k – 1 là: u = v – xk * A[k]

Khi đó giá trị của ba lô là: Fx[k, v]= Fx[k–1,u] + xk * C[k]

Với xk thay đổi từ 0 đến yk, ta chọn giá trị lớn nhất và lưu vào Fx[k, v].

Trong đó yk = v div A[k] là số lượng lớn nhất các món hàng loại k có thể được chọn bỏ vào ba lô, khi khối lượng tối đa của ba lô là v.

Tóm lại: công thức đệ quy là:

Fx[k,v] = Max(Fx[k-1, v – xk * A[k]] + xk * C[k])

Max xét với xk thay đổi từ 0 đến v div A[k], và v – xk * A[k] > 0

Dưới đây là bảng Fx[k,v] và X[k, v] tính được trong ví dụ trên. Bảng màu xám là X[k, v]:

k/v	1		2		3		4		5		6		7		8		9		10		11		12		13	
1	0	0	0	0	4	1	4	1	4	1	8	2	8	2	8	2	12	3	12	3	12	3	16	4	16	4
2	0	0	0	0	4	0	4	0	5	1	8	0	9	1	9	1	12	0	13	1	14	2	16	0	17	1
3	0	0	0	0	4	0	4	0	5	0	8	0	9	0	10	1	12	0	13	0	14	0	16	0	17	0
4	0	0	0	0	4	0	4	0	7	1	8	0	10	2	11	1	13	3	14	2	16	4	17	3	19	5
5	0	0	1	1	4	0	5	1	7	0	8	0	10	0	11	0	13	0	14	0	16	0	17	0	19	0

```
Procedure TaoBang;
Var xk, yk, k: Byte;
FMax, XMax, v : Word;
Begin
    For v:= 1 To W Do
    begin
        X[1, v] := v div A[1];
        F[1, v] := X[1, v] * C[1];
    end;
    For k:= 2 To n Do
    For v:= 1 To W Do
    begin
        FMax := F[k-1, v] ;
        XMax := 0;
        yk := v div A[k];
        For xk:= 1 To yk Do
        If (v - xk * A[k] > 0) and F[k-1, v - xk * A[k]] + xk * C[k]
    > FMax) Then
        begin
            FMax := F[k-1, v - xk * A[k]] + xk * C[k];
            XMax:= xk;
        end;
        F[k, v] := FMax;
        X[k, v] := XMax;
    end;
End;
```

Giải thuật tra bảng:

Fx[n, W] là giá trị lớn nhất của ba lô.

Bắt đầu từ X[n, W] là số món hàng loại k được chọn.

Tính v = W – X[n, W]* A[n].

Tìm đến ô [n – 1, v] ta tìm được X[n – 1, v]. Cứ tiếp tục ta tìm được X[1, v].

Chú ý: khi tra bảng, ta không dùng mảng Fx[k, v], nên ta có thể cải tiến: dùng 2 mảng một chiều thay cho mảng hai chiều Fx.

8. Bài toán xâu con chung dài nhất

Đề bài: Cho hai *xâu* kí tự A và B. Tìm xâu kí tự C có nhiều kí tự nhất, với C vừa là xâu con của xâu A, vừa là xâu con của xâu B. (Xâu con là xâu kí tự có được khi bỏ bớt một số kí tự trong xâu cha).

Dư liệu: Vào từ tập tin văn bản **XauChung.inp** gồm hai dòng, mỗi dòng là một xâu kí tự.

Kết quả: Đưa ra tập tin văn bản **XauChung.out** gồm 2 dòng:

• Dòng đầu là độ dài của xâu con chung dài nhất.
• Dòng thứ hai là xâu con chung C.

Ví dụ

XauChung.INP
CEACEEC AECECA

XauChung.OUT
4 AEEC

Hướng dẫn:

+ Tổ chức dữ liệu

Mảng hai chiều L[0..Max, 0..Max] để lưu kết quả các bài toán con.

+ Tạo bảng

Gọi L[i,j] là độ dài lớn nhất của dãy con chung của hai dãy khi dãy A có i phần tử và dãy B có j phần tử để chọn A1...Ai và B1..Bj. Với i<=m và j<=n (m,n là độ dài hai xâu).

+ **Trường hợp đơn giản**: nếu i=0 hoặc j=0 (một trong hai xâu rỗng) thì L[i,j] =0.

+ Công thức đệ quy

Nếu (i>0) and (j>0) and (Ai<>Bj)

thì có L[i,j] =Max(L[i-1,j], L[i,j-1])

Nếu (i>0) and (j>0) and (Ai=Bj) thì L[i,j] = 1+ L[i-1,j-1]

+ Tạo bảng

```
Procedure Taobang;
Var i,j:byte;
Begin
        Fillchar(L,sizeof(L),0);
        For i:=1 to m do
                For j:=1 to n do
                        If A[i]=B[j] then L[i,j]:= 1 + L[i-1,j-1]
                        Else L[i,j]:=Max(L[i-1,j],L[i,j-1]);
End;
```

+ Tra bảng

Xâu chung C:=''

Xuất phát từ ô L[n,m] hướng về ô L[0,0].

Nếu A[i]=B[j] chọn A[i] hoặc B[j] đưa vào đầu dãy C rồi lùi về ô L[i-1,j-1]

Nếu Ai<>Bj thì:

Nếu L[i-1,j] > L[i,j-1] lùi về L[i-1,j]

Nếu L[i-1,j] <= L[i,j-1] thì lùi về L[i,j-1]

```
Procedure Trabang;
Var i, j:byte;
Begin
        C:='';
        I:=m; j:=n;
        Repeat
                If A[i]=B[j] then
                Begin
                        C:= A[i]+C;
                        Dec(i); Dec(j);
                End
                Else if L[i,j-1] > L[i-1,j] then j:=j-1 else i:=i-1
        Until (i=0) or (j=0);
```

End;

9. Bài toán chia kẹo

Đề bài: Cho n gói kẹo (n ≤ 50). Gói thứ i có A[i] viên kẹo. Cần chia các gói kẹo này cho 2 em bé sao cho tổng số viên kẹo mỗi em nhận được chênh lệch ít nhất. Mỗi em nhận nguyên gói. Không mở gói kẹo ra để chia lại. Hãy liệt kê số kẹo trong các gói kẹo mỗi em nhận được.

Input:

n

A[1] A[2] … A[n]

Output: Số kẹo trong các gói kẹo mỗi em nhận được, và tổng số kẹo mỗi em nhận được.

Hướng dẫn giải:

Gọi S là tổng số viện kẹo S := A[1] + A[2] + … + A[n];

S2 là nửa tổng số kẹo: S2 := S div 2;

Cho em bé thứ nhất chọn trước những gói kẹo sao cho tổng số viên kẹo mà em nhận được là lớn nhất nhưng không vượt quá số kẹo S2.

Gói kẹo nào em bé thứ nhất không chọn thì em bé thứ hai chọn.

Bài toán được đưa về **Bài toán ba lô 1**(đã hướng dẫn ở trên).

10. Bài toán đổi tiền

Đề bài: Cho n loại tờ giấy bạc. Tờ giấy bạc thứ i có mệnh giá A[i]. Số tờ mỗi loại không giới hạn. Cần chi trả cho khách hàng số tiền M đồng. Hãy cho biết mỗi loại tiền cần bao nhiêu tờ sao cho tổng số tờ là ít nhất. Nếu không đổi được, thì thông báo "KHONG DOI DUOC". N < 50; A[i] < 256; M < 10000.

Input:

n M

A[1] A[2] … A[n]

Ví dụ:

3 18

3 10 12

Output: Tổng số tờ phải trả. Số tờ mỗi loại.

Hướng dẫn giải: Tương tự **Bài toán ba lô 3**

Gọi Fx[i, j] là số tờ ít nhất được dùng để trả số tiền j đồng khi có i loại tiền từ loại 1 đến loại i. Với i = 1 .. n; j = 1 .. M.

X[i, j] là số tờ giấy bạc loại thứ i được dùng chi trả số tiền j đồng.

* Trường hợp đơn giản chỉ có 1 loại tiền để chọn: Ta tính Fx[1, j] với mọi j

F[1,j] = j div A[1] nếu j mod A[1] = 0

F[1,j] = ∞ nếu j mod A[1] $\neq$ 0 (không đổi được)

* Giả sử ta đã tính được Fx[i–1 , j] đến dòng i–1, với mọi j $\in$ [1,M]. Khi có thêm loại tiền thứ i để chọn, ta cần tính Fx[i , j] ở dòng i, với mọi j $\in$ [1, M].

Nếu ta chọn k tờ loại i, thì số tiền còn lại dành cho các loại tiền khác từ loại 1 đến loại i – 1 là: u = j – k * A[k]

Khi đó tổng số tờ là: Fx[i, j]= Fx[i–1,u] + k

Với k thay đổi từ 0 đến kMax, ta chọn giá trị nhỏ nhất và lưu vào Fx[i, j].

Trong đó kMax = j div A[k] là số tờ nhiều nhất của loại tiền i để đổi số tiền j.

Tóm lại: công thức đệ quy là:

Fx[i,j] = Min(Fx[i-1, j – k * A[i]] + k)

Min xét với k thay đổi từ 0 đến j div A[i], và j – k * A[i] > 0

+

V.Bài tập

Bài 1 : Túi ba gang

Trong truyện cổ tích "Cây Khế" ta đã biết rằng chim thần chở người anh với một cái túi ba gang đến hòn đảo đầy vàng bạc châu báu. Người em băn khoăn không biết chọn đồ vật nào cho vào túi vì chỉ có một cái túi ba gang..

Giả sử rằng trên hòn đảo kia có N đồ vật khác nhau, đồ vật thứ i có giá trị là a_i và có thể tích là b_i. Cũng giả sử rằng cái túi mà người em mang đi chỉ có thể tích là M. Bạn hãy giúp người em chọn ra trong N đồ vật trên một số đồ vật sao cho tổng thể tích của các đồ vật được chọn không vượt quá M và tổng giá trị các đồ vật được chọn là lớn nhất.

Input: Cho trong file văn bản CAYKHE.INP

- Dòng đầu tiên ghi hai số N, M (N,M≤100)
- N dòng tiếp theo, dòng thứ i ghi hai số a_i và b_i lần lượt là giá trị và thể tích của đồ vật thứ i (a_i, b_i≤100)

Output: Ghi ra file văn bản CAYKHE.OUT

- Dòng đầu tiên ghi tổng giá trị lớn nhất có thể cho vào trong túi
- Dòng thứ hai ghi số hiệu các đồ vật được cho vào trong túi. Đầu tiên ghi K là số lượng đồ vật được chọn, tiếp theo là K số thể hiện số hiệu các đồ vật được chọn.

Ví dụ:

CAYKHE.INP	CAYKHE.OUT
5 10 20 3 19 1 30 7 24 3 15 6	63 3 1 2 4

Bài 2: Kinh doanh bất động sản.

Tại thành phố Silicon, Một người nọ được thừa kế một khoản tiền N ngàn USD, người đó quyết định đầu tư vào việc kinh doanh bất động sản bằng cách mua các mảnh đất hình vuông có kích thước là các số nguyên, biết rằng mỗi mét vuông đất có giá trị 1 ngàn USD. Hãy chỉ cách cho người nọ mua đất sao cho tổng số tiền mua đất đúng bằng N ngàn USD và số mảnh đất mua được càng ít càng tốt.
Dữ liệu: Vào từ file văn bản MUADAT.INP gồm một số nguyên dương duy nhất N có giá trị không vượt quá 60000

Kết quả: Ghi ra file văn bản MUADAT.OUT một dãy số nguyên dương xếp theo thứ tự giảm dần là kích thước các mảnh đất mua được

Ví dụ:

MUADAT.INP	MUADAT.OUT
30	4 3 2 1

Bài 3: Xây tháp

Có N khối đá hình hộp chữ nhật. Người ta muốn xây một cái tháp bằng cách chồng các khối đá này lên nhau. Để đảm bảo an toàn, các khối đá được đặt theo nguyên tắc:

+ Chiều cao của mỗi khối là kích thước nhỏ nhất trong ba kích thước
+ Các mép của các khối đươc đặt song song với nhau sao cho không có phần nào của khối nằm trên bị chìa ra ngoài so với khối nằm dưới.

Hãy tìm phương án xây dựng để tháp đạt được độ cao nhất.

Dữ liệu vào: Cho trong file TOWN.INP:

- Dòng đầu tiên là số N
- N dòng tiếp, mỗi dòng ghi 3 số nguyên dương là kích thước của một khối đá. Các khối đá đươc đánh số từ 1 theo trình tự xuất hiện trong file.

Kết quả: ghi ra file TOWN.OUT:

- Dòng thứ nhất ghi số M là số lượng khối đá dùng để xây tháp
- M dòng tiếp theo ghi các khối xếp từ đáy tháp lên đỉnh tháp, mỗi dòng gồm 4 số theo thứ tự K a b c, trong đó K là số hiệu khối đá, a là kích thước chọn làm đáy nhỏ, b là kích thước chọn làm đáy lớn, c là kích thước chọn làm chiều cao.

Các số trên một dòng trong các file được ghi cách nhau ít nhất một dấu cách. Giới hạn số khối đá không quá 5000 và các kích thước của các khối đá không quá 255.

Ví dụ:

TOWN.INP	TOWN.OUT
9	4
7 5 5	1 5 7 5
4 4 8	9 5 5 5
1 1 5	5 5 5 1
4 2 2	4 2 4 2
5 1 5	
4 2 7	
2 9 2	
1 3 3	
5 5 5	

Bài 4: Vòng quanh thế giới *(Đề thi học sinh giỏi Toàn quốc 2003)*

Trên tuyến đường của xe du lịch vòng quanh thế giới xuất phát từ bến xe x có N khách sạn đánh số từ 1 đến N theo thứ tự xuất hiện trên tuyến đường, trong đó 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 khỏe cho hành khách, theo tính toán của các nhà chuyên môn, sau khi đã chạy được P km xe nên dừng lại cho khách nghỉ ngơi 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 phạt là $(Q-P)^2$.

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 ít nhất.

Dữ liệu vào từ file văn bản TOURISM.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[n] (hai số liên tiếp trên dòng được ghi cách nhau bởi dấu cách; A[i]≤2000000, i=1,2,...,N)

Kết quả ghi ra file văn bản TOURISM.OUT:

- Dòng đầu ghi Z là lượng phạt mà lái xe phải trả
- Dòng thứ 2 ghi k là tổng số khách sạn mà lái xe cần dừng lại cho khách nghỉ
- Dòng thứ ba chứa chỉ số của k khách sạn mà xe dừng lại cho khách nghỉ (trong đó nhất thiết phải bao gồm cả chỉ số của khách sạn thứ N)

Ví dụ:

TOURISM.INP	TOURISM.OUT
4	500
300	2
250 310 550 590	2 4

Bài 5: Duyệt điểm

Xét lưới ô vuông tạo thành từ n*n dường, các đường của lưới được đánh số từ 1 đến n từ trái qua phải và từ trên xuống dưới (1≤n≤20000). Ở mỗi hàng thứ i, người ta cho đoạn thẳng xác định bởi hai điểm l_i và r_i ($1≤l_i≤r_i≤n$, i=1…n).

Yêu cầu: Xác định độ dài của đường đi ngắn nhất dọc theo các cạnh của lưới từ điểm (1,1) đến điểm (n,n) và thoả mãn các điều kiện:

- Chỉ đi sang phải, sang trái hoặc xuống dưới
- Đi qua tất cả các điểm thuộc các đoạn thẳng đã cho

Dữ liệu: Vào từ file văn bản VISIT.INP:

- Dòng đầu tiên chứa số nguyên dương n
- Dòng thứ i trong n dòng sau chứa hai số nguyên dương l_i, r_i

Kết quả: Đưa ra file văn bản VISIT.OUT một số nguyên duy nhất là độ dài của đường đi ngắn nhất tìm được.

Ví dụ:

VISIT.INP	VISIT.OUT
6 2 6 3 4 1 3 1 2 3 6 4 5	24

Bài 6: Những vị khách sộp vào nhà hàng

Một nhà hàng bắt đầu mở cửa tại thời điểm 0 và đóng cửa tại thời điểm $T=2.10^9$. Tại cửa ra vào nhà hàng có treo một bảng hiện thị số. Tại thời điểm 0, số trên bảng là 0 và cứ sau 1s số trên bảng giữ nguyên giá trị hoặc tăng, giảm một đơn vị. Bảng chỉ hiện thị được các số không âm.

Có N vị khách sộp đi qua nhà hàng, vị khách sộp thứ i đi qua nhà hàng tại thời điểm T_i và sở thích của ông ta là số S_i. Nếu như ở thời điểm ông ta đi qua nhà hàng biển số trước cửa nhà hàng hiện đúng số ông ta thích thì ông ta sẽ vào và tiêu một số tiền là P_i.

Yêu cầu: Hãy giúp nhà hàng điều khiển bảng số sao cho tổng số tiền mà các vị khách sộp vào nhà hàng là lớn nhất.

Input: Cho trong file WELCOME.INP

- Dòng đầu tiên ghi số nguyên dương N ($N \leq 200$)
- Trong N dòng tiếp theo, dòng thứ i ghi thông tin về vị khách thứ i gồm ba số nguyên dương T_i, S_i và P_i.

Output: Ghi ra file WELCOME.OUT một số nguyên duy nhất là tổng số tiền lớn nhất mà nhà hàng nhận được.

Ví dụ:

WELCOME.INP	WELCOME.OUT
3 2 1 3 3 2 4 1 3 10	7

Bài 7: Di chuyển từ Tây sang Đông

Cho hình chữ nhật M x 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 vuông chứa một số nguyên. Có thể di chuyển từ một ô sang một ô khác thuộc cột bên phải cùng dòng hoặc lệch một dòng. Tìm cách di chuyển từ cột 1 sang cột N sao cho tổng các số của các ô đi qua là nhỏ nhất.

Input: Từ file văn bản WTOE.INP

- Dòng đầu tiên chứa hai số nguyên M, N (1≤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

Output: Ghi ra file văn bản WTOE.OUT

- Dòng thứ nhất ghi tổng các số trong các ô đi qua
- Dòng thứ 2 ghi N số lần lượt là chỉ số hàng của các ô trên hành trình bắt đầu từ cột 1 đến cột N

Ví dụ:

WTOE.INP
2 2 2 6 3 5

WTOE.OUT
9 2 1

Bài 8: Dãy con đối xứng dài nhất (daydx.pas)

Dãy số có A_1, A_2, …, A_N được gọi là đối xứng nếu các cặp số ở các vị trí i và N-i+1 bằng nhau (với i=1..N). Cho trước một dãy số có N phần tử, mỗi phần tử là số nguyên. Hãy tìm cách loại bỏ một số phần tử trong dãy để dãy thu được tạo thành một dãy đối xứng dài nhất.

Dữ liệu vào: File văn bản DAYDX.INP có cấu trúc như sau:

+ Dòng 1: Số nguyên N (2≤N≤5000);

+ Dòng thứ 2 ghi N số nguyên là các số hạng trong dãy có giá trị tuyệt đối ≤1000, mỗi số cách nhau một dấu cách.

Dữ liệu ra: File văn bản DAYDX.OUT với yêu cầu nhhư sau:

+ Dòng đầu ghi số nguyên M là số các các số hạng của dãy đối xứng tìm được;

+ Dòng thứ 2 ghi M số hạng của dãy tìm được, mỗi số cách nhau một dấu cách.

Ví dụ:

DAYDX.INP	DAYDX.OUT
13 1 3 2 3 1 5 2 3 4 1 4 3 2	7 2 3 4 1 4 3 2

Bài 9: Nối mạng máy tính

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: vào từ file văn bản CABLE.INP:

- 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 file văn bản CABLE.OUT độ dài của cáp nối cần sử dụng.

Ví dụ:

CABLE.INP	CABLE.OUT
6 2 2 3 2 2	7