Chuyên đề Tin học

PHƯƠNG PHÁP QUY HOẠCH ĐỘNG

Biên soạn: Trần Quang Quá
Giáo viên trường PTTH chuyên Lương Thế Vinh, Biên Hoà
Tháng 4 năm 2002

1. GIỚI THIỆU:

Phương pháp quy hoạch động (dynamic programming) là một kĩ thuật được áp dụng để giải nhiều lớp bài toán, đặc biệt là các bài toán tối ưu.

Phương pháp quy hoạch động dùng kĩ 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. Trong quá trình

"đi từ dưới lên" chúng ta sẽ sử dụng một bảng để lưu giữ lời giải của các bài toán con đã giải, không cần quan tâm đến nó được sử dụng ở đâu sau này.

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

1.1. Ưu điểm của phương pháp quy hoạch động:

- Chương trình chạy nhanh.

1.2. Phạm vi áp dụng của phương pháp quy hoạch động:

- Các bài toán tối ưu: như tìm xâu con chung dài nhất, bài toán balô, tìm đường đi ngắn nhất, bài toán Ôtômat với số phép biến đổi ít nhất, ...
- Các bài toán có công thức truy hồi.

1.3. 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.

2. CẤU TRÚC CHUNG CỦA CHƯƠNG TRÌNH CHÍNH:

BEGIN {Chương trình chính}

Chuẩn bị: đọc dữ liệu và khởi gán một số giá trị;

Tạo bảng;

Tra bảng và in kết quả;

END.

3. PHƯƠNG PHÁP QUY HOẠCH ĐỘNG:

1) Tính nghiệm tối ưu của bài toán trong trường hợp riêng đơn giản nhất.
2) 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.
3) Tính nghiệm tối ưu từ dưới lên (bottom up) và ghi lại các nghiệm tối ưu của các bài toán con đã tính để sử dụng sau này.

4. VÍ DỤ:

4.1. Dãy Fibonaci:

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

Biết: $F_1 = 1$

$F_2 = 1$

$F_3 = 2$ $\qquad F_i = F_{i-1} + F_{i-2}$ với $i > 2$

$F_4 = 3$

Giải thuật:

Giải thuật:

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

$F_1 = F_2 = 1$

2) 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:

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

4.2. 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 $\qquad C_n^0 = C_n^n = 1$

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

Giải thuật:

1) 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;
```

2) 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) Có thể cải tiến: dùng 2 mảng một chiều thay cho 1 mảng hai chiều.

4.3. *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:

1) 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).

2) Giải thuật tạo bảng: (Tính mảng B và mảng Trước)

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

For i từ 2 đến n Do

- Xét 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:

i	1	2	3	4	5	6	7	8	9	10
Dãy 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	1	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;
  For i := 2 to n do
    begin
      BMax := 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;
```

Có thể cải tiến không cần dùng biến BMax và chiSo

3) Tra bảng: để tìm các phần tử của dãy C:

a) 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.

b) A[ChiSoMax] là phần tử cuối của dãy C. Nhờ vào mảng Trước, 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;
```

4.4. Bài toán balô 1:

Đề bài: Cho n món hàng ($n \le 50$). 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. ($W \le 100$). 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:

1) 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 $k \in [1, n]$, $v \in [1, W]$.

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] $\leq$ 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.

2) 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 làkhối 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, với mọi v $\in$ [1, W].

Khi có thêm món thứ k để chọn, ta cần tính Fx[k , v] ở dòng k, với mọi 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];

Tóm lại: 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:

k v	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	5	5	5	5	5	5
2	0	2	2	2	5	5	7	7	7	7
3	0	2	2	4	5	6	7	7	9	9
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;
```

3) 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.

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

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

```
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;
```

4.5. *Bài toán chia kẹo:*

Đề bài: Cho n gói kẹo ($n \le 50$). 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 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 ba lô 1.

4.6. *Bài toán balô 2:*

Đề bài: Cho n món hàng ($n \le 50$). 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 ($W \le 100$).

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:

Ví dụ:

```
5 13
3 4
4 5
5 6
2 3
1 1
```

<u>OutPut</u>:

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)

<u>Hướng giải</u>:

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

4.7. *Bài toán balô 3:*

Đề bài: Cho n loại hàng ($n \le 50$). 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 vượt quá khối lượng W cho trước ($W \le 100$).

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 du:

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 giải:

1) 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 $k \in [1, n]$, $v \in [1, W]$.

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

2) 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.

Input:

n W

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];
```

```
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;

3) 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.

4.8. 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 du:
```
3 18
3 10 12
```

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

Cách giải thứ nhất: Tương tự bài 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

$$Fx[1, j] = \begin{cases} j \text{ div } A[1] & \text{nếu } j \bmod A[1] = 0 \\ \infty & \text{nếu } j \bmod A[1] \neq 0 \text{ (không đổi được)} \end{cases}$$

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

<u>Cách giải thứ hai</u>:

Gọi Fx[i] là số tờ ít nhất được dùng để đổi số tiền i. Với i = 1 .. M.
Với quy ước Fx[i] = ∞ (hoặc 0) khi không đổi được.

X[i] là loại tiền *cuối cùng* được dùng đổi số tiền i. (chỉ lưu 1 loại tiền)

<u>Giải thuật tạo bảng</u>:

Xếp mệnh giá A[i] tăng dần.
Khởi gán Fx[i] = VôCực, X[i] = 0 với mọi j = 1 .. M
Gán Fx[0] = 0

Với số tiền i chạy từ 1 đến M, ta tính Fx[i] và X[i], bằng cách:

Nếu chọn loại tiền j thì số tiền còn lại là i – A[j]

Fx[i] = Min(Fx[i – A[j]] + 1)   nếu i >= A[j]

Min xét với loại tiền j chạy từ 1 đến n.

X[i] = j ứng với giá trị min của Fx[i]

Dưới đây là mảng Fx[i] và X[i] tính được trong ví dụ trên

(dùng 3 loại tiền 3 đồng, 10 đồng, 12 đồng để đổi số tiền 18 đồng)

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Fx[i]	∞	∞	∞	1	∞	∞	2	∞	∞	3	1	∞	1	2	∞	2	3	∞	3
X[i]	0	0	0	1	0	0	1	0	0	1	2	0	3	2	0	3	2	0	3

```
Procedure TaoBang;
Var i: Word;
    j: Byte;
Begin
  For i:= 1 to M Do Fx[i] := VoCuc; {VôCực = MaxInt – 1 }
  Fx[0] := 0;
  FillChar(X, SizeOf(X), 0);
  For i := 1 to M Do                 { i là số tiền }
    For j := n DownTo 1 Do           {j là loại tiền }
        If i >= A[j] Then
          If (Fx[i] > Fx[i - A[j]] + 1) Then
            Begin
              Fx[i]:= Fx[i - A[j]] + 1;
              X[i] := j;
            End;
End;
```

4.9. Phân công kĩ sư

(Đề thi tuyển sinh sau Đại học khoá 1997 Đại học Tổng hợp Tp HCM)

Một cơ sở phần mềm có n phòng máy vi tính. Cơ sở này phải tuyển chọn m kĩ sư để bảo trì máy. Sau khi tham gia ý kiến của các chuyên gia và kinh nghiệm của các đơn vị khác, người ta hiểu rằng nếu phân công i kĩ sư chuyên bảo trì tại phòng máy j thì số máy hỏng hằng năm phải thanh lí là a[i,j]. Do hạn chế về thời gian và điều kiện đi lại chỉ có thể phân công mỗi kĩ sư bảo trì tại một phòng máy. Bảng ví dụ dưới đây với m = 5 (kĩ sư) và n = 3 (phòng máy).

Số kĩ sư	phòng máy 1	phòng máy 2	phòng máy 3
0	14	25	20
1	10	19	14
2	7	16	11
3	4	14	8
4	1	12	6
5	0	11	5

Yêu cầu: Tìm ra phương án phân công mỗi phòng máy phân bao nhiêu kĩ sư sao cho tổng số máy phải thanh lí hằng năm là ít nhất.

Dữ liệu: vào từ file văn bản KiSu.inp có 2 dòng:

- dòng đầu gồm 2 số nguyên dương m, n. (m, n < 50)
- m+1 dòng tiếp theo bảng a[i, j].

Kết quả: đưa ra file văn bản KiSu.out gồm 2 dòng:

- Dòng đầu chứa tổng số máy (ít nhất) phải thanh lí hằng năm.

- Dòng thứ hai chứa n số nguyên dương là số kĩ sư được phân công bảo trì mỗi phòng máy.

Trong ví dụ trên, phân công 3 kĩ sư cho phòng máy 1, 1 kĩ sư cho phòng máy 2, và 1 kĩ sư cho phòng máy 3. Khi đó, hằng năm số máy ít nhất phải thanh lí là 37 máy.

Ví dụ:

KiSu.inp (đối với ví dụ trên)
5 3
14 25 20
10 19 14
7 16 11
4 14 8
1 12 6
0 11 5

KiSu.out
37
3 1 1

Hướng dẫn giải:

Gọi F[i, j] là số máy hư ít nhất hằng năm khi có i kĩ sư được phân công bảo trì j phòng máy đầu tiên.

4.10. Tam phân đa giác:

Cho một đa giác lồi n đỉnh. Hãy phân đa giác này thành n – 2 tam giác bằng n – 3 đường chéo, sao cho tổng của độ dài của các đường chéo này là nhỏ nhất. Các đường chéo này không cắt nhau (chỉ có thể giao nhau ở đỉnh của đa giác).

Dữ liệu: vào từ file văn bản TAMPHAN.INP có n + 1 dòng:

- Dòng đầu chứa một số nguyên n là số đỉnh của đa giác (3 < n < 50).
- Mỗi dòng trong n dòng kế tiếp chứa hai số thực là hoành độ và tung độ của mỗi đỉnh của đa giác.

Kết quả: đưa ra file văn bản TAMPHAN.OUT, gồm dòng đầu chứa một số thực (có 4 chữ số thập phân) là tổng nhỏ nhất của độ dài của các đường chéo. Mỗi dòng trong n – 3 dòng tiếp theo chứa 2 số nguyên là chỉ số của hai đỉnh của mỗi đường chéo được chọn.

Ví dụ:

TAMPHAN.INP
6
2 1
2 4
6 6
10 6
10 3
7 0

TAMPHAN.OUT
17.4859
2 6
3 6
3 5

Hướng dẫn: Gọi Fx[i, j] là tổng độ dài ngắn nhất của các đường chéo khi tam phân đa giác có i đỉnh kể từ đỉnh thứ j.

4.11. Trạm bưu điện

Trên một con đường thẳng, dài, số nhà của những nhà dọc theo một bên đường là số đo độ dài tính từ đầu con đường (số nguyên). Người ta chọn ra k nhà làm trạm bưu điện.

Hãy xác định số nhà của k nhà đó sao cho các nhà còn lại cách một trạm bưu điện nào đó là gần nhất hay tổng khoảng cách của các nhà còn lại đến một trạm bưu điện gần nhất nào đó là nhỏ nhất.

Dữ liệu: vào từ file văn bản **BuuDien.inp** gồm 2 dòng:

- Dòng đầu: chứa hai số nguyên n và k (n < 300; k < 30), với n là tổng số nhà trên con đường đó, k là số trạm bưu điện.
- Dòng thứ hai là các số nhà theo thứ tự tăng dần.

Kết quả: đưa ra file văn bản BuuDien.out gồm 2 dòng:

- Dòng đầu: là k nhà dùng làm trạm bưu điện.
- Dòng thứ hai là tổng khoảng cách của các nhà còn lại đến một trạm bưu điện gần nhất nào đó.

Ví dụ:

BuuDien.inp	BuuDien.out
10 5 1 2 3 6 7 9 11 22 44 50	2 7 22 44 50 9

4.12. Xâu con chung dài nhất:

Cho hai xâu kí tự s1 và s2. Tìm xâu kí tự s có nhiều kí tự nhất, với s vừa là xâu con của xâu s1, vừa là xâu con của xâu s2. (Xâu con là xâu kí tự có được khi bỏ bớt một số kí tự trong xâu cha).

Dữ 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 s.

Ví dụ:

XauChung.inp	XauChung.out
luong the vinh bien hoa ngo quyen dong nai	9 ng en n a