Chuyên đề

PHƯƠNG PHÁP QUY HOẠCH ĐỘNG

I. GIỚI THIỆU

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;

Tính toán 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 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;
```

8. 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).

9. 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 ∈ [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

+