



CHƯƠNG V. Giải thuật qui hoạch động

NỘI DUNG:

- 5.1. Giới thiệu thuật toán
- 5.2. Bài toán chia số
- 5.3. Bài toán cái túi
- 5.4. Bài toán Lập lịch có trọng số
- 5.5. Một số bài toán khác
- 5.6. CASE STUDY

5.1.Thuật toán qui hoạch động(Dynamic Programming)

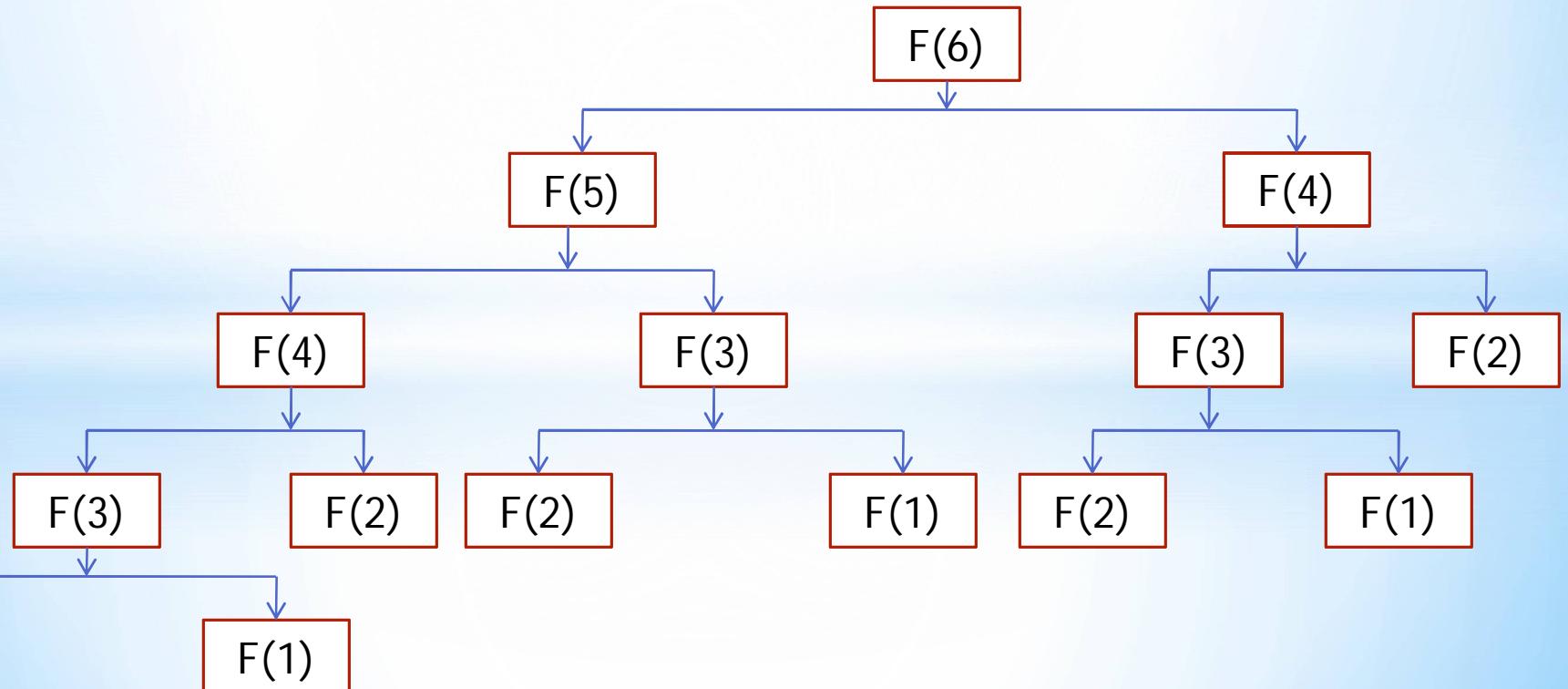
Phương pháp qui hoạch động dùng để giải lớp các bài toán thỏa mãn những điều kiện sau:

- Bài toán lớn cần giải có thể phân rã được thành nhiều bài toán con. Trong đó, sự phối hợp lời giải của các bài toán con cho ta lời giải của bài toán lớn. Bài toán con có lời giải đơn giản được gọi là cơ sở của qui hoạch động. Công thức phối hợp nghiêm của các bài toán con để có nghiêm của bài toán lớn được gọi là công thức truy hồi của qui hoạch động.
- Phải có đủ không gian vật lý lưu trữ lời giải các bài toán con (Bảng phương án của qui hoạch động). Vì qui hoạch động đi giải quyết tất cả các bài toán con, do vậy nếu ta không lưu trữ được lời giải các bài toán con thì không thể phối hợp được lời giải giữa các bài toán con.
- Quá trình giải quyết từ bài toán cơ sở (bài toán con) để tìm ra lời giải bài toán lớn phải được thực hiện sau hữu hạn bước dựa trên bảng phương án của qui hoạch động.

Ví dụ. Tìm số Fibonacci thứ n . Cho $F_1 = F_2 = 1$; $F_n = F_{n-1} + F_{n-2}$ với $n \geq 3$. Hãy tìm $F_6 = ?$.

```
#include <iosteam.h>
int F( int i ) {
    if ( i < 3 ) return (1);
    else return (F(i-1) + F(i-2));
}
void main(void) {
    cout<<"\n F[6] "<<F(6);
}
```

```
#include <iostream.h>
int main()
{
    int F[100]; F[1]=1; F[2] =1;
    for (int i=3; i<=6; i++)
        F[i] = F[i-1] + F[i-2];
    cout<<"\n F[6] = " <<F[6];
}
```



MỘT VÀI NHẬN XÉT

1. **Cách giải thứ nhất:** Sử dụng lời gọi đệ qui đến $F(6)$. Để tính được $F(6)$ cách giải thứ nhất cần phải tính 1 lần $F(5)$, 2 lần $F(4)$, 3 lần $F(3)$, 5 lần $F(2)$, 3 lần $F(1)$.
2. **Cách giải thứ hai (Qui hoạch động):** về bản chất cũng là đệ qui. Tuy nhiên phương pháp thực hiện theo các bước sau:
 - **Bước cơ sở** (*thường rất đơn giản*): Tính $F[1] = 1$, $F[2] = 1$.
 - **Công thức truy hồi** (*thường không đơn giản*): Lưu lời giải các bài toán con biết trước vào bảng phương án (*ở đây là mảng một chiều $F[100]$*). Sử dụng lời giải của bài toán con trước để tìm lời giải của bài toán con tiếp theo. Trong bài toán này là $F[i] = F[i-1] + F[i-2]$ với $n \geq 3$. Bằng cách lưu trữ vào bảng phương án, ta chỉ cần giải mỗi bài toán con một lần. Tuy vậy, cái giá phải trả là liệu ta có đủ không gian nhớ để lưu trữ lời giải các bài toán con hay không?
 - **Truy vết** (*thường đơn giản*): Đưa ra nghiệm của bài toán bằng việc truy lại dấu vết phối hợp lời giải các bài toán con để có được nghiệm của bài toán lớn. Trong bài toán này, vết của ta chính là $F[6]$ do vậy ta không cần phải làm gì thêm nữa.

Ví dụ. Tìm số các cách chia số tự nhiên n thành tổng các số tự nhiên nhỏ hơn n . Các cách chia là hoán vị của nhau chỉ được tính là một cách.

Lời giải. Gọi $F[m, v]$ là số cách phân tích số v thành tổng các số nguyên dương v nhỏ hơn hoặc bằng m . Bài toán đặt ra là tìm $F[n, n]$ là số các cách chia số n thành tổng các số nguyên dương nhỏ hơn n . Ta có:

- **Bước cơ sở:** $F[0,0] = 1$; $F[0,v] = 0$ với $v > 0$.
- **Công thức truy hồi:** Các cách chia số nguyên dương v thành tổng các số nhỏ hơn m được chia thành 2 loại. Loại 1 là số các cách phân tích không chứa số m . Khi đó, số các cách phân tích loại này là số các cách phân tích số v thành các số nhỏ hơn hoặc bằng $m-1$. Như vậy loại 1 chính là $F[m-1, v]$. Loại 2 là số các cách phân tích có ít nhất một số m . Khi đó nếu ta bỏ đi m thì số cách phân tích loại này là $F[m, v-m]$. Số Loại 1 chỉ xảy ra khi $m > v$, loại 2 chỉ xảy ra khi $m \geq v$. Từ đó ta có:
 - $F[m, v] = F[m-1, v]$ nếu $v > m$;
 - $F[m, v] = F[m-1, v] + F[m, v-m]$ nếu $m \geq v$.
- **Lưu ý.** Nhiệm vụ của bài toán là tìm $F[n, n]$.

Bảng phương án thực hiện theo phương pháp qui hoạch động được ghi nhận trong mảng hai chiều $F[m,v]$ như sau:

- Dòng 0 ghi lại $F[0, v]$. Trong đó, $F[0,0] = 1$; $F(0,v) = 0$, với $v > 1$.
- Các dòng tiếp theo được tính toán theo hệ thức truy hồi để tìm ra $F[n,n]$. Bảng phương án dưới đây ghi nhận với $n = 5$.
 - $F[m, v] = F[m-1, v]$ nếu $v > m$;
 - $F[m, v] = F[m-1, v] + F[m, v-m]$ nếu $m \geq v$.
 - Từ đó ta có $F[5, 5] = F[4,5] + F[5,0] = 7$ (cách).

| $F[m,v]$ | 0 | 1 | 2 | 3 | 4 | 5 | v |
|----------|---|---|---|---|---|---|-----|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 2 | 1 | 1 | 2 | 2 | 3 | 3 | |
| 3 | 1 | 1 | 2 | 3 | 4 | 5 | |
| 4 | 1 | 1 | 2 | 3 | 5 | 6 | |
| 5 | 1 | 1 | 2 | 3 | 5 | 7 | |

↓
m

Bài tập. Hãy cho biết kết quả thực hiện chương trình dưới đây?

```
#include      <iostream.h>
void main (void ) {
    int F[100][100], n, m, v;
    cout<<“Nhập n=”>>n; //Nhập n=5
    for ( int j=0; j <=n; j++) F[0][j] =0; //Thiết lập dòng 0 là 0.
    F[0][0] = 1; //Duy chỉ F[0][0] thiết lập là 1.
    for (m =1; m<=n; m++) {
        for (v= 0; v<=n; v++){
            if ( m > v ) F[m][v] = F[m-1][v];
            else F[m][v] = F[m-1][v] + F [m][v-m];
        }
    }
    cout<<“ Kết quả:”<<F[n][n]<<endl;
}
```

Bài tập 2. Hãy cho biết kết quả thực hiện chương trình dưới đây?

```
#include <iostream.h>
int F0[100], F[100], n, m, v, dem=0;
void Init(void){
    cout<<"Nhập n="; cin>>n;
    for(int i=0; i<=n; i++) F0[i]=0;
    F0[0]=1;
}
void Result(void) {
    cout<<"\n Kết quả bước "<<++dem<<":";
    for(int i=0; i<=n; i++)
        cout<<F0[i]<<" ";
}
void Replace(void ) {
    for(int i=0; i<=n; i++)
        F0[i]=F[i];
}
```

```
int main (void ) {
    Init();
    for (m=1; m<=n; m++) {
        for (v= 0; v<=n; v++){
            if ( v < m ) F[v] = F0[v];
            else F[v] = F0[v] + F[v-m];
        }
        Result();Replace();
    }
    Result();
    cout<<"\n Kết quả:"<<F[n]<<endl;
    system("PAUSE");return 0;
}
```

Bài tập 3. Hãy cho biết kết quả thực hiện chương trình dưới đây?

```
#include <iostream.h>
int F[100], n, m, v, dem=0;
void Init(void){
    cout<<"Nhập n="; cin>>n; //Nhập n=5
    for(int i=0; i<=n; i++) F[i]=0;//Thiết lập F0[i]=0; i=0, 1, 2, ..,n;
    F[0]=1;
}
void Result(void) {
    cout<<"\n Kết quả bước "<<++dem<<": ";
    for(int i=0; i<=n; i++)
        cout<<F[i]<<" ";
}
int main (void ) {  Init();
    for (m=1; m<=n; m++) { Result();
        for (v= m; v<=n; v++){
            F[v] = F[v] + F[v-m];
        }
    }
    Result();
    cout<<"\n Kết quả:"<<F[n]<<endl;
    system("PAUSE");return 0;
}
```

Bài tập 4. Hãy cho biết kết quả thực hiện chương trình dưới đây?

```
#include      <iostream.h>
int  n, dem=0;
int F(int m, int v){
    if (m==0) {
        if (v==0) return(1);
        else return(0);
    }
    else {
        if (m>v ) return (F(m-1, v));
        else return(F(m-1,v)+F(m,v-m));
    }
}
int main (void ) {
    cout<<"\n Nhập n=";cin>>n;
    cout<<"\n Kết quả:"<<F(n,n)<<endl;
    system("PAUSE");return 0;
}
```

CASE STUDY

- I. Sử dụng thuật toán sinh, thuật toán quay lui, thuật toán nhánh cận giải các bài toán dưới đây?
- II. Sử dụng tài liệu trên Internet, hãy cho biết những kết quả nghiên cứu tiếp theo của các bài toán trên và ứng dụng của nó?

Bài tập 1. *Bài toán cái túi.* Một nhà thám hiểm cần đem theo một cái túi trọng lượng không quá B . Có N đồ vật cần đem theo. Đồ vật thứ i có trọng lượng a_i , có giá trị sử dụng c_i ($i=1, 2, \dots, N$; $a_i, c_i \in \mathbb{Z}^+$). Hãy tìm cách đưa đồ vật vào túi cho nhà thám hiểm sao cho tổng giá trị sử dụng các đồ vật trong túi là lớn nhất

- **Tập phương án của bài toán:** Mỗi phương án của bài toán là một xâu nhị phân có độ dài N . Trong đó, $x_i = 1$ tương ứng với đồ vật i được đưa vào túi, $x_i = 0$ tương ứng với đồ vật i không được đưa vào túi. Tập các xâu nhị phân $X = (x_1, \dots, x_N)$ còn phải thỏa mãn điều kiện tổng trọng lượng không vượt quá B . Nói cách khác, tập phương án D của bài toán được xác định như công thức dưới đây.

$$D = \left\{ X = (x_1, x_2, \dots, x_N) : g(X) = \sum_{i=1}^N a_i x_i \leq B; x_i = 0, 1 \right\}$$

- **Hàm mục tiêu của bài toán:** Ứng với mỗi phương án $X = (x_1, \dots, x_N) \in D$, ta cần tìm phương án $X^* = (x_1^*, \dots, x_N^*)$ sao cho tổng giá trị sử dụng các đồ vật trong túi là lớn nhất. Do vậy, hàm mục tiêu của bài toán được xác định như sau:

$$f(X) = \sum_{i=1}^N c_i x_i \rightarrow \max$$

Bài tập 2. Bài toán người du lịch. Một người du lịch muốn đi tham quan N thành phố T_1, T_2, \dots, T_N . Xuất phát tại một thành phố nào đó, người du lịch muốn qua tất cả các thành phố còn lại mỗi thành phố đúng một lần rồi trở lại thành phố ban đầu. Biết c_{ij} là chi phí đi lại từ thành phố T_i đến thành phố T_j . Hãy tìm một hành trình cho người đi du lịch có tổng chi phí là nhỏ nhất.

- **Tập phương án của bài toán:** Không hạn chế tính tổng quát của bài toán, ta cố định xuất phát là thành phố $T_1 = 1$. Khi đó, mỗi hành trình của người du lịch $T_1 \rightarrow T_2 \rightarrow \dots \rightarrow T_N \rightarrow T_1$ được xem như một hoán vị của $2, \dots, N$ là $X = (x_1, x_2, \dots, x_N)$, trong đó $x_1 = 1$. Như vậy, tập phương án D của bài toán là tập các hoán vị $X = (x_1, x_2, \dots, x_N)$ với $x_1 = 1$.

$$D = \left\{ X = (x_1, x_2, \dots, x_N) : x_1 = 1 \wedge (\forall i \neq j) : x_i \neq x_j; i, j = 1, 2, \dots, N \right\}$$

- **Hàm mục tiêu của bài toán:** Ưng với mỗi phương án $X = (x_1, \dots, x_N) \in D$, chi phí đi lại từ thành phố thứ i đến thành phố j là $C[X[i]][X[i+1]]$ ($i=1, 2, \dots, N-1$). Sau đó ta quay lại thành phố ban đầu với chi phí là $C[X[N]][X[1]]$. Như vậy, hàm mục tiêu của bài toán được xác định như sau:

$$f(X) = \sum_{i=1}^{N-1} c_{x_i x_{i+1}} + c_{x_N x_1} \rightarrow \min$$

Bài tập 3. *Bài toán cho thuê máy.* Một ông cũ có một chiếc máy cho thuê. Đầu tháng ông nhận được yêu cầu của M khách hàng thuê máy cho N ngày kế tiếp. Mỗi khách hàng i cho biết tập N_i ngày họ cần thuê máy. Ông chỉ có quyền hoặc từ chối yêu cầu của khách hàng, hoặc nếu chấp nhận yêu cầu của khách ông phải bố trí máy theo đúng những ngày mà khách yêu cầu. Hãy tìm phương án thuê máy giúp ông chủ sao cho tổng số ngày thuê máy là nhiều nhất.

- **Tập phương án của bài toán:** Gọi $I = \{1, 2, \dots, M\}$ là tập chỉ số khách hàng, S là tập của tất cả các tập con của I . Khi đó, tập các phương án cho thuê máy là:

$$D = \left\{ J \subset S : N_k \cap N_p = \emptyset, \forall k, p \in J \right\}$$

- **Hàm mục tiêu:** Ứng với mỗi phương án $J \in D$, tổng số ngày cho thuê máy là:

$$f(J) = \sum_{j \in J} |N_j| \rightarrow \max$$

Bài tập 4. Bài toán phân công công việc. Một hệ gồm có N quá trình thực hiện N việc song hành. Biết mỗi quá trình đều có thể thực hiện được N việc kể trên nhưng với chi phí thời gian khác nhau. Biết c_{ij} là thời gian quá trình i thực hiện việc j. Hãy tìm phương án giao việc cho mỗi quá trình sao cho tổng thời gian thực hiện N việc kể trên là ít nhất.

- **Tập phương án của bài toán:** Gọi $X = (x_1, x_2, \dots, x_N)$ là một hoán vị của 1, 2, ..., N. Nếu $x_i = j$ thì ta xem quá trình thứ i được thực hiện việc j . Như vậy, tập phương án của bài toán chính là tập các hoán vị của 1, 2, ..., N.

$$D = \left\{ X = (x_1, x_2, \dots, x_N) : (\forall i \neq j) | x_i \neq x_j, i, j = 1, 2, \dots, N \right\}$$

- **Hàm mục tiêu của bài toán:** Ứng với mỗi phương án $X \in D$, thời gian thực hiện của mỗi phương án là:

$$f(X) = \sum_{i=1}^N c[i, x[i]] \rightarrow \min$$