QUY HOẠCH ĐỘNG

1. Giới thiệu

Quy hoạch động là phương pháp giải các bài toán đếm hoặc bài toán tối ưu có bản chất đệ quy dựa trên nguyên lý tối ưu do Richard Bellman (Mỹ) đề xuất.

Tư tưởng cơ bản của quy hoạch động là tránh tính toán lại mọi thứ.

2. Bài toán quy hoạch động

Quy hoạch động là một phương pháp hiệu quả để giải các bài toán tối ưu có 3 tính chất sau:

- Bài toán lớn có thể phân rã thành các bài toán con đồng dạng.
- Lời giải tối ưu của bài toán con có thể giúp tìm ra lời giải tối ưu của bài toán lớn (nguyên lý tối ưu Bellman).
- Các bài toán con trong quá trình phân rã có thể có chung một số bài toán con khác.

Hai tính chất đầu tiên là điều kiện cần của một bài toán quy hoạch động. Tính chất cuối cùng cho thấy phương pháp quy hoạch động hiệu quả hơn phương pháp đệ quy thông thường.

3. Phương pháp quy hoạch động

Quy hoạch động giải bài toán theo kỹ thuật bottom-up:

- Chia nhỏ bài toán ban đầu thành các bài toán con, tìm công thức truy hồi mô tả liên hệ giữa nghiệm của bài toán lớn với nghiệm của các bài toán con.
- Tìm nghiệm của bài toán nhỏ nhất, từ đó dần tìm nghiệm của bài toán lớn hơn theo công thức truy hồi.
- Tổ chức lưu trữ nghiệm của các bài toán con đủ để phục vụ tra cứu: bảng phương án.

Các bước tiếp cận phương pháp quy hoạch động:

- Xác định bài toán: xác định các tham số.
- Định nghĩa một hàm số tìm đại lượng tương ứng nghiệm.
- Xác định công thức truy hồi để tìm nghiệm của bài toán ban đầu từ nghiệm các bài toán con.

- Xác định nghiệm của bài toán con nhỏ nhất.
- Tìm cấu trúc dữ liệu thích hợp để tổ chức bảng phương án lưu trữ lại nghiệm của các bài toán con.

Ví dụ: Tìm phần tử thứ n của dãy Fibonacci

- Tham số bài toán: n
- Gọi $f(n)$ là nghiệm của bài toán, nghĩa là $f(n)$ là giá trị phần tử thứ n của dãy
- Công thức truy hồi $f(n) = f(n-1) + f(n-2)$
- Nghiệm của bài toán con nhỏ nhất: $f(1) = f(2) = 1$
- Tổ chức mảng 1 chiều để lưu trữ nghiệm của tất cả bài toán con

Điều kiện áp dụng phương pháp quy hoạch động

- Xây dựng được công thức truy hồi: nghiệm của bài toán có thể tổng hợp được từ nghiệm các bài toán con.
- Có thể có nhiều công thức truy hồi và hiệu quả của thuật toán phụ thuộc vào việc lựa chọn công thức truy hồi.
- Tổ chức được CTDL lưu trữ: nghiệm của tất cả bài toán con có thể lưu trữ theo một hình thức chấp nhận được

Một số bài toán có thể giải quyết bằng phương pháp quy hoạch động

- Bài toán đếm: đếm số cấu hình, đếm thứ tự từ điển, …
- Bài toán tối ưu.
- Bài toán lập bảng phương án.

Công đoạn cuối cùng của quy hoạch động là chỉ ra nghiệm của bài toán bằng cách lần ngược con đường dẫn đến kết quả để tìm nghiệm gọi là truy vết. Nếu bảng phương án không chứa đầy đủ thông tin phục vụ cho việc lần ngược để tìm nghiệm thì cần tổ chức các CTDL bổ sung để lưu lại vết trong quá trình tính công thức truy hồi.

4. Một số bài toán áp dụng

Bài toán 1: Đếm số dãy nhị phân độ dài n trong đó không tồn tại 2 bit 1 nằm liên tiếp nhau

Phân tích bài toán:

- Tham số bài toán: n

- Gọi $f(n)$ là nghiệm của bài toán, nghĩa là $f(n)$ là số dãy nhị phân độ dài n trong đó không tồn tại 2 bit 1 nằm liên tiếp nhau
- Mô hình hóa dãy nhị phân độ dài n thành dãy $\overline{a_1 a_2 \dots a_n}$
- Bản chất của bài toán: chia tập các dãy nhị phân độ dài n thành 2 tập con không giao nhau
  + Tập con gồm các dãy nhị phân có bit cuối là 0
  + Tập con gồm các dãy nhị phân có bit cuối là 1
- Xét bit cuối của dãy bằng 0, nghĩa là $\overline{a_1 a_2 \dots a_n} = \overline{a_1 a_2 \dots a_{n-1} 0}$, ta có
  + Số dãy $\overline{a_1 a_2 \dots a_n}$ bằng với số dãy $\overline{a_1 a_2 \dots a_{n-1}}$
  + Suy ra $f(n) = f(n-1)$
- Xét bit cuối của dãy bằng 1, nghĩa là $\overline{a_1 a_2 \dots a_n} = \overline{a_1 a_2 \dots a_{n-1} 1}$, ta có
  + Vì không tồn tại 2 bit 1 liên tiếp nhau nên $\overline{a_1 a_2 \dots a_{n-1} 1} = \overline{a_1 a_2 \dots a_{n-2} 01}$
  + Suy ra số dãy $\overline{a_1 a_2 \dots a_n}$ bằng với số dãy $\overline{a_1 a_2 \dots a_{n-2}}$
  + Suy ra $f(n) = f(n-2)$
- Theo nguyên lý cộng ta có công thức truy hồi $f(n) = f(n-1) + f(n-2)$
- Nghiệm của bài toán con nhỏ nhất $f(1) = 2, f(2) = 3$
- Độ phức tạp của thuật toán $O(n)$.

Bài toán 2: Cho dãy gồm n phần tử $a_1 a_2 \dots a_n$. Hãy tìm một dãy con (gồm các phần tử không nhất thiết liên tiếp) tăng dài nhất.

Phân tích bài toán:

- Tham số bài toán n
- Gọi $L(n)$ là độ dài của dãy con tăng dài nhất nhận a_n là phần tử cuối
- Công thức truy hồi $L(n) = \max\{L(i), \forall i < n : a_i < a_n\} + 1$
- Nghiệm của bài toán con nhỏ nhất $L(1) = 1$
- Độ phức tạp $O(n^2)$

Ví dụ:

A	8	4	10	15	5	6	7	3
L	1	1	2	3	2	3	4	1
i	1	2	3	4	5	6	7	8

Cài đặt:

```
void DP()
{
    L[1] = 1;
    for (int i = 2; i <= n; ++i)
    {
        max = 0;
        for (int j = 1; j <= i-1; ++j)
            if (a[j] < a[i] && L[j] > max)
                max = L[j];

        L[i] = max + 1;
    }
}
```

Truy vết:

- Để tìm các phần tử của dãy con tăng ta cần tổ chức thêm mảng `Prev[i]` cho biết chỉ số của phần tử nằm trước `a[i]` trong dãy con tăng dài nhất

Cài đặt hoàn chỉnh:

```
void DP()
{
    L[0] = 0;
    for (int i = 1; i <= n; ++i)
    {
        maxID = 0;
        for (int j = 1; j <= i-1; ++j)
            if (a[j] < a[i] && L[j] > L[maxID])
                maxID = j;

        L[i] = L[maxID] + 1;
        Prev[i] = maxID;
    }
}
```

Cài đặt truy vết tìm các phần tử thuộc dãy con tăng dài nhất

```
void Trace()
{
    maxID = 1;
    for (int i = 2; i <= n; ++i)
```

```
            if (L[i] > L[maxID]) maxID = i;
        while (maxID > 0)
        {
            ans.push_back(maxID);
            maxID = Prev[max];
        }
    }
```

Cải tiến:

- Gọi $id(len)$ là chỉ số của phần tử cuối nhỏ nhất trong dãy con tăng có đội dài len, nghĩa là $a[id(len)]$ là phần tử cuối có giá trị nhỏ nhất trong số các dãy con tăng có chiều dài len.

Ví dụ:

A	8	4	10	15	5	6	7	3
i	1	2	3	4	5	6	7	8

id	8	5	6	7
len	1	2	3	4

- Với mỗi phần tử `a[i]`, ta tìm trong dãy con tăng hiện tại (có độ dài `len`) phần tử nhỏ nhất lớn hơn `a[i]` và thay phần tử này bằng `a[i]`. Vì các phần tử trong mảng `id` có thứ tự tăng nên chặt nhị phân để tìm m sao cho `a[id[m]]` nhỏ nhất và `a[id[m]] >= a[i]`.
- Độ phức tạp của thuật toán $O(nlogn)$.

Cài đặt:

```
int BinSearch(int x, int l, int r)
{
    while (l < r)
    {
        int m = (l + r) / 2;
        if (a[id[m]] < x) l = m+1;
        else r = m;
    }
    return ((l+r) / 2);
}
```

```
void DP()
{
    id[1] = 1; len = 1;
    for (int i = 2; i <= n; ++i)
    {
        if (a[i] < a[id[1]]) id[1] = i;
```

```
        else if (a[i] > a[id[len]])
        {
            ++len;
            id[len] = i;
        }
        else //if (a[i] <= a[id[len]])
        {
            m = BinSearch(a[i], 1, len);
            id[m] = i;
        }
}
```

Bài toán 3: cho 2 dãy $a_1, a_2, \dots, a_m$ và $b_1, b_2, \dots, b_n$. Hãy tìm dãy con chung dài nhất của 2 dãy

Phân tích bài toán:

- Tham số bài toán: m, n
- Gọi $L[m][n]$ là độ dài của dãy con chung dài nhất của 2 dãy $a_1, a_2, \dots, a_m$ và $b_1, b_2, \dots, b_n$
- Xét 2 dãy $a_1, a_2, \dots, a_i (i \le m)$ và $b_1, b_2, \dots, b_j (j \le n)$, ta cần tính $L[i][j]$
- Xét trường hợp $a_i = b_j$ ta có $L[i][j] = L[i-1][j-1] + 1$
- Xét trường hợp $a_i \neq b_j$
  + Nếu $a_i \notin \{b_1, b_2, \dots, b_j\}$: $L[i][j] = L[i-1][j]$
  + Nếu $b_j \notin \{a_1, a_2, \dots, a_i\}$: $L[i][j] = L[i][j-1]$
  + Để tìm dãy con chung dài nhất ta cần tìm giá trị lớn nhất trong 2 giá trị trên
- Công thức truy hồi

$$L[i][j] = \begin{cases} L[i-1][j-1] + 1, & a_i = b_j \\ \max\{L[i-1][j], L[i][j-1]\}, & a_i \neq b_j \end{cases}$$

- Nghiệm bài toán con nhỏ nhất $L[0][j] = L[i][0] = 0$
- Độ phức tạp $O(m \times n)$

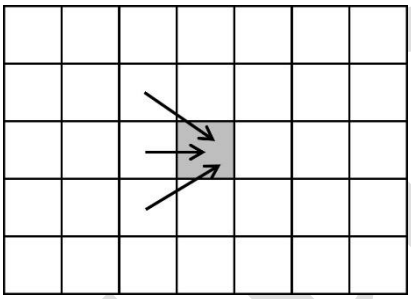
```
void Init()
{
    for (int j = 0; j <= n; ++j) L[0][j] = 0;
    for (int i = 0; i <= m; ++i) L[i][0] = 0;
}
void DP()
{
    for (int i = 1; i <= m; ++i)
```

```
        for (int j = 1; j <= n; ++j)
            if (a[i] == b[j])
                L[i][j] = L[i-1][j-1] + 1;
            else L[i][j] = max(L[i-1][j], L[i][j-1]);
 }
```

Bài toán 4: Cho bảng 2 chiều kích thước $m \times n (1 \leq m, n \leq 10^3)$. Phần tử ở dòng i, cột j có giá trị $a[i][j] (1 \leq a[i][j] \leq 10^5)$. Hãy tìm một đường đi xuất phát từ ô bất kỳ ở cột 1 và kết thúc tại ô bất kỳ ở cột n sao cho tổng giá trị các phần tử trên đường đi là lớn nhất.

Phân tích bài toán:

- Tham số bài toán: m, n
- Gọi $f[i][j]$ là tổng giá trị lớn nhất trên đường đi đến ô (i, j).
  + Nhận thấy ô (i, j) có thể đến được từ 1 trong 3 ô sau: $(i-1, j-1)$, $(i, j-1)$, $(i+1, j-1)$.



  + Công thức truy hồi: $f[i][j] = \max\{f[i-1][j-1], f[i][j-1], f[i+1][j-1]\} + a[i][j]$
- Nghiệm bài toán con nhỏ nhất: $f[i][1] = a[i][1] \ \forall i$.
- Nghiệm của bài toán: $\max\{f[i][n]\} \ \forall i$
- Độ phức tạp của bài toán: $O(m \times n)$