BÀI GIẢNG VỀ KỸ THUẬT LẬP TRÌNH VỚI C++

05 Phân tích và thiết kế thuật toán bằng phương pháp quy hoạch động (T1)

1. Chiến lược và nguyên lý tối ưu

Quy hoạch động (Dynamic Programming) được nhà toán học người Mỹ Richard Bellman đưa ra năm 1953. Đây là phương pháp hiệu quả để phân tích và thiết kế thuật toán. Phương pháp có nguyên lý và chiến lược như sau:

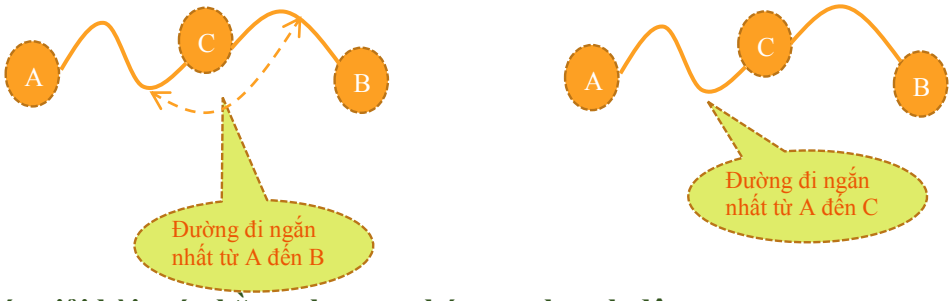
a. Chiến lược

Để giải một bài toán P, ta tiến hành phân chia bài toán thành nhiều bài toán con giao nhau. Tiến hành giải các bài toán con này và lưu vào một bảng. Từ các lời giải này, ta sẽ tìm ra lời giải cho bài toán P.

b. Nguyên lý tối ưu

Để có thể giải bài toán P bằng phương pháp quy hoạch động khi và chỉ khi P có ***cấu trúc con tối ưu***. Nghĩa là, với một lời giải tối ưu cho bài toán P, thì khi xét lời giải đó trên bài toán con P' ta cũng được một lời giải tối ưu.

Ví dụ: Xét đường đi ngắn nhất từ A đến B. Nếu có thể chia con đường từ A đến B thành hai đoạn, A đến C và từ C đến B. Thì cũng con đường đó, nếu đi từ A đến C thì cũng là đường đi ngắn nhất.



2. Các bước giải bài toán bằng phương pháp quy hoạch động

- **Phân chia và Khởi tạo**

  Tiến hành phân chia bài toán ban đầu thành các bài toán con:

  - Nhỏ hơn;
  - Dễ giải hơn;

  Trong lớp các bài toán con nhỏ được chia ra, có bài toán con nhỏ nhất mà lời giải của nó có thể tính trực tiếp được.

  Ở bước này, ta tính toán lời giải cho các bài toán con nhỏ nhất đó và ta còn gọi là **bước khởi tạo**.

- **Tìm Công thức quy hoạch động**

  Sau bước phân chia, sẽ là tiến hành giải các bài toán con dựa vào các lời giải của các bài toán con nhỏ hơn đã được giải. Công thức liên hệ giữa các bài toán con đó được gọi là **công thức quy hoạch động**.

  Lời giải của các bài toán con cần được lưu lại để sử dụng cho việc giải bài toán con khác.

- **Truy vấn lời giải của bài toán ban đầu**

Sau khi đã giải các bài toán con cần thiết, ta sẽ sử dụng các lời giải này để tìm ra lời giải bài toán ban đầu.

3. Một số bài toán quy hoạch động

Bài toán 1. Chọn các số hạng

Cho dãy các số nguyên $a_1, a_2, \ldots, a_n$. Hãy chọn các số hạng của dãy sao cho:

- Không được chọn hai số hạng kề nhau.
- Tổng các số hạng được chọn là lớn nhất.

Dữ liệu cho trong file SELSEQ.INP như sau:

- Dòng đầu ghi số nguyên dương n ($n \le 10^6$) là số số hạng của dãy.
- Dòng tiếp theo ghi n số nguyên $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10^6$).

Kết quả ghi ra file SELSEQ.OUT gồm một số nguyên s là tổng của các số hạng được chọn.

Ví dụ:

SELSEQ. INP	SELSEQ.OUT
6 1 -1 3 3 -10 9	13

Giải thích: Chọn các số hạng: 1, 3, 9; tổng bằng 13.

Bước 1. Phân chia và khởi tạo

Ta phân chia thành các bài toán con như sau:

Bài toán con	Dãy số nguyên	Kích thước bài toán
1	a_1	1 số hạng
2	a_1, a_2	2 số hạng
3	a_1, a_2, a_3	3 số hạng
.....	.....	.....
i	$a_1, a_2, .., a_i$	i số hạng
.....	.....	.....
n	$a_1, a_2, .., a_n$	n số hạng

Ta nhận thấy, bài toán con nhỏ nhất chỉ gồm một số hạng a_1 và các bài toán được chia có giao nhau.

Lời giải cho bài toán con nhỏ nhất là: Chọn số hạng a_1.

Lưu lời giải

Để lưu lời giải, ta sử dụng mảng $F[i]$ là nghiệm của bài toán con thứ i. Tức *F[i]* là tổng các số hạng đạt giá trị lớn nhất khi chọn các số hạng thuộc $a_1, a_2, .., a_i$ mà không chọn hai số hạng ở vị trí liên tiếp.

Khởi tạo: $F[1] = a[1]$;

Bước 2. Tìm công thức quy hoạch động

Tính $F[i]$, $(i = 2, 3, .., n)$ tức là ta đang xét bài toán $a_1, a_2, .., a_i$.

Ta có hai phương án trong cách chọn.

+ Không chọn a_i, nghĩa là các số hạng được chọn thỏa mãn yêu cầu thuộc dãy $a_1, a_2, .., a_{i-1}$. Đây chính là bài toán $F[i-1]$

+ Có chọn a_i, suy ra sẽ không chọn a_{i-1}. Như vậy sẽ có hai khả năng có thể xảy ra:

- Không chọn số hạng nào khác ngoài a_i → lời giải sẽ là a_i
- Có chọn các số hạng khác → Các số hạng khác này sẽ thuộc dãy $a_1, a_2, .., a_{i-2}$. Để tổng lớn nhất thì các số hạng được chọn trong dãy $a_1, a_2, .., a_{i-2}$ cũng phải lớn nhất. Do vậy lời giải cho ở đây là $F[i-2] + ai$.

Công thức quy hoạch động:

$$F[i] = max(F[i-1], a[i], F[i-2] + a[i]);$$

Bước 3. Truy vấn và tìm lời giải bài toán ban đầu

Ta nhận thấy, lời giải của bài toán ban đầu là $F[n]$.

```
#include<bits/stdc++.h>
using namespace std;
const int   N = 1000001;
long long   n;
long long   a[N];
long long   f[N];
int main(){
     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
     freopen("SELSEQ.INP","r",stdin);
     freopen("SELSEQ.OUT","w",stdout);
     cin>>n;
     for(int i = 1; i <= n; ++i)  cin>>a[i];
     //Khoi tao
     f[1] = a[1];                    //Chi co cach chon duy nhat la a[1]
     f[2] = max(a[1], a[2]);         //Chi co the chon mot so trong hai so
     //Quy hoach dong
     for(int i = 3; i <= n; ++i){
        int x = a[i];                // Chi chon a[i];
        int y = f[i-1];              // Khong chon a[i];
        int u = a[i] + f[i-2];       // Chon a[i] va cac so hang khac thuoc a[1], a[2], ..., a[i-2]
        f[i] = max(x, max(y, u));    // f[i] la max{x, y, u}
     }
     // Dua ket qua;
     cout<<f[n];
}
```

BÀI TẬP

1☼. Chọn các số hạng

Cho dãy các số nguyên $a_1, a_2, \ldots, a_n$. Hãy chọn các số hạng của dãy sao cho:

- Không được chọn hai số hạng kề nhau.
- Tổng các số hạng được chọn là lớn nhất.

Dữ liệu cho trong file SELSEQ.INP như sau:

- Dòng đầu ghi số nguyên dương n ($n \le 10^6$) là số số hạng của dãy.
- Dòng tiếp theo ghi n số nguyên $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10^6$).

Kết quả ghi ra file SELSEQ.OUT gồm một số nguyên s là tổng của các số hạng được chọn.

Ví dụ:

SELSEQ.INP	SELSEQ.OUT
6 1 -1 3 3 -10 9	13

Giải thích: Chọn các số hạng: 1, 3, 9; tổng bằng 13.

2☼. Tựa Fibonacci

Cho dãy số tựa Fibonacci $f_1, f_2, .., f_n, ..$ được định nghĩa:

$f_1 = f_2 = 1, f_i = f_{i-1} + 2f_{i-2}$ với $i = 3, 4, \ldots,$

Yêu cầu: Cho n, tính số dư khi chia f_n cho 2019.

Dữ liệu cho trong file LikeFib.Inp gồm một số nguyên dương n ($n \le 10^6$).

Kết quả ghi ra file LikeFib.Out là số dư khi chia f_n cho 2019.

Ví dụ:

LikeFib.Inp	LikeFib.Out
2	1

3☼. Tổng tiền tố

Cho dãy số nguyên $a_1, a_2, .., a_N$. Tổng các số hạng $S_i = a_1 + a_2 + .. + a_i$ được gọi là tổng tiền tố thứ i của dãy $a_1, a_2, .., a_N$.

Yêu cầu: Cho T chỉ số $i_1, i_2, .., i_T$, cần tính tổng tiền tố $S_{i_1}, S_{i_2}, .., S_{i_T}$.

Dữ liệu cho trong file SUMPREFIX.INP gồm:

- Dòng đầu ghi hai số nguyên dương N và T ($T \le N \le 10^5$).
- Dòng thứ 2 ghi N số nguyên $a_1, a_2, .., a_N$ ($|a_i| \le 10^6$).
- Dòng thứ 3 ghi T chỉ số $i_1, i_2, .., i_T$.

Kết quả ghi ra file SUMPREFIX.OUT gồm T dòng, dòng thứ k ghi tổng tiền tố S_{i_k}.

SUMPREFIX.INP	SUMPREFIX.OUT
5 2 1 2 5 1 3 1	8 1

4☼. Xếp hàng mua vé

Có n xếp hàng vào mua vé, được đánh số thứ tự từ 1 đến n. Mỗi người có thể mua 1 hoặc 2 vé hoặc không mua vé nào, thời gian mua một vé của người thứ i là t_i (nếu mua hai vé thì thời gian sẽ là $2t_i$). Do xếp thành hàng nên, người 1 sẽ được mua vé đầu tiên, tiếp theo là người thứ $2, 3, \ldots, n$. Người thứ i có thể nhờ người kề trước (tức là người thứ $i - 1$) mua vé thay cho mình, khi đó người thứ $i - 1$ không được nhờ người thứ $i - 2$ mua vé và sẽ mua hai vé.

Yêu cầu: Để mỗi người có một vé, thì tổng thời gian mua vé ít nhất là bao nhiêu?

Dữ liệu cho trong file Ticket.Inp như sau:

- Dòng đầu ghi số nguyên dương n ($n \le 10^5$).
- Dòng tiếp theo ghi n số nguyên dương $t_1, t_2, \ldots, t_n$ tương ứng là thời gian mua một vé của n người ($t_i \le 10^3$).

Kết quả ghi ra file Ticket.Out là tổng thời gian mua vé ít nhất để mỗi người có một vé.

Ví dụ:

Ticket.Inp	Ticket.Out	Giải thích
4 1 2 3 4	8	Người 1 mua hai vé cho người 1 và người 2, mất thời gian là 2; người 3 mua hai vé cho người 3 và người 4, mất thời gian là 6. Tổng thời gian mất là 2+6 = 8

5☼. Smart Frog1

Có N tảng đá được đánh số thứ tự $1, 2, 3, .., N$ (từ trái sang phải). Độ cao của tảng đá thứ i là h_i. Có một chú ếch đang ở tảng đá 1 và muốn nhảy đến tảng đá N. Khi chú ếch đang ở hòn đá i thì có thể nhảy đến hòn đá j với $j = i + 1$ hoặc $j = i + 2$ và năng lượng để nhảy là $|h_i - h_j|$.

Yêu cầu: Tính xem để nhảy đến hòn đá N, chú ếch sử dụng ít nhất bao nhiêu năng lượng.

Dữ liệu cho trong file SmartFrog1.Inp gồm:

- Dòng đầu ghi số nguyên dương N ($N \le 10^5$).
- Dòng sau ghi N số nguyên dương $h_1, h_2, .. , h_N$ ($h_i \le 10^6$) là độ cao của N hòn đá.

Kết quả ghi ra file SmartFrog1.Out gồm:

- Dòng đầu là tổng năng lượng ít nhất để chú ếch có thể nhảy đến hòn đá N.
- Dòng thứ hai gồm N bít 0 hoặc 1, trong đó bít thứ i là bít 1 có nghĩa là ếch nhảy đến cột thứ i, nếu bít thứ i là bít 0 có nghĩa là ếch không nhảy đến cột thứ i.

Ví dụ:

SmartFrog1.Inp	SmartFrog1.Out	Giải thích
4 10 30 40 20	30 1101	Cột 1 → 2 → 4. Tổng năng lượng: |10 – 30| + |30 – 20| = 30.

6☼. Smart Frog2

Có N tảng đá được đánh số thứ tự $1, 2, 3, .., N$ (từ trái sang phải). Độ cao của tảng đá thứ i là h_i. Có một chú ếch đang ở tảng đá 1 và muốn nhảy đến tảng đá N. Khi chú ếch đang ở hòn đá i thì có thể nhảy đến hòn đá j với $j \in \{i+1, i+2, i+3, .., i+k\}$ và năng lượng để nhảy là $|h_i - h_j|$.

Yêu cầu: Tính xem để nhảy đến hòn đá N, chú ếch sử dụng ít nhất bao nhiêu năng lượng.

Dữ liệu cho trong file SmartFrog2.Inp gồm:

- Dòng đầu ghi hai số nguyên dương N và K ($N \leq 10^5$, $K \leq 100$).
- Dòng sau ghi N số nguyên dương $h_1, h_2, .. , h_N$ ($h_i \leq 10^6$) là độ cao của N hòn đá.

Kết quả ghi ra file SmartFrog2.Out gồm:

- Dòng đầu là tổng năng lượng ít nhất để chú ếch có thể nhảy đến hòn đá N.
- Dòng thứ hai gồm N bít 0 hoặc 1, trong đó bít thứ i là bít 1 có nghĩa là ếch nhảy đến cột thứ i, nếu bít thứ i là bít 0 có nghĩa là ếch không nhảy đến cột thứ i.

Ví dụ:

SmartFrog2.Inp	SmartFrog2.Out	Giải thích
5 3 10 30 40 50 20	30 11001	Cột 1 → 2 → 5. Tổng năng lượng: |10 – 30| + |30 – 20| = 30.

6☼. Dãy con tăng dài nhất

Cho dãy số nguyên A gồm n số hạng $a_1, a_2, \ldots, a_n$. Ta gọi một dãy con của dãy A là dãy được tạo thành khi xóa đi (hoặc không xóa) một số các số hạng của dãy A, các số hạng còn lại vẫn giữ nguyên vị trí. Như vậy một dãy con của dãy A sẽ có dạng: $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ trong đó $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. Số số hạng của dãy được gọi là độ dài của dãy.

Yêu cầu: Tìm dãy con tăng dài nhất của A, tức là tìm dãy con $a_{i1}, a_{i2}, ..., a_{ik}$ sao cho:

- $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.
- $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$
- k đạt giá trị lớn nhất.

Dữ liệu cho trong file DAYTANG.INP gồm:

- Dòng đầu ghi số nguyên dương n ($n \leq 2000$).
- Dòng sau ghi n số nguyên $a_1, a_2, \ldots, a_n$ ($|a_i| \leq 10^9$).

Kết quả ghi ra file DAYTANG.OUT gồm:

- Dòng đầu ghi số số hạng trong dãy con tăng dài nhất, tức là đưa ra k.
- Dòng thứ hai ghi các chỉ số của các số hạng trong dãy con tăng dài nhất, tức là đưa ra các chỉ số $i_1, i_2, \ldots, i_k$.

Ví dụ:

DAYTANG.INP	DAYTANG.OUT
6 1 2 1 6 7 1	4 1 2 4 5