BÀI GIẢNG VỀ KỸ THUẬT LẬP TRÌNH VỚI C++

07 Phân tích và thiết kế thuật toán bằng phương pháp quy hoạch động (T3)

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:

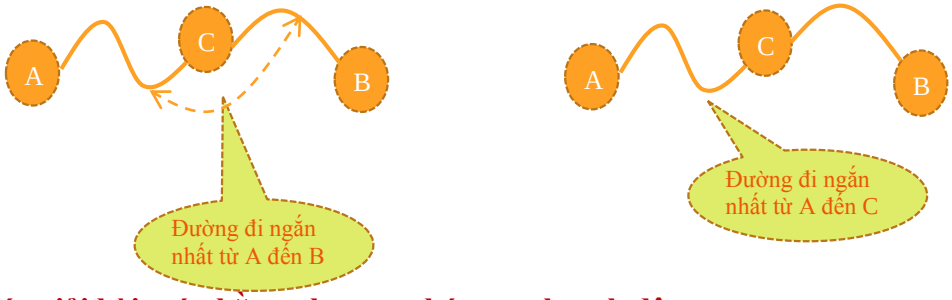
a. Chiến lược

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 $\boldsymbol{F[n]}$.

```
#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☼. Đường đi trên bảng số

Cho bảng số gồm m dòng và n cột, các dòng được đánh số từ 1 đến m, các cột được đánh số từ 1 đến n. Ô ở dòng i, cột j được gọi là ô (i, j). Trên mỗi ô (i, j) có ghi một số nguyên a_{ij}. Một người đặt một con robot tại ô $(1, 1)$ và tiến hành dịch chuyển con robot tới ô (m, n) với quy tắc dịch chuyển đó là: Nếu robot đang đứng tại ô (i, j) thì robot có thể dịch chuyển sang ô kề phải hoặc sang ô kề dưới, tức là sang ô $(i, j + 1)$ hoặc sang ô $(i + 1, j)$, tất nhiên là không dịch chuyển ra ngoài bảng.

Yêu cầu: Tìm cách dịch chuyển các con robot để tổng các số trên các ô mà robot đi qua đạt giá trị lớn nhất, kể cả hai ô $(1, 1)$ và (m, n).

Dữ liệu cho trong file MAXPATH.INP như sau:

- Dòng đầu ghi hai số nguyên dương m và n ($m, n \leq 1000$).
- m dòng tiếp theo, mỗi dòng ghi n số nguyên, các số nguyên thuộc [-10^6, 10^6].

Kết quả ghi ra file MAXPATH.OUT là tổng lớn nhất tìm được.

Ví dụ:

MAXPATH.INP	MAXPATH.OUT
4 5 **1 20 200 -200** -200 10 10 -1000 **100 100** 1 1 1 1 **1** 30 1 1 1 **1**	223

2☼. Đường đi trên lưới

Cho bảng số gồm m dòng và n cột. Các dòng được đánh chỉ số từ 1 đến m (từ trên xuống dưới), các cột được đánh chỉ số từ 1 đến n (từ trái qua phải). Ô ở dòng i, cột j được gọi là ô (i, j). Trên mỗi ô (i,j) có ghi số nguyên a_{ij}. Một con robot xuất phát từ ô (1,1) muốn đi tới ô (m, n). Robot đi theo quy tắc, nếu robot đang đứng ở ô (i, j) thì có thể đi sang ô $(i, j+1)$ hoặc $(i+1, j)$, tất nhiên là không đi ra khỏi bảng. Do phải nạp năng lượng nên robot phải đi qua ô (u, v).

Yêu cầu: Cho biết ô (u, v), tìm cho robot một đường đi từ (1,1) đến (m, n) sao cho tổng các ô trên đường đi của robot là lớn nhất.

Dữ liệu cho trong file **ROBOT.INP** gồm:

- Dòng đầu ghi bốn số nguyên dương m, n, u, v ($1 \leq u \leq m \leq 1000$, $1 \leq v \leq n \leq 1000$).
- m dòng sau, mỗi dòng ghi n số nguyên mô tả các số ghi trên các ô của dòng tương ứng.

Các số ghi trên bảng thuộc [-10^6, 10^6].

Kết quả ghi ra file **ROBOT.OUT** là tổng các số trên đường đi có tổng các ô đi qua đạt giá trị lớn nhất.

Ví dụ:

ROBOT.INP	ROBOT.OUT
4 4 2 2 1 1 1 1 2 2 1 1 1 5 1 1 1 2 2 1	15

3☼. Knapsack1

Có N đồ vật được đánh số thứ tự $1, 2, .., N$ và một cái túi có thể chứa tối đa (nhiều nhất) W đơn vị khối lượng. Vật thứ i có khối lượng là w_i và giá trị là c_i.

Yêu cầu: Hãy chọn các đồ vật để bỏ vào túi sao cho:

- Tổng khối lượng các vật được chọn không quá W.
- Tổng giá trị các vật được chọn đạt giá trị lớn nhất.

Dữ liệu cho trong file Knapsack1.Inp gồm:

- Dòng đầu ghi hai số nguyên dương N và W.
- Dòng thứ 2 ghi N số nguyên dương $w_1, w_2, .. , w_N$.
- Dòng thứ 3 ghi N số nguyên dương $c_1, c_2, .. , c_N$.

Kết quả ghi ra file Knapsack1.Out là tổng giá trị lớn nhất có thể đạt được của các vật được chọn.

Ví dụ:

Knapsack1.Inp	Knapsack1.Out	Giải thích
5 10 6 5 6 2 1 10 10 2 10 10	30	Chọn vật 2, 4, 5. Tổng khối lượng 5+2+1 = 8 Tổng giá trị 10+10+10 = 30

Giới hạn:

- $1 \le N \le 100$;
- $1 \le W \le 10^5$;
- $1 \le w_i \le W$;
- $1 \le c_i \le 10^9$;

4☼. Knapsack2

Có N đồ vật được đánh số thứ tự $1, 2, .., N$ và một cái túi có thể chứa tối đa (nhiều nhất) W đơn vị khối lượng. Vật thứ i có khối lượng là w_i và giá trị là c_i.

Yêu cầu: Hãy chọn các đồ vật để bỏ vào túi sao cho:

- Tổng khối lượng các vật được chọn không quá W.
- Tổng giá trị các vật được chọn đạt giá trị lớn nhất.
- Đưa ra số cách bỏ thỏa mãn.

Dữ liệu cho trong file Knapsack2.Inp gồm:

- Dòng đầu ghi hai số nguyên dương N và W.
- Dòng thứ 2 ghi N số nguyên dương $w_1, w_2, .. , w_N$.
- Dòng thứ 3 ghi N số nguyên dương $c_1, c_2, .. , c_N$.

Kết quả ghi ra file Knapsack2.Out gồm:

Dòng 1 là tổng giá trị lớn nhất có thể đạt được của các vật được chọn.

Dòng 2 là số cách chọn thỏa mãn.

Ví dụ:

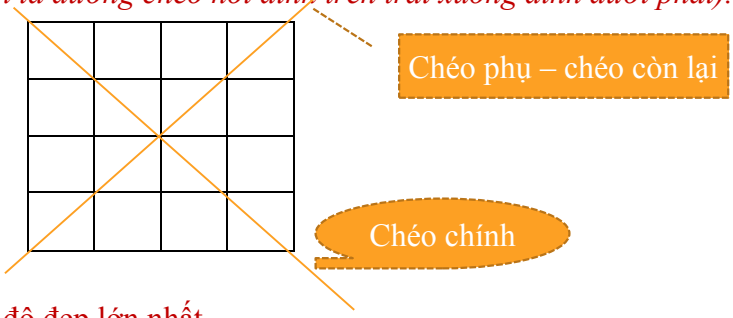
Knapsack2.Inp	Knapsack2.Out	Giải thích
5 10 6 5 6 2 1 10 10 2 10 10	30 2	Chọn vật: 2, 4, 5. Tổng khối lượng 5+2+1 = 8 Tổng giá trị 10+10+10 = 30 **Chọn vật**: 1, 4, 5.

Giới hạn:

- $1 \le N \le 100$;
- $1 \le W \le 10^5$;
- $1 \le w_i \le W$;
- $1 \le c_i \le 10^9$;

5 ☼. Hình vuông đẹp.

Cho một hình vuông được chia thành n dòng và n cột. Trên mỗi ô vuông nhỏ có ghi một số nguyên. Với một hình vuông con của hình vuông ban đầu, ta định nghĩa độ đẹp của hình vuông đó là hiệu của tổng các số trên các ô thuộc đường chéo chính trừ cho tổng các số trên các ô thuộc đường chéo còn lại *(đường chéo chính là đường chéo nối đỉnh trên trái xuống đỉnh dưới phải).*



Yêu cầu: Hãy tìm hình vuông con có độ đẹp lớn nhất.

Dữ liệu cho trong file NSQ.INP như sau:

- Dòng đầu tiên ghi số n ($2 \le n \le 400$).
- n dòng sau, mỗi dòng gồm n số nguyên có giá trị thuộc đoạn [-1000, 1000] là các số tương ứng ghi trên các ô vuông nhỏ.

Kết quả ghi ra file NSQ.OUT là độ đẹp của hình vuông tìm được.

Ví dụ:

NSQ.INP	NSQ.OUT
2 1 -2 4 5	4
3 -3 4 5 7 9 -2 1 0 -6	5

6 ☼. Đổi dấu

Cho dãy số nguyên $A_1, A_2, .. , A_N$. Phép đổi dấu trên dãy được thực hiện:

Chọn một chỉ số i ($1 \le i < N$), đổi dấu hai số hạng A_i và A_{i+1}, tức là $A_i = -A_i$, $A_{i+1} = -A_{i+1}$.

Yêu cầu: Cho phép thực hiện đổi dấu tùy ý lần (có thể không thực hiện lần nào). Từ dãy số nguyên $A_1, A_2, .. , A_N$ ban đầu, ta được dãy số nguyên $B_1, B_2, .. , B_N$ sao cho $S = B_1 + B_2 + .. + B_N$ đạt giá trị lớn nhất.

Dữ liệu cho trong file FlippSign.Inp gồm:

- Dòng đầu ghi số nguyên dương N ($N \le 5 \times 10^5$).
- Dòng thứ 2 ghi N số nguyên $A_1, A_2, .. , A_N$ ($|A_i| \le 10^9$).

Kết quả ghi ra file FlippSign.Out là tổng S đạt giá trị lớn nhất.

Ví dụ:

FlippSign.Inp	FlippSign.Out	Giải thích
3 1 -2 -3	6	Chọn chỉ số i = 2; đổi dấu -2 thành 2; -3 thành 3. Ta được dãy: 1, 2, 3, tổng bằng 6.
5 1 1 1 1 1	5	Không cần đổi dấu lần nào.

7☼ Dãy con không giảm dài nhất

Cho dãy số nguyên $a_1, a_2, ..., a_n$. Bằng cách đổi dấu không quá 3 số hạng, hãy tìm dãy con tăng dài nhất của dãy đã cho, tức là tìm dãy chỉ số $i_1, i_2, ..., i_k$ sao cho:

- $1 \le i_1 < i_2 < ... < i_k \le n$;
- Dãy con $a_{i1}, a_{i2}, ..., a_{ik}$ là dãy không giảm nếu đổi dấu không quá 3 số hạng của dãy này;
- k có giá trị lớn nhất có thể.

Dữ liệu cho trong file SEQD.INP như sau:

- Dòng thứ nhất ghi số nguyên dương n ($n \le 1000$);
- Dòng thứ hai ghi n số nguyên $a_1, a_2, ..., a_n$ ($|a_i| \le 10^9$).

Kết quả ghi ra file SEQD.OUT là độ dài của xâu con dài nhất tìm được.

Ví dụ:

SEQD.INP	SEQD.OUT	SEQD.INP	SEQD.OUT
5 9 1 2 3 4	5	6 9 9 1 0 -22 33	5