QUY HOẠCH ĐỘNG CƠ BẢN

Phan Văn Việt - Chuyên Vĩnh Phúc

Ngày 15 tháng 6 năm 2023

Quy hoạch động cơ bản và bài tập.

1 Bài toán ví dụ

1.1 Tóm tắt đề bài

IOI1994 - day 1 - Digital trialge

cho tam giác số gồm r ($2 \leq r \leq 100$) hàng. Các số trong tam giác là số nguyên nằm trong khoảng từ 0 đến 99. Hãy tìm một đường đi từ đỉnh tam giác xuống tới một vị trí bất kỳ ở đáy của tam giác sao cho tổng các số trên đường đi là lớn nhất có thể. Mỗi một bước chỉ có thể di chuyển xuống vị trí dưới hoặc dưới bên phải của vị trí hiện tại.

```
          7
        3   8
      8   1   0
    2   7   4   4
  4   5   2   6   5
```

Trong ví dụ trên, đường đi tối ưu là: $7 \to 3 \to 8 \to 7 \to 5$ có tổng là $7 + 3 + 8 + 7 + 5 = 30$.

```
          [7]
        [3]   8
      [8]   1   0
    2   [7]   4   4
  4   [5]   2   6   5
```

1.2 Giải pháp thô

- Thử mọi con đường có thể, so sánh để tìm ra đường đi có tổng lớn nhất;
- Độ phức tạp thuật toán: $\Theta(2^r)$. Quá giới hạn thời gian vì $r \leq 100$.

1.3 Giải pháp tốt hơn

- Nhận xét: dễ dàng nhận thấy rằng mỗi bước di chuyển trên đường đi tối ưu đều là phương án tối ưu nhất đến thời điểm đó.
- Tại mỗi vị trí, chỉ có hai lựa chọn di chuyển đến hàng tiếp theo là di chuyển đến số ngay dưới nó hoặc di chuyển đến vị trí ngay dưới bên phải nó. Do đó, chỉ cần ghi lại trọng số tối đa của vị trí hiện tại (đường đi có tổng lớn nhất, đi từ đỉnh tam giác tới vị trí hiện tại) để sử dụng nó cập nhật trọng số tối đa của bước tiếp theo.
- Với cách làm trên, ta đã cố gắng giảm quy mô bài toán bằng cách chia nó thành nhiều bài toán nhỏ hơn. Để có được phương án tối ưu trên hàng r, ta chỉ cần biết phương án tối ưu trên hàng $r - 1$.
- Lúc này vẫn còn một vấn đề cần giải quyết: sẽ có nhiều bài toán con xuất hiện lặp lại nhiều lần. Để giải quyết vấn đề này, ta cần lưu trữ lời giải của từng bài toán con để sử dụng lại khi cần thiết.

1.4 Giải pháp dùng mảng hai chiều

- Nhận thấy rằng việc thực hiện theo đúng tư tưởng của đề bài sẽ vô cùng rắc rối và nhiều khả năng vượt quá giới hạn thời gian. Vì vậy, ta sẽ thử đảo ngược giải pháp.
- Nhìn vào ví dụ, ta thấy, nếu bắt đầu từ giá trị 2 ở hàng thứ 2 từ dưới lên, ta chỉ có hai lựa chọn hoặc đi tới 4 ở dưới trái hoặc đi tới 5 ở dưới phải.
- Dựa trên nhận xét trên, ta gọi $f[i][j]$ ($0 \leq i \leq r, 1 \leq j \leq i, \forall i = 0 \to r - 1$) là tổng của đường đi tốt nhất từ hàng cuối tam giác đến vị trí thứ j, hàng thứ i của tam giác. Ta có tổng cần tìm chính là $f[0][0]$.

- Bảng các giá trị của f như sau:

			30					
		23			21			
	20			13			10	
7			12			10		10
4		5			2		6	5

Mã triển khai C++

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 101;
int n;
int a[1000][1000];

int main()
{
#define taskname "triangle"

    if (fopen (taskname".inp", "r"))
    {
        freopen (taskname".inp", "r", stdin);
        freopen (taskname".out", "w", stdout);
    }

    cin.tie (0)->sync_with_stdio (0);
    cin >> n;

    for (int i = 0; i < n; i++)
        for (int j = 0; j <= i; j++)
            cin >> a[i][j];

    for (int i = n - 2; i >= 0; i--)
    {
        for (int j = 0; j <= i; j++)
            a[i][j] += max (a[i + 1][j], a[i + 1][j + 1]);
    }

    cout << a[0][0] << endl;
}
```

1.5 Giải pháp dùng mảng một chiều

- Phương trình chuyển trạng thái: $a[j] = maxa[j], a[j+1] + p(a[j])$

Mã triển khai C++

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 101;
int n, a[N], i, j, ans, p;

int main()
{
#define taskname "triangle"

    if (fopen (taskname".inp", "r"))
    {
        freopen (taskname".inp", "r", stdin);
        freopen (taskname".out", "w", stdout);
    }
```

```
15
16      cin.tie (0)->sync_with_stdio (0);
17      cin >> n;
18
19      for (i = n; i; i--)
20          for (j = i; j <= n; j++)
21              cin >> p, a[j] = max (a[j], a[j + 1]) + p;
22
23      for (i = 1; i <= n; i++)
24          ans = max (ans, a[i]);
25
26      cout << ans << '\n';
27  }
```

2 Nguyên lý lập trình Quy hoạch động

Các bài toán giải quyết được bằng quy hoạch động cần thoả mãn 3 điều kiện:

- Đặc trưng hoá cấu trúc của lời giải tối ưu;
  - Hãy cẩn thận để đảm bảo sẽ kiểm tra tất cả các bài toán con được sử dụng trong giải pháp tối ưu.
    1. Việc đầu tiên để chứng minh một giải pháp tối ưu cho một vấn đề là đưa ra lựa chọn;
    2. Đối với một vấn đề nhất định, trong số các lựa chọn đầu tiên có thể của nó, hãy giả sử rằng bạn đã biết lựa chọn nào sẽ dẫn đến giải pháp tối ưu mà không cần quan tâm đến việc làm thế nào để có được lựa chọn này;
    3. Xác định những bài toán con có thể xuất hiện và xác định cách mô tả tốt nhất cho không gian bài toán con;
    4. Chứng minh rằng với tư cách là một phần của lời giải tối ưu cho bài toán ban đầu, lời giải mỗi bài toán con là lời giải tối ưu của chính nó. Phương pháp này là phương pháp phản chứng, giả sử nghiệm của một bài toán con nào đó không phải nghiệm tối ưu của chính nó thì tồn tại một nghiệm tối ưu khác để tạo ra nghiệm tối ưu cho bài toán ban đầu, do đó mâu thuẫn với giả thiết về lời giải tối ưu ban đầu.
  - Giữ cho không gian bài toán con càng đơn giản càng tốt và chỉ mở rộng khi thực sự cần thiết;
  - Cấu trúc con tối ưu khác nhau theo hai cách:
    1. Có bao nhiêu bài toán con tham gia vào lời giải tối ưu của bài toán ban đầu;
    2. Có bao nhiêu lựa chọn cần được xem xét trong việc xác định bài toán con nào sẽ được sử dụng cho lời giải tối ưu.
  - Mỗi đỉnh trong đồ thị bài toán con tương ứng với một bài toán con, và các lựa chọn được xem xét tương ứng với các cạnh liên kết với các đỉnh của bài toán con.
- Lời giải của bài toán con trước đó không bị thay đổi bởi các lựa chọn sau này;
- Lưu trữ lời giải các bài toán con để tránh việc giải lại một bài toán con nhiều lần. Từ đó giảm độ phức tạp thuật toán.

3 Ý tưởng cơ bản khi giải một bài toán bằng quy hoạch động

1. Chia bài toán ban đầu thành nhiều **giai đoạn**, mỗi giai đoạn tương ứng với một bài toán con và rút ra đặc điểm của bài toán con này (gọi là **trạng thái quy hoạch động**).
2. Tìm **quyết định** khả dĩ cho mỗi trạng thái (còn gọi là phương trình chuyển đổi trạng thái).

3. Giải quyết các vấn đề của từng giai đoạn theo thứ tự.

Nếu ta sử dụng ý tưởng của lý thuyết đồ thị để hiểu, ta xây dựng một đồ thị **DAG**, mỗi trạng thái ứng với một nút, quyết định chuyển trạng thái ứng với một cạnh nối giữa hai nút. Lúc này, vấn đề trở thành tìm đường đi dài nhất (hoặc ngắn nhất) trên đồ thị **DAG**.

3.1 Xâu con chung dài nhất [LCS]

Bài toán: Cho dãy A có độ dài n và dãy B có độ dài m $(n, m \leq 5000)$, tìm dãy dài nhất sao cho dãy này vừa là một dãy con của A vừa là một dãy con của B. Ví dụ: các dãy con chung của chuỗi 'abcde' và chuỗi 'acde' là 'a', 'c', 'd', 'e', 'ac', 'ad', 'ae', 'cd ', 'ce', 'de', 'ade', 'ace', 'cde', 'acde', độ dài của dãy con chung dài nhất là 4.

Gọi $f(i, j)$ là độ dài của dãy con chung dài nhất khi chỉ xét các phần tử i đầu tiên của A và các phần tử j đầu tiên của B, và tìm dãy con chung dài nhất tại thời điểm này là **câu hỏi phụ**. $f(i, j)$ là **trạng thái**, $f(n, m)$ là trạng thái cuối cùng cần đạt được, tức là câu trả lời của bài toán gốc.

Đối với mỗi $f(i, j)$, có ba quyết định: nếu $A_i = B_j$, có thể nối kí tự này với phần cuối của dãy con chung; hai quyết định còn lại là bỏ qua A_i hoặc bỏ qua B_j. Phương trình chuyển trạng thái như sau:

$$f(i,j) = \begin{cases} f(i-1, j-1) + 1 & \text{với } A_i = B_j \\ max(f(i-1, j), f(i, j-1)) & \text{với } A_i \neq B_j \end{cases}$$

Bạn có thể tham khảo `http://lcs-demo.sourceforge.net/` để hiểu rõ hơn về quy trình triển khai LCS. Độ phức tạp thời gian của giải pháp này là $\Theta(n \times m)$.

Ngoài ra, còn có một thuật toán $\Theta(\frac{n \times m}{w})$ cho câu hỏi này. Học sinh quan tâm có thể tự tìm hiểu về nó.

Mã triển khai:

```
int a[MAXN], b[MAXM], f[MAXN][MAXM];

int dp()
{
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            if (a[i] == b[j])
                f[i][j] = f[i - 1][j - 1] + 1;

            else
                f[i][j] = max (f[i - 1][j], f[i][j - 1]);

    return f[n][m];
}
```

3.2 Dãy con không giảm dài nhất

Bài toán: cho dãy A có độ dài n $(n \leq 5000)$, hãy tìm dãy con dài nhất của A, thoả mãn điều kiện phần tử tiếp theo của dãy con không nhỏ hơn phần tử đứng trước nó.

Giải pháp 1:

Gọi $f(i)$ là độ dài của dãy con không giảm dài nhất kết thúc tại A_i, thì độ dài dãy con dài nhất thoả mãn là $\max_{1 \leq i \leq n} f(i)$.

Trường hợp cơ sở: $f[1] = 1$. Khi tính toán $f(i)$, hãy cố gắng chọn A_i vào các chuỗi con không giảm dài nhất khác để cập nhật câu trả lời. Phương trình chuyển trạng thái như sau: $f(i) = \max_{1 \leq j < i, A_j \leq A_i}(f(j) + 1)$.

Dễ dàng thấy rằng độ phức tạp thời gian của thuật toán này là $\Theta(n^2)$.

Mã triển khai:

```
int a[MAXN], d[MAXN];

int dp()
{
    d[1] = 1;
    int ans = 1;

    for (int i = 2; i <= n; i++)
    {
        d[i] = 1;

        for (int j = 1; j < i; j++)
            if (a[j] <= a[i])
            {
                d[i] = max (d[i], d[j] + 1);
                ans = max (ans, d[i]);
            }
    }

    return ans;
}
```

Giải pháp 2:

Khi phạm vi của n được mở rộng đến $n \leq 10^5$, giải pháp 1 không đủ nhanh, dưới đây là giải pháp với độ phức tạp $\Theta(n \log n)$:

- Đầu tiên, xác định $a_1 \dots a_n$ là dãy ban đầu, d là dãy con không giảm hiện tại và len là độ dài của dãy con đó, sau đó d_{len} là dãy con không giảm của len Phần tử ở cuối dãy.
- Khởi tạo: $d_1 = a_1, len = 1$.
- Bây giờ chúng ta biết rằng độ dài của dãy con không giảm dài nhất là 1, sau đó chúng ta cho i lặp từ 2 đến n và lần lượt tìm độ dài của dãy con không giảm dài nhất trong số các phần tử i đầu tiên, ta cần duy trì mảng d và len. **Điều quan trọng là làm thế nào để duy trì chúng**.
- Xét phần tử a_i:
  1. Nếu a_i lớn hơn hoặc bằng d_{len}, thì chèn trực tiếp a_i vào cuối dãy d.
  2. Nếu a_i nhỏ hơn d_{len}, tìm phần tử **đầu tiên** lớn hơn phần tử đó, chèn phần tử đó và loại bỏ tất cả các phần tử sau phần tử đó.

Mã triển khai:

```
for (int i = 0; i < n; ++i) scanf ("%d", a + i);

memset (dp, 0x1f, sizeof dp) ;
mx = dp[0];

for (int i = 0; i < n; ++i)
{
    *std::upper_bound (dp, dp + n, a[i]) = a[i];
}

ans = 0;

while (dp[ans] != mx) ++ans;
```

4 Bài tập vận dụng

Dãy con tăng dài nhất [LIS]

Cho dãy gồm n số nguyên $a_1, a_2, \cdots, a_n$. Một dãy con của dãy đã cho là dãy thu được bằng cách xoá đi một số phần tử của dãy ban đầu mà vẫn giữ nguyên thứ tự.

Yêu cầu: Hãy tìm độ dài dãy con tăng (phần tử đứng sau luôn lớn hơn phần tử đứng trước) của dãy.

Dữ liệu

- Dòng 1: ghi số nguyên n ($1 \leq N \leq 2 \times 10^5$)
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10^9, \forall i = 1 \to n$).

Kết quả

- Ghi một số nguyên duy nhất là độ dài của dãy con tăng dài nhất tìm được.

Ví dụ

Input	**Output**
8 7 3 5 3 6 2 9 8	4

Giải thích ví dụ

- Một trong những con tăng dài nhất tìm được là $\{3, 5, 6, 9\}$

Xâu con chung dài nhất [LCS]

Cho hai xâu A và B. Xâu con của một xâu là xâu nhận được bằng cách xoá đi một số kí tự của xâu ban đầu mà vẫn giữ nguyên thứ tự.

Yêu cầu: Tìm độ dài xâu con chung dài nhất của hai xâu đã cho.

Dữ liệu

- Dòng 1: ghi xâu A gồm không quá 3000 kí tự chữ cái tiếng Anh in thường;
- Dòng 2: ghi xâu B gồm không quá 3000 kí tự chữ cái tiếng Anh in thường.

Kết quả

- Ghi một xâu duy nhất là độ dài xâu con chung dài nhất tìm được. Nếu có nhiều xâu con thoả mãn thì chỉ cần in ra một xâu trong đó.

Input	**Output**
axyb abyxb	axb

Đổ xúc xắc [DICE]

Hãy viết chương trình đếm xem có bao nhiêu cách để đạt được tổng số điểm là n bằng cách gieo xúc xắc một hoặc nhiều lần. Mỗi lần gieo xúc xắc cho ra một số trong đoạn từ 1 đến 6.

Dữ liệu

- Dòng 1: ghi số nguyên N ($1 \leq N \leq 10^6$)

Kết quả

- Ghi một số nguyên duy nhất là số cách đếm được. Vì số cách có thể rất lớn nên bạn hãy chia lấy dư cho $10^9 + 7$ trước khi in ra nhé.

Ví dụ

Input	**Output**
1	1
3	4

Giải thích ví dụ 2

- Có 4 cách là:
  1. $(1 + 1 + 1)$;
  2. $(1 + 2)$;
  3. $(2 + 1)$;
  4. (3).

Số đồng xu tối thiểu [MINCOIN]

Xét một hệ thống gồm n loại đồng tiền xu.Hãy tìm cách để đổi x đồng ra tiền xu sao cho số đồng xu là ít nhất có thể.

Ví dụ: với ba loại đồng xu có mệnh giá tương ứng là 1,5,7 và số tiền cần đổi là $x = 11$ thí đáp án là 3 đồng xu $5 + 5 + 1 = 11$.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, x $(1 \leq n \leq 100; 1 \leq x \leq 10^6)$ tương ứng với số loại đồng tiền xu và số tiền cần đổi;
- Dòng 2: ghi n số nguyên $c_1, c_2, ...c_n$ tương ứng với mệnh giá của các loại đồng xu.

Kết quả

- Một số nguyên duy nhất là số đồng tiền xu ít nhất có thể đổi. Nếu không có cách đổi ra xu, ghi -1.

Ví dụ

input	**output**
3 11 1 3 5	3

Giải thích ví dụ

- Chọn ba đồng xu $1 + 5 + 5 = 11$

Đổi xu 1 [COIN1]

Hãy xem xét một hệ thống tiền bao gồm n đồng xu. Mỗi đồng xu có giá trị là một số nguyên dương. Nhiệm vụ của bạn là tính số lượng các cách khác nhau mà bạn có thể tạo ra một khoản tiền là x cách sử dụng các đồng xu có sẵn. Ví dụ: nếu các đồng xu là $(2, 3, 5)$ và tổng mong muốn là 9 , có 8 cách:

- $2 + 2 + 5$
- $2 + 5 + 2$
- $5 + 2 + 2$
- $3 + 3 + 3$
- $2 + 2 + 2 + 3$
- $2 + 2 + 3 + 2$
- $2 + 3 + 2 + 2$
- $3 + 2 + 2 + 2$

Dữ liệu

- Dòng 1: ghi hai số nguyên n, x $(1 \leq n \leq 100, 1 \leq x \leq 10^6)$ - số lượng đồng xu và tổng số tiền mong muốn.
- Dòng 2: ghi n số nguyên phân biệt biệt $c_1, c_2, \cdots, c_n (1 \leq c_i \leq 10^6, \forall i = 1 \rightarrow n)$ - giá trị của mỗi đồng xu.

Kết quả

- Ghi một số nguyên duy nhất số lượng cách sau khi chia lấy dư cho $10^9 + 7$.

Ví dụ

input	**output**
3 9 2 3 5	8

Đổi xu 2 [COIN2]

Hãy xem xét một hệ thống tiền bao gồm n đồng xu. Mỗi đồng xu có giá trị là một số nguyên dương. Nhiệm vụ của bạn là đếm xem có bao nhiêu cách khác nhau để tạo ra một khoản tiền x bằng cách sử dụng các đồng xu có sẵn, các đồng xu được lấy theo thứ tự tăng dần.

Ví dụ: nếu các đồng xu là $\{2, 3, 5\}$ và tổng mong muốn là 9, có 3 cách:

1. $\{2 + 2 + 5\}$
2. $\{3 + 3 + 3\}$
3. $\{2 + 2 + 2 + 5\}$

Dữ liệu

- Dòng 1: ghi hai số nguyên n, x $(1 \leq N \leq 100, 1 \leq x \leq 10^6)$;
- Dòng 2: ghi n số nguyên $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10^6, \forall i = 1 \rightarrow n)$.

Kết quả

- Ghi một số nguyên duy nhất là số cách đếm được.

Ví dụ

Input	**Output**
3 9 2 3 5	3

Giảm số [SUBNUM]

Cho số nguyên n. Mỗi thao tác bạn có thể chọn một chữ số của n rồi giảm n đi một lượng bằng chữ số đó. Cần tối thiểu bao nhiêu thao tác để làm cho $n = 0$?

Dữ liệu

- Dòng 1: ghi số nguyên n ($1 \leq N \leq 10^6$);

Kết quả

- Ghi một số nguyên duy nhất là số thao tác tối thiểu để làm cho $n = 0$.

Ví dụ

Input	Output
27	5

Giải thích ví dụ

- $27 \to 20 \to 18 \to 10 \to 9 \to 0$.

Đường đi trên lưới [GRIDPATH]

Cho một bản đồ biểu diễn dưới dạng ma trận $n \times n$. Mỗi ô là một kí tự '.' biểu diễn vùng đất bằng có thể đi qua được hoặc kí tự '*' biểu diễn vùng đất có cạm bẫy không thể đi qua được. Xuất phát từ vị trí ô trên cùng bên trái, nỗi lần di chuyển, bạn chỉ có thể đi từ ô hiện tại tới ô cạnh bên phải hoặc tới ô cạnh ngay dưới. Hỏi có bao nhiêu cách để đi tới ô dưới cùng bên phải của bản đồ.

Dữ liệu

- Dòng 1: ghi số nguyên n ($1 \leq n \leq 1000$) là kích thước bản đồ;
- Dòng 2 đến dòng $n + 1$: mỗi dòng ghi một xâu gồm n kí tự '.' hoặc '*'.

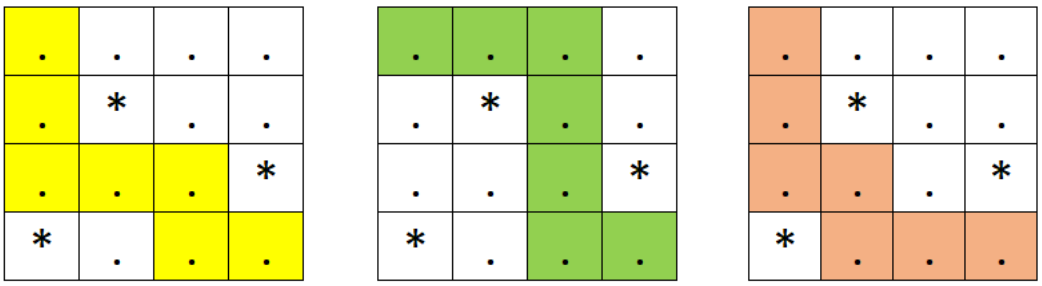
Kết quả

- Một số nguyên duy nhất là số cách để đi tới ô dưới cùng bên phải bản đồ. Vì số cách có thể rất lớn nên bạn chỉ cần in ra kết quả là số cách chia lấy dư cho $10^9 + 7$

Ví dụ

input	output
4 .... .*.. ...* *...	3

Giải thích ví dụ



Cái ba lô [KNAPSACK]

Có n món hàng, món hàng thứ i có trọng lượng là w_i và giá trị là c_i.

Tên trộm Dalton chỉ có một cái túi có thể chứa được tối đa là T. Hãy tính xem hắn có thể lấy vào chiếc túi các mặt hàng với tổng giá trị tối đa là bao nhiêu.

Dữ liệu

- Dòng 1: ghi hai số nguyên n, T ($1 \leq n \leq 100, 1 \leq T \leq 10^5$);
- Dòng 2: tiếp theo là n dòng, mỗi dòng ghi hai số nguyên w_i, c_i ($1 \leq w_i \leq T, 1 \leq c_i \leq 10^9$).

Kết quả

- Ghi một số nguyên duy nhất là tổng giá trị tối đa các mặt hàng Dalton có thể cho vào túi.

Input	**Output**
3 70 71 100 69 1 1 2	3

Giải thích ví dụ

- Dalton chỉ có thể lấy hai mặt hàng: thứ hai và thứ ba với tổng giá trị là $1 + 2 = 3$.