Nguyễn Xuân Huy

**Dynamic Programming Method**

**Các chuyên đề chọn lọc**

Hà Nội - 2023

# Phương pháp quy hoạch động

Các bài toán *quy hoạch động* chiếm một vị trí khá quan trọng trong tổ chức hoạt động và sản xuất. Chính vì lẽ đó mà trong các kì thi học sinh giỏi quốc gia và quốc tế chúng ta thường gặp loại toán này.

Thông thường những bạn nào dùng phương pháp quay lui, vét cạn thì chỉ có thể vét được các tập dữ liệu nhỏ, kích thước chừng vài chục byte. Nếu tìm được đúng hệ thức thể hiện bản chất quy hoạch động của bài toán và khéo tổ chức dữ liệu thì ta có thể xử lí được những tập dữ liệu khá lớn.

Có thể tóm lược nguyên lí quy hoạch động do Bellman phát biểu như sau:

|  |
| --- |
| **Quy hoạch động** |
| *Quy hoạch động là lớp các bài toán mà quyết định ở bước thứ i*  *phụ thuộc vào quyết định ở các bước đã xử lí trước hoặc sau bước đó.* |

Để giải các bài toán quy hoạch động, ta có thể theo sơ đồ đơn giản sau đây:

|  |
| --- |
| **Sơ đồ giải bài toán quy hoạch động** |
| 1. *Lập hệ thức:* Lập hệ thức biểu diễn tương quan quyết định của bước đang xử lí với các bước đã xử lí trước hoặc sau bước đó. Hệ thức này được gọi là *phương trình Bellman.* Khi đã có hệ thức tương quan, chúng ta đã có thể xây dựng ngay thuật giải, tuy nhiên các hệ thức này thường là các biểu thức *đệ quy*, do đó dễ gây ra hiện tượng tràn miền nhớ và tốn thời gian khi ta tổ chức chương trình trực tiếp bằng đệ quy. 2. *Tổ chức dữ liệu và chương trình:* Tổ chức dữ liệu tính toán dần theo từng bước. Nên tìm cách *khử đệ quy*. Muốn khử đệ quy ta cần lưu lại các giá trị đã tính tại mỗi bước. Trong các bài toán quy hoạch động thuộc chương trình phổ thông thường đòi hỏi một vài mảng một hoặc hai chiều đảm nhận việc lưu trữ này. 3. *Làm tốt:* Làm tốt thuật toán bằng cách thu gọn hệ thức quy hoạch động và giảm kích thước miền nhớ: thay mảng hai chiều bằng một vài mảng một chiều. |

### Chia thưởng

Cần chia hết m phần thưởng cho n học sinh sắp theo thứ tự từ giỏi trở xuống sao cho mỗi bạn không nhận ít phần thưởng hơn bạn xếp sau mình. Hãy tính số cách chia? 1 ≤ m, n ≤ 70.

Ví dụ, với số phần thưởng m = 7, và số học sinh n = 4 sẽ có 11 cách chia 7 phần thưởng cho 4 học sinh theo yêu cầu của đầu bài. Đó là:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phương án | Học sinh | | | |
| ① | ② | ③ | ④ |
| 1 | 7 | 0 | 0 | 0 |
| 2 | 6 | 1 | 0 | 0 |
| 3 | 5 | 2 | 0 | 0 |
| 4 | 5 | 1 | 1 | 0 |
| 5 | 4 | 3 | 0 | 0 |
| 6 | 4 | 2 | 1 | 0 |
| 7 | 4 | 1 | 1 | 1 |
| 8 | 3 | 3 | 1 | 0 |
| 9 | 3 | 2 | 2 | 0 |
| 10 | 3 | 2 | 1 | 1 |
| 11 | 2 | 2 | 2 | 1 |

Bài này còn có cách phát biểu khác như sau: *Có bao nhiêu cách viết số tự nhiên m thành tổng của n số hạng theo trật tự giảm dần: số hạng viết sau không lớn hơn số hạng viết trước nó.*

#### Thuật toán

#### Lập hệ thức quy hoạch động

Gọi Chia(*pt, hs*) là số cách chia *pt* phần thưởng cho *hs* học sinh, ta thấy:

* Nếu không có học sinh (*hs* = 0) thì không có cách chia nào (Chia = 0).
* Nếu không có phần thưởng (*pt* = 0) thì chỉ có một cách chia (Chia(0, *hs*) = 1: mỗi học sinh nhận 0 phần thưởng).
* Nếu số phần thưởng ít hơn số học sinh (*pt* < *hs*) thì trong mọi phương án chia, từ học sinh thứ *pt* + 1 trở đi sẽ không được nhận phần thưởng nào: Chia(*pt,hs*) = Chia(*pt, pt*) nếu *pt* < *hs*.
* Ta xét tất cả các phương án chia trong trường hợp *pt* ≥ *hs*. Ta tách các phương án chia thành *hai nhóm không giao nhau* dựa trên số phần thưởng mà học sinh đứng cuối bảng thành tích, học sinh thứ *hs*, được nhận:
* Nhóm thứ nhất gồm các phương án trong đó học sinh cuối bảng không được nhận thưởng, tức là *pt* phần thưởng chỉ chia cho *hs* - 1 học sinhđầu danh sách và do đó, số cách chia, tức là số phần tử của nhóm này sẽ là: Chia(*pt, hs* - 1).
* Nhóm thứ hai gồm các phương án còn lại, trong đó học sinh cuối bảngcũng được nhận thưởng. Khi đó, do học sinh đứng cuối bảng thành tích được nhận thưởng, thì mọi học sinh khác cũng sẽ có thưởng. Do ai cũng được thưởng nên ta bớt của mỗi người một phần thưởng (để họ lĩnh sau), số phần thưởng còn lại (*pt* - *hs*) sẽ được chia cho *hs* học sinh. Số cách chia khi đó sẽ là Chia(*pt* - *hs, hs*).

Tổng số cách chia cho trường hợp *pt* ≥ *hs* sẽ là tổng số phần tử của hai nhóm, ta có:

Chia(*pt, hs*) = Chia(*pt, hs - 1*) + Chia(*pt - hs, hs*)

Tổng hợp lại, hàm *Chia(pt,hs)* có các tính chất sau đây:

|  |  |
| --- | --- |
| *Điều kiện* | *Chia(pt, hs)* |
| *hs* = *0:* | *0* |
| *pt* = *0 and hs ≠ 0:* | *1* |
| *pt* < *hs:* | *Chia(pt, pt)* |
| *pt* ≥ *hs:* | *Chia(pt,hs–1)+Chia(pt–hs,hs)* |

#### Phương án đệ quy

Ta có phương án đầu tiên của giải thuật Chia như sau:

#### Chương trình

// Chia thuong. Phuong an 1: De quy

#include <bits/stdc++.h>

using namespace std;

// Phuong an de quy

int Chia(int pt, int hs) {

if (hs == 0) return (pt == 0) ? 1 : 0; // ko co hs

if (pt == 0) return 1; // co hs, ko co phan thuong

return (pt < hs) ? Chia(pt,pt) // co hs, co phan thuong < hs

: Chia(pt, hs-1) + Chia(pt-hs,hs);

}

main() {

cout << Chia(7,4); // 11

cout << "\n T h e E n d";

return 0;

}

Phương án này chạy chậm vì phát sinh ra quá nhiều lần gọi hàm trùng lặp. Bảng dưới đây liệt kê số lần gọi hàm Chia khi giải bài toán chia thưởng với 7 phần thưởng và 4 học sinh. Ví dụ, hàm Chia(1,1) sẽ được gọi 9 lần,… Tổng số lần gọi hàm Chia là 79. 79 lần gọi hàm để sinh ra kết quả 11 là quá tốn kém.

Bạn có thể kiểm tra sự tốn kém này bằng cách đặt thêm một biến tổng thể d để đếm số lần gọi hàm Chia.

#### Cải tiến lần 1

Phương án đệ quy khá dễ triển khai nhưng chương trình sẽ chạy rất lâu, bạn hãy thử gọi Chia(66,32) để trải nghiệm được điều trên. Diễn tả đệ quy thường trong sáng, nhàn tản, nhưng khi thực hiện sẽ sinh ra hiện tượng gọi lặp lại những lần gọi hàm. Cải tiến đầu tiên là *tránh những lần gọi lặp* như vậy. Muốn thế, chúng ta tính sẵn các giá trị của hàm *Chia* theo các trị của đầu vào khác nhau và điền vào một mảng hai chiều cc theo công thức:

*cc*[*pt*][*hs*] = *Chia*(*pt, hs*)

Theo các tính chất của hàm hai ngôi *Chia*, ta có

|  |
| --- |
| cc[pt][hs] = Chia(pt, hs) |
| = 0, nếu hs = 0 và pt > 0 |
| = 1, nếu hs > 0 và pt = 0 |
| = cc[pt][pt], nếu 0 < pt < hs |
| = cc[pt][hs-1] + cc[pt-hs][hs], nếu pt ≥ hs > 0 |

Từ đó ta suy ra quy trình điền trị vào bảng hai chiều cc như sau:

Dòng 0, cc[0][\*] = (1,1,…,1): Tại dòng 0 ứng với pt = 0, ta điền toàn 1 vì không có phần thưởng thì có 1 phương án chia là mỗi hs nhận 0 phần thưởng.

Cột 0, cc[\*][0] = (1,0,…,0) Tại cột 0 ta điền toàn 0, vì không có hs mà có pt thì không thể chia hết pt được. Riêng cc[0][0] = 1 theo quy ước.

Cột 1, cc[\*][1] = (1,1,…,1) Tại cột 1 ta điền toàn 1, vì chỉ có một hs nên có bao nhiêu phần thưởng đều dành cho em đó. Như vậy là chỉ có 1 cách chia.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *0* | *1* | *…* | *j* | *…* |  | *hs* |
| *0* | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| *1* | 0 | 1 |  |  |  |  | 1 |
| *…* |  | 1 |  | \* |  |  |  |
| *...* |  | 1 |  | ... |  |  |  |
| *i* |  | 1 | \* | ? |  |  |  |
| *…* |  | 1 |  |  |  |  |  |
| *pt* | 0 | 1 |  |  |  |  |  |

Các phần tử cc[i][j]còn lại sẽ được điền theo cột, từ cột j = 2 đến j = hs. Mỗi cột j sẽ được điền trị như sau:

Giữ nguyên các trị cc[i][j] của cột trước với i < j: cc[i][j] = cc[i][j-1] vì khi số phần thưởng (i) ít hơn hoặc bằng số học sinh (j) thì kết quả sẽ là Chia(i,j) = cc[i][i].

Các giá trị còn lại của cột j sẽ được điền từ trên xuống theo hệ thức

cc[i][j] = cc[i][j-1] + cc[i-j][j], i = j .. m.

#### Cải tiến lần 2

Dùng mảng hai chiều chúng ta chỉ có thể tính toán được với dữ liệu không quá lớn. Bước cải tiến sau đây khá quan trọng: chúng ta dùng mảng một chiều. Giả sử, với kích thước k = m×n = 100, nếu dùng mảng hai chiều thì ta tốn miền nhớ là k2 = 104, nếu dùng mảng một chiều thì ta chỉ phải dùng 100 phần tử, tiết kiệm miền nhớ được k lần.

Quan sát kỹ quy trình gán trị cho mảng hai chiều theo từng cột chúng ta dễ phát hiện ra rằng cột thứ *j* có thể được tính toán từ cột thứ *j* - 1. Hơn nữa, khi tính toán tại bước j ta có, nếu i < j thì

c[i] tại bước j = cc[i][j] = cc[i][j-1] = giá trị c[i] tại bước j-1

nghĩa là giá trị c[i] tại bước j được bảo lưu khi i < j.

Ta có phương án dùng một mảng một chiều c như sau:

#### Chương trình

// Chia thuong. Phuong an 3: Dung mang 1 chieu

#include <bits/stdc++.h>

using namespace std;

const int MN = 71;

int c[MN];

int Chia(int pt, int hs) {

if (hs == 0) return (pt == 0) ? 1 : 0; // ko co hs

if (pt == 0) return 1; // ko co pt

if (pt < hs) hs = pt;

fill(c, c+pt+1, 1); // cot 1 toan 1

for(int j = 2; j <= hs; ++j) // dien theo cot j = 2..hs

// bao luu cac gia tri dau 0..j-1

for(int i = j; i <= pt; ++i) // tinh cac gia tri tu hs..m

c[i] += c[i-j];

return c[pt];

}

main() {

cout << "\n " << Chia(7, 4); // 11

cout << "\n " << Chia(70, 14); // 1614987, 0.06 giay

cout << "\n T h e E n d";

return 0;

}

## Luyện tập Phương pháp quy hoạch động

### Palindrome

*Olympic Tin học Quốc tế, 2000, Bắc Kinh.*

Dãy ký tự s được gọi là đối xứng (palindrome) nếu các phần tử cách đều đầu và cuối giống nhau. Cho dãy s tạo bởi n ký tự gồm các chữ cái hoa và thường phân biệt và các chữ số. Hãy cho biết cần xoá đi từ s ít nhất bao nhiêu ký tự để thu được một dãy đối xứng. Giả thiết rằng sau khi xoá một số ký tự khỏi s thì các ký tự còn lại sẽ tự động xích lại sát nhau.

Ví dụ, với dãy s gồm 9 ký tự, s = "baeadbadb" thì cần xoá ít nhất 4 ký tự, chẳng hạn, các ký tự thứ 4, 6, 7 và 8, để thu được dãy đối xứng chiều dài 5 là "baeab": "baeadbadb" → "baeab".

Dĩ nhiên là có nhiều cách xoá. Tuy nhiên đáp số là số ít nhất các ký tự cần loại bỏ khỏi s là duy nhất và bằng 4.

### Flowers

Olympic Tin học Quốc tế, 1999.

Cần cắm hết k bó hoa khác nhau vào n lọ xếp thẳng hàng sao cho bó hoa có số hiệu nhỏ được đặt trước bó hoa có số hiệu lớn. Với mỗi bó hoa i ta biết giá trị thẩm mĩ khi cắm bó hoa đó vào lọ j là v[i][ j].

Yêu cầu: Xác định một phương án cắm hoa sao cho tổng giá trị thẩm mĩ là lớn nhất.

Dữ liệu vào ghi trong tệp văn bản FLOWER.INP:

Dòng đầu tiên là hai số k và n.

Từ dòng thứ hai trở đi là các giá trị nguyên v[i][ j] trong khoảng 0..10, với i = 1..k và j = 1..n; 1 ≤ k ≤ n ≤ 100.

Dữ liệu ra ghi trong tệp văn bản FLOWER.OUT:

Dòng đầu tiên là tổng giá trị thẩm mĩ của phương án cắm hoa tối ưu.

Từ dòng thứ hai là dãy k số hiệu lọ được chọn cho mỗi bó hoa.

Các số liệu vào và ra đều là số tự nhiên và được ghi cách nhau trên mỗi dòng.

Ví dụ

|  |  |
| --- | --- |
| FLOWER.INP | FLOWER.OUT |
| 4 6  1 1 3 4 3 10  9 1 4 7 2 7  7 2 6 10 2 3  6 10 7 1 3 9 | 24  1 3 4 6 |

*Kết quả cho biết tổng giá trị thẩm mĩ sẽ đạt là 24 (điểm) nếu cắm hoa như sau:*

*- Bó hoa 1 cắm vào lọ 1;*

*- Bó hoa 2 cắm vào lọ 3;*

*- Bó hoa 3 cắm vào lọ 4;*

*- Bó hoa 4 cắm vào lọ 6.*

Dữ liệu input được ghi trong text file FLOWER.INP cho biết:

*Cần cắm hết 4 bó hoa vào 4 trong số 6 lọ.*

*Nếu cắm bó hoa 1*

*vào lọ 1 sẽ đạt độ thẩm mỹ 1,*

*vào lọ 2 sẽ đạt độ thẩm mỹ 1,*

*vào lọ 3 sẽ đạt độ thẩm mỹ 3,*

*vào lọ 4 sẽ đạt độ thẩm mỹ 4,*

*vào lọ 5 sẽ đạt độ thẩm mỹ 3,*

*vào lọ 6 sẽ đạt độ thẩm mỹ 10,*

*Nếu cắm bó hoa 2*

*vào lọ 1 sẽ đạt độ thẩm mỹ 9,*

*vào lọ 2 sẽ đạt độ thẩm mỹ 1,*

*. . .*

### Bố trí phòng học

Câu lạc bộ - Học sinh giỏi Tin học, Hà Nội, năm 2000

Cần bố trí hết k nhóm học sinh vào k trong số n phòng học chuyên đề sao cho nhóm có số hiệu nhỏ được xếp vào phòng có số hiệu nhỏ hơn phòng chứa nhóm có số hiệu lớn. Với mỗi phòng có xếp học sinh, các ghế thừa phải được chuyển ra hết, nếu thiếu ghế thì phải lấy từ kho vào cho đủ mỗi học sinh một ghế. Biết số học sinh trong mỗi nhóm và số ghế trong mỗi phòng. Hãy chọn phương án bố trí sao cho tổng số lần chuyển ghế ra và chuyển ghế vào là ít nhất.

### Bống

Tấm được Bụt trao cho k chú bống. Nhiệm vụ của Tấm là phải thả k bống này vào k trong số n giếng để chăm nuôi, Các giếng được xếp thẳng hàng theo thứ tự 1..n. Bống có số hiệu nhỏ phải được thả vào giếng số hiệu nhỏ. Nếu nuôi bống b trong giếng g thì sau này bống sẽ sinh ra số quà tặng là v(b, g).

Bạn hãy giúp Tấm thả cá vào các giếng để sau này nhận được số quà nhiều nhất.

A fish and a roll of toilet paper

Description automatically generated

Dữ liệu vào ghi trong tệp văn bản BONG.INP:

Dòng đầu tiên là hai trị k và n.

Từ dòng thứ hai trở đi là các giá trị v(b, g) trong khoảng 0..10, với b = 1..k

và g = 1..n; 1 ≤ b ≤ g ≤ 100.

Kết quả hiển thị trên màn hình một phương án thả bống và tổng giá trị quà của phương án tối ưu.

Ví dụ

|  |  |
| --- | --- |
| BONG.INP |  |
| 4 5  1 1 6 4 10  9 1 4 7 7  7 2 6 10 3  6 10 7 1 9 | *Tổng giá trị quà sẽ đạt là 24 (điểm)*  *nếu thả cá như sau:*  *- thả bống 1 vào giếng số 1;*  *- thả bống 2 vào giếng số 3;*  *- thả bống 3 vào giếng số 4;*  *- thả bống 4 vào giếng số 5.* |

A fish and a roll of toilet paper

Description automatically generated

### Tìm các đường ngắn nhất

Cho một đồ thị có hướng gồm n đỉnh mã số từ 1..n với các cung (u, v) có hướng đi từ đỉnh u đến đỉnh v và chiều dài u → v. Viết chương trình tìm mọi đường đi ngắn nhất từ một đỉnh s cho trước tới các đỉnh còn lại của đồ thị.

Dữ liệu vào được ghi trong file văn bản tên DIJ.INP có cấu trúc như sau:

Dòng đầu ghi hai số tự nhiên n và s cách nhau, trong đó n là số lượng đỉnh của đồ thị, s là số hiệu của đỉnh xuất phát 2 ≤ n ≤ 100, 1 ≤ s ≤ n.

Từ dòng thứ hai ghi lần lượt độ dài đường đi từ đỉnh i đến các đỉnh 1, 2,..., n;   
i = 1..n. Giá trị 0 cho biết không có cung nối hai đỉnh tương ứng. Với mọi đỉnh i = 1..n, cung (i, i) được hiểu là không tồn tại và đường đi từ đỉnh i tới chính nó có chiều dài là 0. Các số cùng dòng được ghi cách nhau. Dạng dữ liệu cho như vậy được gọi là ma trận kề của đồ thị.

Ví dụ sau đây cho biết đồ thị có bảy đỉnh, cần tìm các đường đi ngắn nhất từ đỉnh s = 2 tới các đỉnh còn lại của đồ thị.

|  |  |  |
| --- | --- | --- |
| DIJ.INP | A picture containing text, clock, watch  Description automatically generated | Đỉnh 1 không nối với các đỉnh còn lại  Cung (2, 1) có chiều dài 4, cung (2,3) dài 1, …, cung (2,7) dài 5.  … |
| 7 2  0 0 0 0 0 0 0  4 0 1 0 0 0 5  0 0 0 0 0 0 1  0 0 0 0 0 2 0  0 0 0 3 0 0 0  1 0 0 0 0 0 5  0 0 0 1 0 0 0 |

Dữ liệu ra cần ghi trong file văn bản DIJ.OUT gồm n dòng. Thông tin về mỗi đường đi ngắn nhất từ đỉnh s đến mỗi đỉnh còn lại được ghi trên 1 dòng. Số đầu tiên của dòng là chiều dài đường đi. Nếu không tồn tại đường đi thì ghi giá trị 0. Tiếp đến, trong trường hợp có đường đi từ đỉnh s đến đỉnh i thì ghi dãy đỉnh xuất hiện lần lượt trên đường đi, đỉnh đầu tiên là s, đỉnh cuối cùng là i. Đường đi từ đỉnh i tới chính đỉnh đó được coi là không tồn tại, i = 1..n. Thí dụ trên cho ta kết quả

|  |
| --- |
| DIJ.OUT |
| 4 2 1  0  1 2 3  3 2 3 7 4  0  5 2 3 7 4 6  2 2 3 7 |

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 1 có chiều dài 4: 2 → 1.*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 2: không có*

*(thực ra, theo lẽ thường là có đường chiều dài 0).*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 3 có chiều dài 1: 2 → 3.*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 4 có chiều dài 3: 2 → 3 → 7 → 4.*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 5: không có.*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 6 có chiều dài 5: 2→3→7→4→6.*

*- Đường ngắn nhất từ đỉnh 2 đến đỉnh 7 có chiều dài 2: 2→3→7.*