

NGUYỄN XUÂN HUY

DYNAMIC PROGRAMMING

Hà Nội, 2021

MỤC LỤC

Sơ đồ chung	3
Chia thưởng	4
1. Lập hệ thức	4
2. Phương án đệ quy	5
Comment	5
3. Improve 1	7
3. Improve 2	9
Comment	10
Palindrome	11
1. Lập hệ thức quy hoạch động (Phương trình Bellman)	11
Phương án đệ quy	12
2. Improve 1: using 2D array vv[][]	13
3. Improve 2: using 2 array 1D a[] and b[]	14
Problem 9. Bống	16
Đề tương tự 1. Flowers	16
Đề tương tự 2. Club	16
Algorithm	17
2. Phương án đệ quy	17
3. Improve1: using 2D array tt[][]	19
4. Improve1: using array t[]	21

Sơ đồ chung

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 Olympiad 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 cho các bài toán quy hoạch động 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 đó.

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

Sơ đồ giải bài toán quy hoạch động

1. *Lập hệ thức quy hoạch động (phương trình Bellman):*
Lập hệ thức biểu diễn tương quan quyết định của bước đang xử lí với các bước đã xử lí trước đó.
2. *Phương án đệ quy:* Khi đã có hệ thức tương quan chúng ta đã có thể xây dựng ngay một chương trình. Tuy nhiên các hệ thức này thường là các biểu thức đệ quy, do đó dễ gây ra hiện tượng tràn miền nhớ khi ta tổ chức chương trình trực tiếp bằng đệ quy vì có những hàm được gọi nhiều lần.
3. *Cải tiến 1:* Tìm cách khử đệ quy bằng một mảng hai chiều, nếu hệ thức quy hoạch động là một hàm hai tham biến.
4. *Cải tiến 2:* Tiếp tục cải tiến chương trình bằng cách thu gọn hệ thức quy hoạch động và giảm kích thước miền nhớ, thay mảng nhiều chiều bằng các mảng ít chiều hơn.

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.

$$1 \leq m, n \leq 70.$$

Hãy tính số cách chia.

Thí 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	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	3	3	1	0
8	3	2	2	0
9	4	1	1	1
10	3	2	1	1
11	2	2	2	1

1. Lập hệ thức

Gọi $\text{Chia}(i, j)$ là số cách chia i phần thưởng cho j học sinh, ta thấy:

Nếu không có học sinh nào ($j = 0$) thì không có cách chia nào ($\text{Chia} = 0$).

Nếu không có phần thưởng nào ($i = 0$) thì chỉ có một cách chia ($\text{Chia}(0, j) = 1$ - mỗi học sinh nhận 0 phần thưởng).

Ta cũng quy ước

$$\text{Chia}(0, 0) = 1.$$

Nếu số phần thưởng ít hơn số học sinh ($i < j$) thì trong mọi phương án chia, từ học sinh thứ $i + 1$ trở đi sẽ không được nhận phần thưởng nào:

$$\text{Chia}(i, j) = \text{Chia}(i, i) \text{ nếu } i < j.$$

Ta xét tất cả các phương án chia trong trường hợp $i \geq j$. 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 j , được nhận:

Nhóm thứ nhất gồm các phương án trong đó học sinh thứ j không được nhận thưởng, tức là i phần thưởng chỉ chia cho $j - 1$ học sinh và do đó, số cách chia, tức là số phần tử của nhóm này sẽ là:

$$\text{Chia}(i, j - 1).$$

Nhóm thứ hai gồm các phương án trong đó học sinh thứ j cũng 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 ($i - j$) sẽ được chia cho j học sinh. Số cách chia khi đó sẽ là

$$\text{Chia}(i - j, j).$$

Tổng số cách chia cho trường hợp $i \geq j$ sẽ là tổng số phần tử của hai nhóm, ta có:

$$\text{Chia}(i, j) = \text{Chia}(i, j - 1) + \text{Chia}(i - j, j).$$

Tổng hợp lại ta có:

Các tính chất của hàm Chia(m, n) Chia m phần thưởng cho n học sinh	
Điều kiện	Chia(i, j)
$n = 0$:	$\text{Chia}(m, n) = 0$
$m = 0$ and $n > 0$:	$\text{Chia}(m, n) = 1$
$0 < m < n$:	$\text{Chia}(m, n) = \text{Chia}(m, m)$
$m \geq n > 0$:	$\text{Chia}(m, n) = \text{Chia}(m, n - 1) + \text{Chia}(m - n, n)$

2. Phương án đệ quy

```
// Recursive 1
#include <iostream>

using namespace std;

// m awards, n students
int Chia(int m, int n) {
    if (n == 0) return 0;
    // n > 0
    if (m == 0) return 1;
    if (m < n) return Chia(m, m);
    else return Chia(m, n-1) + Chia(m-n, n);
}

main() {
    cout << Chia(7, 4); // 11
    cout << "\n T h e   E n d";
    return 0;
}
```

Comment

học sinh

	0	1	2	3	4
0	0	9	1	1	0
1	9	9	2	1	0
2	6	6	1	0	0
3	5	5	2	1	1
4	3	3	1	1	0
5	2	2	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

Phương án này chạy chậm vì phát sinh ra quá nhiều lần gọi hàm trùng lặp. Trong bảng liệt kê số lần gọi hàm Chia khi giải bài toán chia thưởng với bảy phần thưởng ($m = 7$) và 4 học sinh ($n = 4$). Thí 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 lại 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 hàm đệ quy.

Test

```
// Recursive 2

#include <iostream>

using namespace std;

int call; // so lan goi ham

// m awards, n students
int Chia(int m, int n) {
    ++call;
    if (n == 0) return 0;
    // n > 0
    if (m == 0) return 1;
    if (m < n) return Chia(m,m);
    else return Chia(m, n-1) + Chia(m-n,n);
}

void Test(int m, int n) {
    cout << "\n\n Test m = " << m << "   n = " << n;
    call = 0;
    cout << "\n Result: " << Chia(m, n);
    cout << "\n so lan call " << call;
}

main() {
    Test(7, 4);
    Test(66, 32);
    cout << "\n T h e   E n d";
    return 0;
}
```

Result

```
Test m = 7   n = 4
Result: 11
so lan call 79
```

```

Test m = 66  n = 32
Result: 2269557
so lan call 33089827
T h e   E n d

```

3. Improve 1

Cải tiến đầu tiên là tránh những lần gọi lặp như vậy. Muốn thế chúng ta tính sẵn các giá trị của hàm theo các trị của đầu vào khác nhau và điền vào một mảng hai chiều cc với ý nghĩa $cc[m][n] = \text{Chia}(m,n)$.

Muốn khử đệ quy ta cần biết trật tự điền trị cho mảng cc.

Ta khảo sát kỹ thuật toán đệ quy $\text{Chia}(m,n)$ và thử chuyển đổi sang dạng điền trị cho mảng cc:

- | | | |
|--|---|---|
| 1. int Chia(int m, int n) | → | $cc[m][n]$ |
| 2. if (n == 0) return 0; | → | if (n == 0) $cc[m][n] = 0$; |
| 3. if (m == 0) return 1; | → | if (m == 0) $cc[m][n] = 0$; |
| 4. if (m < n) return Chia(m,m); | → | if (m < n) $cc[m][n] = cc[m][m]$; |
| 5. else return Chia(m, n-1) + Chia(m-n,n); | → | else $cc[m][n] = cc[m][n-1] + cc[m-n][n]$; |

Các dòng lệnh 4 và 5 của bảng cho ta biết muốn điền trị tại ô $cc[m][n]$ phải biết trước giá trị của hai ô $cc[m][n-1]$ và $cc[m-n][n]$ tức là mảng hai chiều cc phải được điền theo cột từ cột trái qua cột phải; trên mỗi cột phải điền theo dòng cột từ trên xuống.

			n-1	n			
m-n				*			
m			**	?			

Quy trình điền trị vào mảng 2 chiều cc[][]

Ta quy ước $cc[i, j]$ chứa số cách chia i phần thưởng cho j học sinh.

Theo phân tích của phương án 1, ta có:

```

✂ cc[0][0] = 1; cc[i, 0] = 0, i = 1:m.
✂ cc[i][j] = cc[i][i], nếu i < j
✂ cc[i][j] = cc[i][j-1] + cc[i-j][j], nếu i ≥ j.

```

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

✂ Khởi trị

```

cc[0][0] = 1;
cc[i][0] = 0, i = 1:m;

```

✂ Điền bảng: Lần lượt điền theo từng cột j = 1..n.
Tại mỗi cột j:

```

cc[i][j] = cc[i][i], i = 0:j-1 (i < j);
cc[i][j] = cc[i][j-1] + cc[i-j][j], i = j:m (i ≥ j);

```

✂ Nhận kết quả: cc[m, n].

Program

```

// Improve 1: using 2D array cc[][]
#include <iostream>

using namespace std;

// m awards, n students
int Chia(int m, int n) {
    if (m < n) n = m;
    int cc[m+1][n+1];
    cc[0][0] = 1;
    for (int i = 1; i <= m; ++i) cc[i][0] = 0;
    for (int j = 1; j <= n; ++j) { // column j
        for (int i = 0; i < j; ++i) {
            cc[i][j] = cc[i][i];
        }
        for (int i = j; i <= m; ++i) {
            cc[i][j] = cc[i][j-1] + cc[i-j][j];
        }
    }
    return cc[m][n];
}

void Test(int m, int n) {
    cout << "\n\n Test m = " << m << "   n = " << n;
    cout << "\n Result: " << Chia(m, n);
}

main() {
    Test(7, 4);
    Test(66, 32);
    cout << "\n T h e   E n d";
    return 0;
}

```


Test

```
Test m = 7   n = 4
Result: 11

Test m = 66  n = 32
Result: 2269557
T h e   E n d
```

3. Improve 2

Dùng mảng hai chiều chúng ta chỉ có thể tính toán được với dữ liệu nhỏ. Bước cải tiến sau đây khá quan trọng: chúng ta dùng mảng một chiều. Quan sát kĩ quy trình gán trị cho mảng hai chiều theo từng cột chúng ta dễ phát hiện ra rằng

Tại bước thứ j ta có

```
c[i] = c[i][j] , i = 0:m
```

Từ đó ta suy ra quy trình điền trị vào mảng 1D $c[]$ như sau:

✂ Khởi trị với $n = 0$

```
c[0] = 1;
c[i] = 0, i = 1:m;
```

✂ Điền bảng cho cột $j = 1..n$.

```
c[i] = c[i] + c[i-j] , i = j:m (i ≥ j);
```

✂ Nhận kết quả: $c[m]$.

Program

```
// Improve 2: using 1D array c[]

#include <iostream>

using namespace std;

// m awards, n students
int Chia(int m, int n) {
    if (m < n) n = m;
    int c[m+1];
    c[0] = 1;
    for (int i = 1; i <= m; ++i) c[i] = 0;
    for (int j = 1; j <= n; ++j) { // column j
        for (int i = j; i <= m; ++i) {
            c[i] += c[i-j];
        }
    }
    return c[m];
}
```

```

void Test(int m, int n) {
    cout << "\n\n Test m = " << m << "   n = " << n;
    cout << "\n Result: " << Chia(m, n);
}

main() {
    Test(7, 4); // 11
    Test(66, 32); 2269557
    cout << "\n T h e   E n d";
    return 0;
}

```

Comment

Bài toán trên còn có cách phát biểu khác như sau: Hãy tính số cách biểu diễn số tự nhiên m thành tổng của n số tự nhiên sắp theo trật tự không tăng. Thí dụ, với $m = 7$, $n = 4$ ta có 11 cách biểu diễn như sau:

```

7  = 7 + 0 + 0 + 0 = 6 + 1 + 0 + 0 = 5 + 2 + 0 + 0 = 5 + 1 + 1 + 0 =
   = 4 + 3 + 0 + 0 = 4 + 2 + 1 + 0 = 4 + 1 + 1 + 1 = 3 + 3 + 1 + 0 =
   = 3 + 2 + 2 + 0 = 3 + 2 + 1 + 1 = 2 + 2 + 2 + 1 = .

```

Palindrome

Olympic Quốc tế, năm 2000, Bắc Kinh, Trung Quốc.

Dãy kí 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 kí tự gồm các chữ cái hoa và thường phân biệt và các chữ số. Hãy cho biết cần xoá đi từ s ít nhất là bao nhiêu kí tự để thu được một dãy đối xứng. Giả thiết rằng sau khi xoá bớt một số kí tự từ s thì các kí tự còn lại sẽ tự động xích lại sát nhau.

Dữ liệu vào ghi trong tệp văn bản **PALIN.INP** với cấu trúc như sau:

PALIN.INP	PALIN.OUT
9	4
baeadbab	

Dòng đầu tiên là giá trị n , $1 \leq n \leq 1000$.

Dòng thứ hai là n kí tự của dãy viết liền nhau.

Chỉ số của string s được tính từ 0.

Dữ liệu ra ghi trong tệp văn bản **PALIN.OUT**: số lượng kí tự cần xoá.

Thí dụ, với dãy s gồm 9 kí tự, $s = \text{'baeadbab'}$ thì cần xoá ít nhất 4 kí tự, chẳng hạn, các kí tự thứ 4, 6, 7 và 8 sẽ thu được dãy đối xứng chiều dài 5 là baeab :

baeadbab \rightarrow baeab

Dĩ nhiên là có nhiều cách xoá. Thí dụ, có thể xoá các kí tự thứ 1, 2, 3 và 5 từ dãy s để thu được dãy con đối xứng khác là bdadb với cùng chiều dài 5:

baeadbab \rightarrow bda**db**

Tuy nhiên đáp số là số ít nhất các kí tự cần loại bỏ khỏi s thì là duy nhất và bằng 4.

Bài giải

Bài toán này đã được nhiều bạn đọc công bố lời giải với một mảng hai chiều kích thước n^2 hoặc vài ba mảng một chiều kích thước n , trong đó n là chiều dài của dữ liệu vào.

Với một nhận xét nhỏ ta có thể phát hiện ra rằng chỉ cần dùng một mảng một chiều kích thước n và một vài biến đơn là đủ.

Gọi dãy dữ liệu vào là s . Ta tìm chiều dài của dãy con đối xứng v dài nhất trích từ s . Khi đó số kí tự cần xoá từ s sẽ là $t = \text{length}(s) - \text{length}(v)$. Dãy con ở đây được hiểu là dãy thu được từ s bằng cách xoá đi một số phần tử trong s . Thí dụ với dãy $s = \text{baeadbab}$ thì dãy con đối xứng dài *nhất* của s sẽ là baeab hoặc bdadb , ... Các dãy này đều có chiều dài 5.

1. Lập hệ thức quy hoạch động (Phương trình Bellman)

Gọi $p(i, j)$ là chiều dài của dãy con dài nhất thu được khi giải bài toán với dữ liệu vào là đoạn $s[i:j]$. Khi đó $p(1, n)$ là chiều dài của dãy con đối xứng dài nhất trong dãy n kí tự $s[1:n]$ và do đó số kí tự cần loại bỏ khỏi dãy $s[1:n]$ sẽ là $n - p(1, n)$.

Đó chính là đáp số của bài toán.

Ta liệt kê một số tính chất quan trọng của hàm hai biến $p(i, j)$. Ta có:

Tính chất 1.

Nếu $i > j$, tức là chỉ số đầu trái lớn hơn chỉ số đầu phải, ta quy ước đặt $p(i, j) = 0$.

Tính chất 2.

Nếu $i = j$ thì $p(i, j) = 1$ vì dãy khảo sát chỉ chứa đúng 1 kí tự nên nó là đối xứng.

Tính chất 3.

Nếu $i < j$ và $s[i] = s[j]$ thì $p(i, j) = p(i + 1, j - 1) + 2$. Vì hai kí tự đầu và cuối dãy $s[i..j]$ giống nhau nên chỉ cần xác định chiều dài của dãy con đối xứng dài nhất trong đoạn giữa là $s[i + 1, j - 1]$ rồi cộng thêm 2 đơn vị ứng với hai kí tự đầu và cuối dãy là được.

Tính chất 4.

Nếu $i < j$ và $s[i] \neq s[j]$, tức là hai kí tự đầu và cuối của dãy con $s[i..j]$ là khác nhau thì ta khảo sát hai dãy con là $s[i:(j - 1)]$ và $s[(i + 1):j]$ để lấy chiều dài của dãy con đối xứng dài nhất trong hai dãy này làm kết quả:

$$p(i, j) = \max(p(i, j - 1), p(i + 1, j))$$

Vấn đề đặt ra là cần tính $p(1, n)$. Mà muốn tính được $p(1, n)$ ta phải tính được các $p(i, j)$ với mọi $i, j = 1:n$.

Phương án đệ quy

Các tính chất của hàm $P(i, j)$ cho ta một phương án cài đặt ban đầu sau đây:

```
int P(string &s, int i, int j) {
    if (i > j) return 0;
    if (i == j) return 1;
    return (s[i] == s[j]) ? P(s, i+1, j-1) + 2
        : max(P(s, i+1, j), P(s, i, j-1));
}
```

Program

```
// Recursive
#include <iostream>

using namespace std;

int P(string &s, int i, int j) {
    if (i > j) return 0;
    if (i == j) return 1;
    return (s[i] == s[j]) ? P(s, i+1, j-1) + 2
        : max(P(s, i+1, j), P(s, i, j-1));
}

int Pal(string & s) {
    int n = s.length();
    return n - P(s, 0, n-1);
}

void Test() {
    string s = "baeacghabxbahgcaeab"; // doi xung, ko xoa
```

```

cout << "\n " << s << " : " << Pal(s);
s = "ba1eac2ghabxbahgca3eab"; // xoa 3
cout << "\n " << s << " : " << Pal(s);
}

main() {
    Test();
    cout << "\n T h e   E n d";
    return 0;
}

```

2. Improve 1: using 2D array vv[][]

Gọi đệ quy sẽ phát sinh các lời gọi hàm trùng lặp như đã phân tích trong bài toán Chia thưởng. Ta khắc phục điều này bằng cách sử dụng một mảng hai chiều vv để tính trước các giá trị của hàm $p(i, j)$, cụ thể là:

$$vv[i][j] = p(i, j)$$

Ta gắng tìm cách điền mảng vv sao cho mỗi hàm p được gọi không quá 1 lần.

Phỏng theo phương án đệ quy ta tính được:

```

vv[i][j] = (s[i] == s[j]) ? vv[i+1][j-1] + 2
              : max(vv[i+1][j], vv[i][j-1]);

```

Biểu thức trên cho ta thấy:

Muốn tính được giá trị $vv[i][j]$ thì phải biết các giá trị $vv[i+1][j-1]$, $vv[i+1][j]$ và $vv[i][j-1]$.

Từ đó ta suy ra:

Mảng vv phải được điền từ dòng dưới lên dòng trên. Mỗi dòng phải được điền từ trái qua phải. Các ô $vv[i][j]$ tại nửa dưới đường chéo chính có $i > j$ nên nhận trị 0, các ô tại đường chéo chính có $i = j$ nên được nhận giá trị 1.

		j-1	j	
i		+	?	
i+1		+2	+	

Program

```

// Improve 1: using 2D array vv

#include <iostream>
#include <windows.h>

using namespace std;

string s;
int n;

int Pal() {
    n = s.length();
    int vv[n][n];
    memset(vv, 0, sizeof(vv));
    for (int i = 0; i < n; ++i) vv[i][i] = 1;
    for (int i = n-1; i >= 0; --i) {
        for (int j = i+1; j < n; ++j) {
            vv[i][j] = (s[i] == s[j]) ? vv[i+1][j-1] + 2
                                     : max(vv[i][j-1], vv[i+1][j]);
        }
    }
    return n - vv[0][n-1];
}

void Test() {
    s = "baeacghabxbahgcaeab"; // doi xung, ko xoa
    cout << "\n " << s << " : " << Pal();
    s = "ba1eac2ghabxbahgca3eab"; // xoa 3
    cout << "\n " << s << " : " << Pal();
}

main() {
    Test();
    cout << "\n T h e   E n d";
    return 0;
}

```

3. Improve 2: using 2 array 1D a[] and b[]

Một mảng 2D có kích thức $1000^2 = 1M$. Nếu ta dùng 2 mảng 1D sẽ chỉ tốn kích thước 2K tức là tiết kiệm không gian 500 lần. Ta gọi a là dòng dưới, b là dòng trên. Mỗi lần ta xử lý dòng b theo dữ liệu thu được tại dòng trước đó là a. Sau mỗi nhịp ta hoán vị a và b,

Program

```

// Improve 2: using 2 array 1D a[], b[]
#include <iostream>
#include <windows.h>

using namespace std;

string s;
int n;

```

```

int Pal() {
    n = s.length();
    int *a = new int[n];
    int *b = new int[n];
    int *c;
    memset(a, 0, n*sizeof(int)); // row n-1
    a[n-1] = 1;
    for (int i = n-2; i >= 0; --i) {
        b[i] = 1;
        for (int j = i+1; j < n; ++j) {
            b[j] = (s[i] == s[j]) ? a[j-1] + 2
                : max(b[j-1], a[j]);
        }
        c = a; a = b; b = c;
    }
    int res = n - a[n-1];
    delete [] a; delete [] b;
    return n - a[n-1];
}

void Test() {
    s = "baeacghabxbahgcaeab"; // doi xung, ko xoa
    cout << "\n " << s;
    cout << "\n Delete: " << Pal();

    s = "ba1eac2ghabxbahgca3eab"; // xoa 3
    cout << "\n\n " << s;
    cout << "\n Delete: " << Pal();
}

main() {
    Test();
    cout << "\n T h e   E n d";
    return 0;
}

```

Result

```

baeacghabxbahgcaeab
Delete: 0

ba1eac2ghabxbahgca3eab
Delete: 3
T h e   E n d

```

Problem 9. Bồng

Tấm được Bụt trao cho k chú bồng. Nhiệm vụ của Tấm là phải thả k bồng này vào n giếng để chăm nuôi. Các giếng được xếp thẳng hàng theo thứ tự $1:n$. Bồng có số hiệu nhỏ phải được thả vào giếng số hiệu nhỏ. Nếu nuôi bồng b trong giếng g thì sau này bồng 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.



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 $g = 1:n$; $1 \leq b \leq g \leq 100$.

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

Ví dụ

BONG.INP									
4	6								
<u>1</u>	1	6	4	3	10				
9	1	<u>4</u>	7	2	7				
7	2	6	<u>10</u>	2	3				
6	10	7	1	3	<u>9</u>				

Kết quả cho biết tổng giá trị quà sẽ đạt là 24 (điểm)
nếu thả cá như sau:

- ☞ thả bồng 1 vào giếng số 1;
- ☞ thả bồng 2 vào giếng số 3;
- ☞ thả bồng 3 vào giếng số 4;
- ☞ thả bồng 4 vào giếng số 6.

Đề tương tự 1. Flowers

Olympic Quốc tế năm 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 mỹ 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 mỹ là lớn nhất.

Đề tương tự 2. Club

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

Cần bố trí 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 nhận 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.

Algorithm

1. Lập hệ thức quy hoạch động

Gọi $T(i, j)$ là tổng giá trị quà giải bài toán với i chú bóng 1: i và j giềng 1: j , tức là tổng giá trị quà thu được khi thả hết i chú bóng vào j giềng đầu tiên. Ta thấy:

✧ Nếu không có cá thì không phải làm gì và dĩ nhiên là không có quà:

$$T(0, j) = 0$$

✧ Nếu số cá nhiều hơn số giềng $i > j$ thì không có cách nào, vì đầu bài yêu cầu phải thả hết cá, mỗi cá vào đúng một giềng. Ta có:

$$T(i, j) = 0, \text{ if } i > j$$

✧ Nếu số cá bằng số giềng ($i = j$) thì chỉ có một cách làm là thả mỗi cá c vào giềng c :

$$T(i, j) = \text{sum} \{v(c, c), c = 1:k\}, \text{ if } i = j$$

✧ Ta xét trường hợp số cá ít hơn hẳn số giềng ($i < j$). Có hai tình huống: giềng cuối cùng, tức là giềng thứ j được chọn cho phương án tối ưu và giềng thứ j không được chọn:

- Nếu giềng cuối cùng, giềng thứ j được chọn để thả chú bóng (cuối cùng) i thì $i-1$ chú bóng đầu tiên sẽ được phân phối vào $j-1$ giềng đầu tiên. Tổng giá trị quà khi đó sẽ là

$$T(i-1, j-1) + v[i][j], \text{ if } i < j$$

- Nếu giềng thứ j không được chọn cho phương án tối ưu, thì i chú bóng phải được thả vào $j-1$ giềng đầu tiên và do đó tổng giá trị quà sẽ là

$$T(i, j-1).$$

Vậy trường hợp $i < j$ cho có giá trị tối ưu khi thả hết i cá vào j giềng sẽ là

$$T(i, j) = \max \{T(i-1, j-1) + v[i][j], T(i, j-1)\}$$

2. Phương án đệ quy

Ta tạm thời xây dựng phương án đệ quy tính giá trị quà.

Program

```
#include <iostream>
#include <fstream>

using namespace std;

const int MN = 101;
int k; // bong
int n; // gieng
int v[MN][MN]; // qua
int qua;

void Go() {
```

```

    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Print(const char * msg = "") {
    cout << msg;
    for (int i = 1; i <= k; ++i) {
        cout << "\n ";
        for (int j = 1; j <= n; ++j) {
            cout << " " << v[i][j];
        }
    }
}

void Read() {
    ifstream f("BONG.INP");
    f >> k >> n;
    for (int i = 1; i <= k; ++i) {
        for (int j = 1; j <= n; ++j) {
            f >> v[i][j];
        }
    }
    f.close();
}

int SumDiagonal(int j) {
    int s = 0;
    for (int i = 1; i <= j; ++i)
        s += v[i][i];
    return s;
}

// Recursive
int T(int i, int j) {
    if (i == 0 || j == 0 || i > j) return 0;
    if (i == j) return SumDiagonal(j);
    // i < j
    return max(T(i-1, j-1) + v[i][j], T(i, j-1));
}

void Run() {
    Read();
    cout << "\n k = " << k << "   n = " << n;
    Print();
    cout << "\n " << T(k, n);
}

main() {
    Run();
    cout << "\n T h e   E n d";
    return 0;
}

```

result

```

k = 4  n = 6
1 1 6 4 3 10
9 1 4 7 2 7
7 2 6 10 2 3
6 10 7 1 3 9
24
T h e   E n d

```

3. Improve1: using 2D array tt[][]

Ta khử đệ quy bằng cách dùng một mảng 2 chiều tt để lưu các giá trị của hàm T(i,j);

```
t[i][j] = T(i,j)
```

Dựa theo hệ thức đệ quy ta thu được:

```

if (i == 0 || j == 0 || i > j) return 0;
if (i == j) return SumDiagonal(j);
tt[i][j] = max(tt[i-1][j-1] + v[i][j], tt[i][j-1]);

```

Như vậy mảng tt sẽ được điền trị như sau:

Dòng 0 điền toàn 0 ứng với số bóng = 0
 Cột 0 điền toàn 0 ứng với số giếng = 0
 Tam giác dưới của tt: điền toàn 0, ứng với số bóng > số giếng
 chéo chính: tt[i][i] = sum{v[c][c], c = 1:i}, ứng với số bóng = số giếng
 Phần còn lại: điền theo cột j = 1: n, trên mỗi cột điền theo dòng i = 1:j-1
 Kết quả: tt[k][n].

tt			j-1	j			
m-n							
i-1			*				
i			+v[i][j]	?			

Program

```

#include <iostream>
#include <fstream>
#include <windows.h>

using namespace std;

```

```

const int MN = 101;
int k; // bong
int n; // gieng
int v[MN][MN]; // qua
int tt[MN][MN];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Print(const char * msg = "") {
    cout << msg;
    for (int i = 1; i <= k; ++i) {
        cout << "\n ";
        for (int j = 1; j <= n; ++j) {
            cout << " " << v[i][j];
        }
    }
}

void Read() {
    ifstream f("BONG.INP");
    f >> k >> n;
    for (int i = 1; i <= k; ++i) {
        for (int j = 1; j <= n; ++j) {
            f >> v[i][j];
        }
    }
    f.close();
}

int Fish() {
    if (k == 0 || n == 0 || k > n) return 0;
    memset(tt, 0, sizeof(tt));
    int cc = 0; // cheo chinh
    for (int i = 1; i <= k; ++i) {
        cc += v[i][i];
        tt[i][i] = cc;
    }
    if (k == n) return tt[k][k];
    // tt[i][j] = max(tt[i-1][j-1] + v[i][j], tt[i][j-1]);
    for (int j = 2; j <= n; ++j) {
        for (int i = 1; i < j; ++i) {
            tt[i][j] = max(tt[i-1][j-1] + v[i][j], tt[i][j-1]);
        }
    }
    return tt[k][n];
}

void Run() {
    Read();
    cout << "\n k = " << k << "  n = " << n;
    Print();
}

```

```

    cout << "\n " << Fish();
}

main() {
    Run();
    cout << "\n T h e   E n d";
    return 0;
}

```

4. Improve1: using array t[]

Nếu ta dùng mảng một chiều t thì phải thay đổi chút ít về trật tự điền mảng, cụ thể là ta không được điền vào phần tử sẽ còn được sử dụng. Quan sát kỹ bảng minh họa việc điền trị bạn sẽ phát hiện ra trật tự sau cho mảng t:

```

Khởi trik t toàn 0 ứng với số giếng = 0
Với số giếng j = 1:n, sửa t tại bước j-1 (j-1 giếng) thành t tại bước j (j
giếng) như sau:
    Với số cá i = j:1 tính
        t[i] = max(t[i-1]) + v[i][j], t[i])
Kết quả: t[k].

```

Program

```

#include <iostream>
#include <fstream>
#include <windows.h>

using namespace std;

const int MN = 101;
int k; // bong
int n; // gieng
int v[MN][MN]; // qua
int t[MN];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Print(const char * msg = "") {
    cout << msg;
    for (int i = 1; i <= k; ++i) {
        cout << "\n ";
        for (int j = 1; j <= n; ++j) {
            cout << " " << v[i][j];
        }
    }
}

void Read() {

```

```

ifstream f("BONG.INP");
f >> k >> n;
for (int i = 1; i <= k; ++i) {
    for (int j = 1; j <= n; ++j) {
        f >> v[i][j];
    }
}
f.close();
}

int Fish() {
    if (k == 0 || n == 0 || k > n) return 0;
    memset(t, 0, sizeof(t));
    for (int j = 1; j <= n; ++j) {
        for (int i = j; i > 0; --i) {
            t[i] = max(t[i-1] + v[i][j], t[i]);
        }
    }
    return t[k];
}

void Run() {
    Read();
    cout << "\n k = " << k << "   n = " << n;
    Print();
    cout << "\n " << Fish();
}

main() {
    Run();
    cout << "\n T h e   E n d";
    return 0;
}

```