NGUYỄN XUÂN HUY

The Greedy Method

HÀ NỘI – 2021

MỤC LỤC

Idea

Xác định một trật tự xử lý để thu được kết quả tốt nhất.

Trật tự có thể là:

- *Lấy min*
- *Lấy max*
- *Đan xen min max*
- *...*

Problem 1. Máy phát nhạc

Băng nhạc ghi tuần tự n bản nhạc mã số 0..n–1với thời lượng phát khác nhau t_i. Mỗi ngày tần suất phát mỗi bản nhạc coi như bằng nhau. Để phát bản nhạc thứ I máy bỏ qua i–1 bài đầu tiên rồi phát bài i. Phát xong mỗi bài đầu đọc chuyển về đầu băng nhạc. Cần ghi các bài để tổng thời gian hoạt động của máy là nhỏ nhất.

Cách ghi 1

Bài	*0*	*1*	*2*	*3*	sum
t	**6**	**3**	**2**	**4**	
Phát bài 0	6				**6**
Phát bài 1	6	3			**9**
Phát bài 2	6	3	2		**11**
Phát bài 3	6	3	2	4	**15**
				Total	**41**

Cách ghi 2

Bài	*2*	*1*	*3*	*0*	sum
t	**2**	**3**	**4**	**6**	
Phát bài 0	2	3	4	6	**15**
Phát bài 1	2	3			**5**
Phát bài 2	2				**2**
Phát bài 3	2	3	4		**9**
				Total	**31**

```
#include <iostream>
#include <fstream>
//#include <bitset>
//#include <cmath>
//#include <windows.h>
//#include <cstring>
#include <bits/stdc++.h>
//#include <time.h>
//#include <algorithm>

using namespace std;

#define Open() ifstream f("MUSIC.INP")
#define Close() f.close()

int *t, *id;

bool Less(int i, int j) {
   return t[i] < t[j];
}

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
```

```
  }
}

int Music() {
  Open();
  int n; // so ban nhac
  f >> n;
  cout << "\n n = " << n;
  t = new int[n]; id = new int[n];
  for (int i = 0; i < n; ++i) {
     f >> t[i];
     id[i] = i;
  }

  Print(t, 0, n-1, "\n t: ");
  Print(id, 0, n-1, "\n id: ");
 sort(id, id+n, Less);
  Close();
  Print(id, 0, n-1, "\n Ghi cac ban nhac nhu sau:");
}

main() {
   Music();
 // --------------------------------
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}
```

Result n = 4

```
 t:  6 3 2 4
 id:  0 1 2 3
 Ghi cac ban nhac nhu sau: 2 1 3 0
 T h e   E n d
```

Độ phức tạp tính toán. Săp tăng dãy n phần tử cần n^2 phép so sánh. Nếu dùng các giải thuật nhanh thì cần nlog(n) phép so sánh.

Problem 2. Int To Rome

Chuyển đổi số nguyên dương sang dạng số La Mã.

```
IntToRome(2021) = "MMXXI"
IntToRome(3964) = "MMMCMLXIV"
```

Algorithm Rót nước

Program

```
/**********************************
 Name: INITOROME.CPP
 Copyright: (C) 2021
 Author: DevcFan
 Date: 11/10/21 10:06
 Description:
 int to Rome
 IntToRome(2021) = "MMXXI"
  IntToRome(3964) = "MMMCMLXIV"
**********************************/

#include <iostream>
// #include <fstream>
//#include <bitset>
//#include <cmath>
//#include <windows.h>
//#include <cstring>
#include <bits/stdc++.h>
//#include <time.h>
//#include <algorithm>

using namespace std;

const int MN = 13;
// Roles 49 AB = B-A; AB = A+B
int ival[] =    {1000,900,500,400,100,90,   50,  40,  10,  9,  5,   4,1};
string rval[] = {"M","CM","D","CD","C","XC","L",
"XL","X","IX","V","IV","I"};

void Go() {
  cout << " ? ";
```

```
  fflush(stdin);
  if (cin.get() == '.') exit(0);
}

string IntToRome(int b) {
  string r = "";
  for (int i = 0; i < MN; ++i) {
     while (b >= ival[i]) {
         b -= ival[i];
         r += rval[i];
     }
  }
  return r;
}

main() {
  int n = 2021;
  cout << "\n " << n << " -> " <<  IntToRome(n); // "MMXXI"
  n = 3964;
  cout << "\n " << n << " -> " << IntToRome(3964); // "MMMCMLXIV"
  for (int i = 1; i <= 4000; ++i) {
     cout << "\n " << i << " -> " << IntToRome(i); Go();
 }
 // --------------------------------
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}
```

Result

```
 2021 -> MMXXI
 3963 -> MMMCMLXIV
 1 -> I ?

 2 -> II ?

 3 -> III ?

 4 -> IV ?
```

```
5 -> V ?

6 -> VI ?

7 -> VII ?
```

Problem 3. Ba lô

Bài toán ba lô có nhiều phiên bản. Mục này trình bày một phiên bản phù hợp với phương pháp tham.

Có n vật mã số từ 0 đến n–1, trọng lượng lần lượt là $w = (w_0, w_0, \dots, w_{n-1})$. *Mỗi vật i có giá trị* v_i , $0 <= i < n$. *Cho một ba lô có sức mang tối đa m. Cần chọn cả hay một phần của mỗi vật để xếp vào ba lô sao cho tổng giá trị thu được là lớn nhất.*

Algorithm Greedy

$x_i = v_i/w_i$ là đơn giá của vật i. Ta ưu tiên cho những vật có đơn giá cao.

Lần lượt xét vật i có đơn giá cao trở xuống

Nếu ba lô còn xếp được thì xếp.

Program

```
#include <iostream>
#include <fstream>
//#include <bitset>
//#include <cmath>
//#include <windows.h>
//#include <cstring>
#include <bits/stdc++.h>
//#include <time.h>
//#include <algorithm>

using namespace std;

#define Open() ifstream f("BALO.INP")
#define Close() f.close()

int n;
double m; // suc mang cua balo
double *w; // trong luong
double *v; // gia tri
int *id;
double *x;

void Go() {
  cout << " ? ";
  fflush(stdin);
```

```
  if (cin.get() == '.') exit(0);
}

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}

void Print(double x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}

void Print(double x[], int id[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[id[i]];
   }
}

bool Great(int i, int j) {
   return x[i] > x[j];
}

int Balo() {
  Open();
  f >> n >> m;
  cout << "\n n = " << n << " m = " << m;
  w = new double[n]; // trong luong
 v = new double[n];  // gia tri
 id = new int[n];
 x = new double[n]; // don gia
 for (int i = 0; i < n; ++i) {
     f >> w[i]; // trong luong
```

```
   id[i] = i;
}
for (int i = 0; i < n; ++i) {
   f >> v[i]; // gia tri
   x[i] = v[i] / w[i]; // don gia
}
Close();
Print(w, 0, n-1, "\n w: ");
Print(v, 0, n-1, "\n v: ");
Print(x, 0, n-1, "\n x: ");
Print(id, 0, n-1, "\n id: ");
sort(id, id+n, Great); // sap giam theo x
Print(id, 0, n-1, "\n dec sorted by id: ");
Print(x, id, 0, n-1, "\n x: ");
double vsum = 0;
double *r = new double[n]; // result
double rm = m;
memset(r, 0, n*sizeof(double)); // r = all 0
Print(r, 0, n-1, "\n r: ");
int j;
for (int i = 0; i < n; ++i) {
   j = id[i]; // vat j
   if (w[j] > rm) { // balo het cho
                    break;
               }
         // w[j] <= m: balo con cho
               rm -= w[j];
         vsum += v[j];
         r[j] = 1;
         cout << "\n Xep vat " << j;
         cout << "\n Tong gia tri " << vsum;
         cout << "\n Trong luong con lai " << rm;
         Go();
}
// cout << "\n break at j = " << j;
Print(r, 0, n-1, "\n r: ");
if (rm > 0 && j < n) { // con cho cho vat j
   r[j] = rm / w[j];
```

```
    vsum += v[j]*r[j];
    rm = 0;
    cout << "\n Xep vat " << j;
        cout << "\n Tong gia tri " << vsum;
        cout << "\n Trong luong con lai " << rm;
        Go();
 }
}

main() {
   Balo();
 // -------------------------------
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}
```

Result

```
n = 3 m = 20
w:  18 15 10
v:  25 24 15
x:  1.38889 1.6 1.5
id:  0 1 2
dec sorted by id:  1 2 0
x:  1.6 1.5 1.38889
r:  0 0 0
Xep vat 1
Tong gia tri 24
Trong luong con lai 5 ?

r:  0 1 0
Xep vat 2
Tong gia tri 31.5
Trong luong con lai 0 ?
```

Problem 4. Lập lịch theo thời hạn

Có n công việc cần thực hiện trên một máy tính, mỗi việc đòi hỏi đúng 1 giờ máy. Với mỗi việc ta biết thời hạn phải nộp kết quả thực hiện sau khi hoàn thành việc đó và tiền thưởng thu được nếu nộp kết quả trước hoặc đúng thời điểm quy định. Chỉ có một máy tính trong tay, hãy lập lịch thực hiện một số công việc trên máy tính sao cho tổng số tiền thưởng thu được là lớn nhất và thời gian hoạt động của máy là nhỏ nhất. Giả thiết rằng máy được khởi động vào đầu ca, thời điểm t = 0 và chỉ tắt máy sau khi đã hoàn thành các công việc đã chọn.

Algorithm Greedy

input

	0	1	2	3
d	1	3	5	1
v	15	10	100	27

vài phương án

PA	lịch	Giờ thứ					thưởng
		1	2	3	4	5	
1	1→2			1		2	10+100=110
2	0→2	0				2	15+100=115
3	0→1→2	0		1		2	15+10+100=125
4	3→1→2	3		1		2	27+10+100=137
5	3→2	3				2	27+100=127

Ta ưu tiên cho những việc có tiền thưởng cao, do đó ta sắp các việc giảm dần theo tiền thưởng. Với mỗi việc k ta đã biết thời hạn giao nộp việc đó là d[k]. Ta xét trục thời gian s. Nếu hạn nộp d[k] trên trục đó đã bận do việc khác thì ta tìm từ thời điểm d[k] trở về trước một thời điểm có thể thực hiện được việc k đó. Nếu tìm được một thời điểm j như vậy, ta đánh dấu bằng mã số của việc đó trên trục thời gian s, s[j] = k. Sau khi xếp việc xong, có thể trên trục thời gian còn những thời điểm rỗi, ta dồn các việc đã xếp về phía trước nhằm thu được một lịch làm việc trù mật, tức là không có giờ trống. Với thí dụ đã cho, N = 4, thời hạn giao nộp d = (1, 3, 5, 1) và tiền thưởng v = (15, 10, 100, 27) ta tính toán như sau:

* Khởi trị: trục thời gian với 5 thời điểm ứng với dmax = 5 là thời điểm muộn nhất phải nộp kết quả, dmax = max { thời hạn giao nộp }, s[1:5] = (0, 0, 0, 0, 0).
* Chọn việc k = 2 có tiền thưởng lớn nhất là v[2] = 100. Xếp việc 2 với thời hạn nộp d[2] = 5 vào lịch s: s[5] = 2. Ta thu được lịch s = (0, 0, 0, 0, 2).
* Chọn tiếp việc 3 có tiền thưởng 27. Xếp việc 3 với thời hạn d[3] = 1 vào s: s[1] = 3. Ta thu được s = (3, 0, 0, 0, 2).
* Chọn tiếp việc 0 có tiền thưởng 15 và thời hạn nọp d[1] = 1. Ta thấy không thể xếp được việc 0 vì từ thời điểm 1 trở về trước trục thời gian đã kín. Ta thu được s không đổi, s = (3, 0, 0, 0, 2).
* Chọn nốt việc 1 có tiền thưởng 10. Xếp việc 1 với thời hạn d[1] = 3 vào s: s[3] = 1, ta thu được s = (3, 0,1, 0, 2).
* Dồn việc trên trục thời gian, ta thu được s = (3, 1, 2, 0, 0).
* Ca làm việc kéo dài đúng N = 4 giờ. Tổng tiền thưởng là

v[3] + v[1] + v[2] = 27 + 10 + 100 = 137.

Program

```
#include <iostream>
#include <fstream>
//#include <bitset>
//#include <cmath>
#include <windows.h>
//#include <cstring>
#include <bits/stdc++.h> // sort, bitset
//#include <time.h>
//#include <algorithm>

using namespace std;

#define Open() ifstream f("JOBS.INP")
#define Close() f.close()

int n;
int *d; // deadline
int *v; // profit
int *id; // index
int *job;
int dmax;
int *s; // lich
int total;

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

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}
```

```
void Print(int x[], int id[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[id[i]];
   }
}

bool Great(int i, int j) {
   return v[i] > v[j];
}

int GetJobs() {
  job = new int[n];
  s = new int[dmax+1];
  for (int i = 0; i <= dmax; ++i) s[i] = -1;
  Print(s, 0, dmax, "\n s:");
  int k;
  for (int i = 0; i < n; ++i) {
     k = id[i]; // viec k co profit tot
     cout << "\n viec k =  " << k << " han nop: " << d[k];
     for (int j = d[k]; j > 0; --j) {
        if (s[j] < 0) {
              s[j] = k;
              Print(s, 1, dmax, "\n Cac viec da chon s: "); // Go();
              break;

     }
         }
  }
  // Don viec va tinh tien thuong
  total = k = 0;
  for (int i = 1; i <= dmax; ++i) {
     if (s[i] != -1) {
        s[++k] = s[i];
        total += v[s[i]];
     }
  }
```

```
  return k;
}

int Jobs() {
  Open();
  f >> n;
  cout << "\n n = " << n;
  d = new int[n]; // deadline
 v = new int[n];  // gia tri
 id = new int[n];
 dmax = 0;
 for (int i = 0; i < n; ++i) {
     f >> d[i];
     if (dmax < d[i]) dmax = d[i];
     id[i] = i;
  }
  for (int i = 0; i < n; ++i) {
     f >> v[i]; // gia tri
  }
  Close();
  Print(d, 0, n-1, "\n deadline d: ");
  Print(v, 0, n-1, "\n profit v: ");
  Print(id, 0, n-1, "\n id: ");
  sort(id, id+n, Great);
  Print(id, 0, n-1, "\n sorted id: ");
  int k = GetJobs(); // 4 2 3 1 Thuong = 137`
  Print(s, 1, k, "\n s: ");
  cout << "\n Tong tien thuong: " << total;
}

main() {
   Jobs();
 // --------------------------------
    cout << "\n T h e   E n d \n";
    return 0;
}
```

Result

```
n = 4
deadline d:  1 3 5 1
profit v:  15 10 100 27
id:  0 1 2 3
sorted id:  2 3 0 1
s: -1 -1 -1 -1 -1 -1
viec k =  2 han nop: 5
Cac viec da chon s:  -1 -1 -1 -1 2
viec k =  3 han nop: 1
Cac viec da chon s:  3 -1 -1 -1 2
viec k =  0 han nop: 1
viec k =  1 han nop: 3
Cac viec da chon s:  3 -1 1 -1 2
s:  3 1 2
Tong tien thuong: 137
T h e   E n d
```

Độ phức tạp tính toán

O(nlog(n)) cho sắp xếp.

Problem 5. Merge Sort

Trộn hai dãy sắp tăng để thu được dãy sắp tăng.

Algorithm

So tay lấy vật nhẹ, tay rỗi cầm vật tiếp.

Program

```
#include <iostream>
#include <fstream>
//#include <bitset>
//#include <cmath>
#include <windows.h>
//#include <cstring>
#include <bits/stdc++.h> // sort, bitset
//#include <time.h>
//#include <algorithm>

using namespace std;

const int MN = 100;
int a[MN] = {2, 7, 8, 8, 10, 15, 20};
int b[MN] = {4, 6, 9, 12, 15, 18, 20, 30};
int c[MN];

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

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}

int Merge(int a[], int n, int b[], int m, int c[]) {
```

```
   int i, j, k;
   i = j = k = 0;
  while(i < n && j < n) {
     c[k++] = (a[i] <= b[j]) ? a[i++] : b[j++];
  }
   while(i < n ) {
     c[k++] = a[i++];
  }
  while(j < n) {
     c[k++] = b[j++];
  }
   return n+m;
}

main() {
  int n = 7, m = 8;
   int k = Merge(a, n, b, m, c);
   Print(a, 0, n-1, "\n a: ");
   Print(b, 0, m-1, "\n b: ");
   Print(c, 0, k-1, "\n c: ");
 // -------------------------------
    cout << "\n T h e   E n d \n";
    return 0;
}
```

Result

```
a:  2 7 8 8 10 15 20
b:  4 6 9 12 15 18 20 30
c:  2 4 6 7 8 8 9 10 12 15 15 18 20 20 0
T h e   E n d
```

Độ phức tạp tính toán

Thuật toán duyệt mỗi mảng một lần: O(m+n).

Note

Số phép gán trị cho c = n + m = #a + #b = #c.

Problem 6. n-Array Merge

Cần trộn n mảng mã số 1..n với số phần tử $s_1, s_2, \ldots, s_n$ được sắp tăng để thu đươc mảng C sắp tăng với số phép gán C ít nhất.

	Array 1	Array 2	Array 3	Array 4
size	5	1	2	6

Phương án 1.

- A1 (+) A2 → C: 5 + 1 = 6
- C (+) A3 → C: 6 + 3 = 9
- C (+) A4 → C: 9 + 6 = 15

  Sum = 9 + 6 + 15 = 30

Phương án 2.

- A1 (+) A3 → C: 5 + 2 = 7
- C (+) A4 → C: 7 + 6 = 13
- C (+) A2 → C: 13 +1 = 14

  Sum = 7 + 13 + 14 = 34

Phương án 3 (tối ưu).

- A2 (+) A3 → C: 1 + 2 = 3
- C (+) A1 → C: 3 + 5 = 8
- C (+) A4 → C: 8 +6 = 14

  Sum = 3 + 8 + 14 = 25

Algorithm (Huffman)

Khởi tạo dãy s chứa các size gọi là mảng trọng số s = (5, 1, 2, 6)

```
Lặp n-1 lần
   Chọn 2 phần tử s[i] và s[j] nhỏ nhất trong số các phần tử còn lại
   Đánh dấu i và j
   Thêm phần tử mới có trọng số s[i]+s[j]vào s
```

	size	step	
1	5	2	Step 1. Chọn min i = 2, j = 3 Thêm phần tử 5, trọng số 1+2 = 3 2 (+) 3 → 5 Step 2. Chọn min i = 1, j = 5 Thêm phần tử 6, trọng số 5+3 = 8 1 (+) 5 → 6 Step 3. Chọn min i = 4, j = 6 Thêm phần tử 7, trọng số 6+8 = 14 4 (+) 6 → 7 Kết quả 2(+)3(+)(+)4,
2	1	1	
3	2	1	
4	6	3	
5	3	2	
6	8	3	
7	14		

Program

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

using namespace std;

const int MN = 100;
int size[MN] = {5, 1, 2, 6};
int n = 4;

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

void Print(int x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}

void Print(bool x[], int d, int c, const char * msg = "") {
   cout << msg;
   for (int i = d; i <= c; ++i) {
      cout << " " << x[i];
   }
}

int Min(int s[], bool mark[], int k) {
   int val = 99999999;
   int j;
   for (int i = 0; i <= k; ++i) {
      if (!mark[i]) {
                  if (s[i] < val) {
                     j = i;
                          val = s[j];
```

```
                    }
                }
    }
    return j;
}

int Huffman(int n) {
    int m = n + n - 1;
    int s[m];
    memcpy(s, size, m*sizeof(int)); // copy size -> s
    bool mark[m];
    memset(mark, false, sizeof(mark));
    int i, j, c = 0; // so phep gan
    for (int k = n-1; k < m-1; ++k) {
        i = Min(s,mark,k);
        mark[i] = true;
        j = Min(s,mark,k);
        mark[j] = true;
        s[k+1] = s[i]+s[j];
        c += s[k+1];
        cout << "\n " << i << " (+) " << j << " -> " << k+1
                    << " Chi phi " << c;
        Go();
   }
   cout << "\n Total: " << c;
   return c;
}

main() {
  Huffman(n);
 // -------------------------------
    cout << "\n T h e   E n d \n";
    return 0;
}
```

Result

```
 1 (+) 2 -> 4 Chi phi 3 ?
```

```
4 (+) 0 -> 5 Chi phi 11 ?

3 (+) 5 -> 6 Chi phi 25 ?

Total: 25
T h e   E n d
```

Độ phức tạp tính toán

Thuật toán duyệt n-1 lần, mỗi lần tìm min cần n phép so sánh, vậy độ phức tạp là $O(n^2)$.

Problem 7. Cây khung cực tiểu

Cho đơn đồ thị vô hướng, liên thông n đỉnh mã số từ 0 đến n – 1, m cạnh mã số từ 0 đến m – 1. Đồ thị được gọi là đơn nếu giữa hai đỉnh có tối đa một cạnh nối hai đỉnh đó. Đồ thị được gọi là liên thông nếu ta cầm một đỉnh bất kỳ rồi nhấc đinht đó lên thì toàn bộ đồ thị được nhấc theo.

Cây khung của đồ thị n đỉnh chứa đúng n đỉnh và một số ít nhất các cạnh của đồ thị sao cho vẫn đảm bảo tính liên thông.

Nhận xét 1. Nếu đơn đồ thị liên thông có n đỉnh thì cây khung có đúng n–1 cạnh.

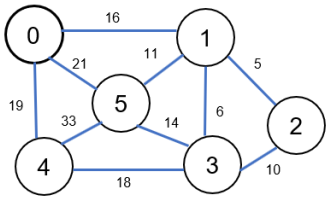
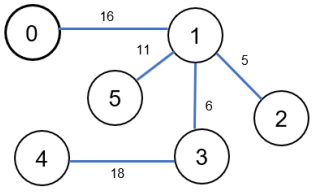
Thuật toán xây dựng cây khung (Kruskal)

```
Algorithm Spanning

Input: Graph G
Output: Spanning Tree S
begin

Khởi trị ây khung S rỗng.
Lấy từng cạnh (x,y) ghép vào S
sao cho không sinh ra chu trình.
return S
end Spanning
```

Vận dụng kỹ thuật Find – Union quản lý các tập đỉnh liên thông.

Độ phức tạp tính toán: Duyệt m cạnh, mỗi cạnh kiểm tra n đỉnh: O(mn).

Cây khung cực tiểu: Dùng lại thuật toán Kruskal duyệt các cạnh theo độ dài tăng dần.

GRAPH.INP	GRAPH	MIN SPANNING TREE
6 10 0 1 16 0 4 19 0 5 21 1 2 5 1 3 6 1 5 11 2 3 10 3 4 18 3 5 14 4 5 33		

Program

```
/************************************
 Name: SPANNINGTREE.CPP
 Copyright: (C) 2021
 Author: Devc Fan
 Date: 11/10/21 15:20
 Description:
 Minimal Spanning tree
************************************/
#include <iostream>
#include <fstream>
#include <algorithm>

using namespace std;

const int NONE = 0;
const int SPAN = 1;

typedef struct Edge {
           int X;
           int Y;
           int Len;
           int Type;
         } Edge;
Edge *c;
int n, m;
int *d;

void Print(Edge e[], int d, int c, const char * msg = "") {
  cout << msg;
  for (int i = d; i <= c; ++i) {
     cout << "\n " << e[i].X << " - " << e[i].Y << " len = " << e[i].Len
              << " Type: " << e[i].Type;
  }
}

void Swap(int & x, int & y) {
  int t = x;
```

```
  x = y; y = t;
}

void Read() {
   ifstream f("GRAPH.INP");
   f >> n >> m;
   cout << "\n " << n << " vertexes and " << m << " edges";
   c = new Edge[m];
   int x, y, d, t;
   for (int i = 0; i < m; ++i) {
      f >> x >> y >> d;
      if (x > y) Swap(x,y);
      c[i].X = x; c[i].Y = y;
      c[i].Len = d;
      c[i].Type = NONE;
   }
   f.close();
}

bool Less(Edge a, Edge b) {
   return a.Len < b.Len;
}

int Find(int x) {
   while (d[x] != x) x = d[x];
   return x;
}

int Union(int x, int y) {
   x = Find(x); y = Find(y);
   if (x == y) return 0;
   if (x < y) d[y] = x;
   else d[x] = y;
   return 1;
}

void Spanning() {
  Read();
```

```
 Print(c, 0, m-1, "\n Read:\n");
 sort(c, c+m, Less); // Sap tang theo len
 Print(c, 0, m-1, "\n Inc sorted by Len:\n");
 d = new int[n];
 for (int i = 0; i < n; ++i) d[i] = i;
 for (int i = 0; i < m; ++i) {
    if (Union(c[i].X, c[i].Y))
            c[i].Type = SPAN;
 }
 Print(c, 0, m-1, "\n Result:\n");
 int total = 0;
 for (int i = 0; i < m; ++i) {
   if (c[i].Type == SPAN) {
           cout << "\n " << c[i].X << " - " << c[i].Y;
                total += c[i].Len;
        }
 }
 cout << "\n Total: " << total;
}

main() {
  Spanning();
 // -------------------------------
    cout << endl << "\n\n T h e   E n d \n";
    return 0;
}
```

Result

```
6 vertexes and 10 edges
Read:

0 - 1 len = 16 Type: 0
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
1 - 2 len = 5 Type: 0
1 - 3 len = 6 Type: 0
1 - 5 len = 11 Type: 0
2 - 3 len = 10 Type: 0
```

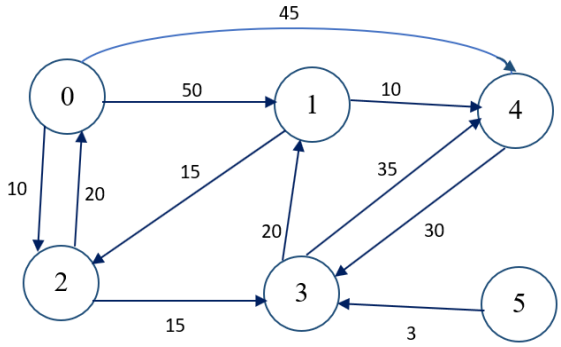
```
3 - 4 len = 18 Type: 0
3 - 5 len = 14 Type: 0
4 - 5 len = 33 Type: 0
Inc sorted by Len:

1 - 2 len = 5 Type: 0
1 - 3 len = 6 Type: 0
2 - 3 len = 10 Type: 0
1 - 5 len = 11 Type: 0
3 - 5 len = 14 Type: 0
0 - 1 len = 16 Type: 0
3 - 4 len = 18 Type: 0
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
4 - 5 len = 33 Type: 0
Result:

1 - 2 len = 5 Type: 1
1 - 3 len = 6 Type: 1
2 - 3 len = 10 Type: 0
1 - 5 len = 11 Type: 1
3 - 5 len = 14 Type: 0
0 - 1 len = 16 Type: 1
3 - 4 len = 18 Type: 1
0 - 4 len = 19 Type: 0
0 - 5 len = 21 Type: 0
4 - 5 len = 33 Type: 0
1 - 2
1 - 3
1 - 5
0 - 1
3 - 4
Total: 56
T h e   E n d
```

Độ phức tạp tính toán

Thuật toán duyệt m cạnh, mỗi cạnh duyệt n đỉnh gọi hàm Union, vậy độ phức tạp là $O(mn)$.

Problem 8. Dijkstra

Cho đồ thị có hướng thể hiện một mạng giao thông với các địa điểm (đỉnh) và các đường một chiều thể hiện bằng các cung có hướng x → y. Trên mỗi cung x → y có nhãn thể hiện chi phí đi từ địa điểm x đến địa điểm y. Cần tìm các đường có chi phí ít nhất từ một đỉnh s cho trước đến các đỉnh còn lại.



Dijkstra (sửa đỉnh)						
	(0)	(1)	(2)	(3)	(4)	(5)
c	0	∞	∞	∞	∞	∞
(0)	*	50	10		45	
(2)			*	25		
(3)		45		*		
(1)		*				
(4)					*	
(5)						*

0 →2: 10

0 →2→3: 10+15 = 25

0→2→3→1: 10+15+20 = 45

0→4: 10+15+10 = 45

0→5: ∞

Kí hiệu C(s, x) là chi phí ít nhất từ đỉnh xuất phát s đến đỉnh y. Ta thấy

C(s, s) = 0: Từ s đến s không mất chi phí nào.

Muốn đi theo chi phí tối thiểu từ s đến y thì ta chọn chi phí ít nhất từ s đến các đỉnh x sát trước đỉnh y cộng thêm một chi phí cho cung đường x → y:

$$C(s, x) = \min \{C(y) + d(x,y), y \rightarrow x\}$$

trong đó y là đỉnh sát trước đỉnh x, d(y, x).

Muốn giải trình đường đi: Đặt con trỏ trước pred[y] = x có nghĩa x → y.

Program

```
/************************************
 Name: DIJKSTRA.CPP
 Copyright: (C) 2021
 Author: Devc Fan
 Date: 11/10/21 15:20
 Description:
```

```
 Single Source Shortest Paths
************************************/
#include <iostream>
#include <fstream>
#include <algorithm>
#include <windows.h>

using namespace std;

int n; // dinh
int m; // canh

int ** d;
int *c;
int s; // dinh xuat phat
int inf;
bool *taken;
int * pred;

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

void Print(int a[], int d, int c,  const char * msg = "") {
  cout << msg;
  for (int i = d; i <= c; ++i) {
     cout << " " << a[i];
  }
}

void Print(int *a[], int d, int c,  const char * msg = "") {
  cout << msg;
  for (int i = d; i <= c; ++i) {
    Print(a[i], d, c, "\n ");
  }
}
```

```
void Read() {
   ifstream f("DIJKSTRA.INP");
   f >> n >> m;
   cout << "\n n = " << n << "   m = " << m;
   d = new int *[n];
   for (int i = 0; i < n; ++i) {
       d[i] = new int[n];
       memset(d[i], 0, n*sizeof(int));
   }
   int x, y, v;
   inf = 0;
  for (int i = 0; i < m; ++i) {
     f >> x >> y >> v;
     cout << "\n " << x << "->" << y << ": " << v;
     d[x][y] = v;
     inf += v;
  }
   f.close();
}

int GetMin() {
  int vmin = inf;
  int imin;
 for (int i = 0; i < n; ++i) {
    if (!taken[i] && c[i] < vmin) {
        vmin = c[i]; imin = i;
    }
 }
 return imin;
}

void Path(int y) {
  if (pred[y] == y) {
    cout << "\n " << y;
   return;
 }
  Path(pred[y]);
```

```
   cout << " -> " << y;
}

void Run(int start) {
  Read();
  Print(d, 0, n-1, "\n Init: ");
  c = new int[n];
  for (int i = 0; i < n; ++i) c[i] = inf;
  taken = new bool[n];
  pred = new int[n];
  for (int i = 0; i < n; ++i) pred[i] = i;
  memset(taken, false, n*sizeof(bool));
  s = start;  c[s] = 0;
  int x, val;
  for (int i = 0; i < n; ++i) {
     x = GetMin(); val = c[x];
     taken[x] = true;
         for (int y = 0; y < n; ++y) {
        if (!taken[y] && d[x][y] > 0 && val + d[x][y] < c[y]) {
            c[y] = c[x] + d[x][y];
            pred[y] = x;
        }
      }
  }
  cout << "\n Result:";
  for (int i = 0; i < n; ++i) {
    Path(i);
    if (c[i] < inf) cout << " : " << c[i];
        else cout << " : no path.";
  }
}

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

Result

```
n = 6   m = 11
0->1: 50
0->2: 10
0->4: 45
1->2: 15
1->4: 10
2->0: 20
2->3: 15
3->1: 20
3->4: 35
4->3: 30
5->3: 3
Init:
 0 50 10 0 45 0
 0 0 15 0 10 0
 20 0 0 15 0 0
 0 20 0 0 35 0
 0 0 0 30 0 0
 0 0 0 3 0 0
Result:
0 : 0
0 -> 2 -> 3 -> 1 : 45
0 -> 2 : 10
0 -> 2 -> 3 : 25
0 -> 4 : 45
5 : no path.
T h e   E n d
```

Độ phức tạp tính toán

Thuật toán sửa n đỉnh, mỗi lần gọi hàm min duyệt n đỉnh, vậy độ phức tạp là $O(n^2)$.

Problem 9. Bống

Tấm được Bụt trao cho k chú bống. Nhiệm vụ của Tấm là phải thả k bống này vào 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.

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 ≤ gn ≤ 100.

Dữ liệu ra ghi trong tệp văn bản BONG.OUT: dòng đầu tiên là tổng giá trị quà của phương án thả bống tối ưu. Từ dòng thứ hai là dãy k số hiệu giếng được chọn cho mỗi chú bống.

Ví dụ

BONG.INP	BONG.OUT
4 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 1 3 4 6

Kết quả cho biết 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 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

Đề bài đòi hỏi hai tiêu chí

- Tiêu chí 1. Các chú bống phải được xếp vào các giếng theo trật tự.
- Tiêu chí 2. Tổng phần thưởng → max.

Đầu tiên ta tạm gác lại tiêu chí 2, chỉ thực hiện theo tiêu chí 1. Vậy ta xếp lần lượt bống vào các giếng đầu tiên:

```
bống i vào giếng i
```

giếng	1	2	3	4	5	6	tổng quà	
bống	**1**	**2**	**3**	**4**				init
quà	1	1	6	1			9	
bống	**1**	**2**	**3**			**6**		dịch bống 4
quà	1	1	6			9	17	
bống	**1**	**2**		**4**		**6**		dịch bống 3
quà	1	1		10		9	21	
bống	**1**		**3**	**4**		**6**		dịch bống 2
quà	1		4	10		9	24	
bống	**1**		**3**	**4**		**6**		dịch bống 1
quà	1		4	10		9	31	

Giờ ta dịch dần các chú bống qua phải.

Bống số 4 có hai khả năng lựa chọn là giếng 5:3 (với số quà là 3) và giếng 6:9 (với số quà 9). Ta dịch bống 4 từ 4 → 6 để nhận thêm quà.

Bống số 3 có hai khả năng lựa chọn là giếng 4:10 và giếng 5:2. Ta dịch bống 3 từ 3 → 4 để nhận thêm quà.

Ta làm tương tự với bống số 2 và 1 là hoàn tất.

Program

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

using namespace std;

const int MN = 101;
int k; // bong
int n; // gieng
int v[MN][MN]; // qua
int bg[MN]; // bong i trong gieng bg[i]
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();
}

// Tim gieng max cho bong i
// bong i hien dat trong gieng bg[i]
// duyet cac gieng g bg[i]:bg[i+1]
int FindMax(int i) {
  int gd = bg[i];
  if (gd == n) return i;
  int gc = (i == k) ? n+1 : bg[i+1];
  for (int g = gd+1; g < gc; ++g) {
     if (v[i][gd] < v[i][g])
```

```
        gd = g;
  }
  return gd;
}

void Run() {
  Read();
  cout << "\n k = " << k << "  n = " << n;
  Print();
  // Init
  qua = 0;
  for (int i = 1; i <= k; ++i) {
    bg[i] = i;
    qua += v[i][i];
  }
  cout << "\n Init: qua = " << qua;
  qua = 0;
  if (k < n) {
    for (int i = k; i >= 1; --i) {
      bg[i] = FindMax(i); // dich bong qua phai
      qua += v[i][bg[i]];
    }
  }
  cout << "\n Result: Tong so qua: " << qua;
  ofstream g("BONG.OUT"); // Open file to write
  g << qua << endl;
  for (int i = 1; i <= k; ++i) {
    cout << "\n Tha ca bong " << i << " vao gieng " << bg[i];
    g << " " << bg[i];
  }
  g.close();
}

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
Init: qua = 9
Result: Tong so qua: 24
Tha ca bong 1 vao gieng 1
Tha ca bong 2 vao gieng 3
Tha ca bong 3 vao gieng 4
Tha ca bong 4 vao gieng 6
T h e   E n d
```

Độ phức tạp tính toán

Thuật toán thả k chú cá, mỗi chú cá cần duyệt tối đa n giếng, vậy độ phức tạp là $O(kn)$.

Comment

Tham lam đôi khi không cho dẫn đến lời giải tối ưu

Ví dụ

```
4 6
1 1 6 4 3 9
9 1 4 7 2 7
7 2 6 7 9 3
6 9 7 1 8 7
```

giếng	1	2	3	4	5	6	tổng quà	
bống	**1**	**2**	**3**	**4**				init
quà	1	1	6	1			9	
bống	**1**	**2**	**3**		**5**			dịch bống 4: 4:(4)→(5)
quà	1	1	6		8		16	
bống	**1**	**2**		**4**	**5**			dịch bống 3: 3:(3)→(4)
quà	1	1		7	8		17	
bống	**1**		**3**	**4**	**5**			dịch bống 2: 2:(2)→(3)
quà	1		4	7	8		20	
bống	**1**		**3**	**4**	**5**			dịch bống 1:
quà	1		4	7	8		20	

Trong khi phương án tối ưu là như sau:

```
4 6
1 1 6 4 3 9
9 1 4 7 2 7
7 2 6 7 9 3
6 9 7 1 8 7
```

giếng	1	2	3	4	5	6	tổng quà
bống			**3**	**4**	**5**	**6**	
quà			6	7	9	7	29

Beauty

Cho một hoán vị của N số 0,..., N–1. Một đoạn gồm k số liên tiếp nhau được gọi là đẹp nếu trong đoạn đó có đủ k số từ 0 đến k–1.

Hãy cho biết có bao nhiêu đoạn đẹp trong hoán vị.

Input: file text BEAUTY.INP

Dòng đầu tiên: số N.

Dòng thứ hai: Một hoán vị của N số 0, …, N–1, các số được ghi cách nhau qua dấu cách.

Giới hạn $1 \le N \le 2000000$

Output: Hiển thị trên màn hình số lượng đoạn đẹp.

Ví dụ

BEAUTY.INP
10 6 9 5 7 4 0 3 2 1 8

Hoán vị này có 4 đoạn đẹp là:

(0)

(0,3,2,1)

(4,0,3,2,1)

(6,9,5,7,4,0,3,2,1,8)

Vị trí	0	1	2	3	4	5	6	7	8	9
Hoán vị	6	9	5	7	4	0	3	2	1	8
Đoạn đẹp 1 [5;5]						B				
Đoạn đẹp 2 [5;8]						B	B	B	B	
Đoạn đẹp 3 [4;8]					B	B	B	B	B	
Đoạn đẹp 4 [0;9]	B	B	B	B	B	B	B	B	B	B

Algorithm

Dễ thấy mọi hoán vị đều có đoạn đẹp là đoạn chứa duy nhất một số 0. Nếu hoán vị đó có trên một phần tử thì có thêm đoạn đẹp thứ hai chứa toàn bộ hoán vị.

Ta ký hiệu mỗi đoạn dưới dạng *cửa sổ* [*L*;*R*] trong đó *L* và *R* là hai *chỉ số đầu* (*cánh trái*) và *cuối đoạn* (*cánh phải*) của cửa sổ. Theo ví dụ trên, ta có hoán vị là một dãy 10 số (6,9,5,7,4,0,3,2,1,8) với các chỉ số vị trí tính từ 0. Dãy này có 4 đoạn đẹp là:

[5;5] = (0)

[5;8] = (0,3,2,1)

[4;8] = (4,0,3,2,1)

[0;9] = (6,9,5,7,4,0,3,2,1,8).

Ta xuất phát từ cửa sổ là đoạn $D = [L;R]$ chứa duy nhất số 0: L = R = 5; $D[5;5] = (0)$. Đây là đoạn đẹp. Gọi m là số nhỏ nhất chưa xuất hiện trong D. m được xác định như sau:

- Nếu D là đoạn đẹp thì D chứa đầy đủ các số từ 0 đến $R–L$ ta chọn $m = R–L + 1$.
- Nếu D là đoạn không đẹp thì ta chọn m là số nhỏ nhất chưa xuất hiện trong D.

Sau khi xác định được m, ta mở rộng đoạn D để kết nạp thêm số m như sau:

- Nếu m nằm ở bên trái L, pos[m] < L thì ta nới rộng cánh trái của D:

  L = pos[m], trong đó pos[m] là vị trí của số m trong hoán vị.
- Nếu m nằm ở bên phải R, R < pos[m] thì ta nới rộng cánh phải:

  R = pos[m].

Tóm lại, L và R sẽ được cập nhật như sau:

```
L = min(L, pos[m]);
       R = max(R, pos[m]);
```

Sau khi cập nhật L và R ta cần tính lại chiều dài của đoạn mới: `len = R–L+1`.

Ta minh họa thuật toán với ví dụ hiện có.

Ví dụ

Vị trí	0	1	2	3	4	5	6	7	8	9
Hoán vị	6	9	5	7	4	0	3	2	1	8

Xuất phát từ đoạn D (cửa sổ) chứa số 0: $D = [5;5] = (0)$, $L = R = 5$, $R–L = 5 – 5 = 0$: Đoạn này đẹp (đoạn đẹp thứ nhất). Cần mở rộng để thêm số nhỏ nhất nằm ngoài D là $m = R–L+1 = 0+1 = 1$: Vì pos[1] = 8 > R = 5 nên ta phải nới cánh phải của cửa số: $[5;5] \rightarrow [5;8] = (0,3,2,1)$. Đoạn này hiện chứa 0 và 1. Ta cần tìm số m nhỏ nhất chưa xuất hiện trong đoạn này kể từ m = 2.

Kiểm tra đoạn [5;8] = (0,3,2,1) với $m \geq 2$: Đoạn này hiện chứa số 0 và 1, nay chứa thêm các số m = 2, 3 do đó là đoạn đẹp (đoạn đẹp thứ hai).

Tiếp theo, ta cần mở rộng để thêm số m = 4. Vì pos [4] = 4 < L = 5 nên ta phải nới cánh trái của cửa sổ để kết nạp số 4: $[5;8] \rightarrow [4;8] = (4,0,3,2,1)$. Đoạn này hiện chứa các số từ 0 đến 4. Ta cần tìm số nhỏ nhất chưa xuất hiện trong đoạn này kể từ m = 5.

Kiểm tra đoạn [4;8] = (4,0,3,2,1) với $m \geq 5$: Đẹp (đoạn đẹp thứ ba).

Tiếp theo, ta cần mở rộng để thêm số $m = 5$. Vì pos[5] = 2 < L = 4 nên ta phải nới cánh trái để kết nạp số 5: [4;8] → [2;8] = (5,7,4,0,3,2,1). Đoạn này hiện chứa các số từ 0 đến 5. Ta cần tìm số nhỏ nhất chưa xuất hiện trong đoạn này kể từ m = 6.

Kiểm tra đoạn [2;8] = (5,7,4,0,3,2,1) với $m \geq 6$: Không đẹp vì thiếu 6. Vậy ta cần mở rộng để thêm số m = 6. Vì pos[6] = 0 < L = 2 nên ta phải nới cánh trái để kết nạp số 6: [2;8] → [0;8] = (6,9,5,7,4,0,3,2,1). Đoạn này hiện chứa các số từ 0 đến 6. Ta cần tìm số nhỏ nhất chưa xuất hiện trong đoạn này kể từ m = 7.

Kiểm tra đoạn [0;8] = (6,9,5,7,4,0,3,2,1) với $m \geq 7$: Không đẹp vì thiếu 8. Cần mở rộng để thêm số m = 8. Vì pos[8] = 9 > R = 8 nên ta cần nới cánh phải để kết nạp số 8: [0;8] → [0;9] = (6,9,5,7,4,0,3,2,1,8). Đoạn này chứa kín dãy số nên là đoạn đẹp (đoạn đẹp thứ tư) và thuật toán dừng.

Thuật toán này lặp tối đa N lần, mỗi lần cửa sổ được nới rộng để chứa thêm ít nhất là một số.

Để kiểm tra đoạn D = [L;R] có đẹp hay không ta lưu ý rằng sau khi được mở rộng thì D chứa mọi số từ 0 đến m, do đó ta chỉ cần xét các số từ m+1 đến N–1 có trong đoạn hay không.

```
Algorithm Beauty
Input: Vị trí của các số trong hoán vị pos[i], 0 ≤ i ≤ N-1
Output:  bt số đoạn đẹp
begin
   // Khởi trị
   // window D = [L;R] chứa số 0
   L ← pos[0];  R ← pos[0];
   bt ← 0; // đếm số đoạn đẹp D
   len ← 1; // #D
   // N là chiều dài của hoán vị
   for m ← 1 to N–1 do
      if (pos[m] < L) or (R < pos[m]) then
         // m nằm ngoài D[L;R]
          if (m = len) then
              bt ← bt + 1; // thêm một đoạn đẹp
          end if
         // m đang nằm ngoài D
    // Mở rộng window D[L;R] để thêm số m
          // Cập nhật A, C và tính len of D
        L ← min(L, pos[m]);
        R ← max(R, pos[m]);
         len ← R-L+1;
      end if m nằm ngòai  D
    end for
    bt ← bt + 1; // Toàn hình là Hình đẹp cuối cùng
  return bt;
end Beauty
```

Độ phức tạp tính toán

O(NlogN).

Comment

- Các hàm logarithm trong sách này đều tính theo cơ số 2.
- Khi đọc số m_i từ input file ta cần ghi vào mảng pos[*i*] = *m* với ý nghĩa số *m* xuất hiện tại vị trí *i* trong dãy, $0 \le i < N$.

Program

```
// BEAUTY.CPP
#include <iostream>
#include <fstream>
using namespace std;

const int MAXN = 2000000;
int n; // len day so
int pos[MAXN+1]; // pos[m] = i: so m nam tai vi tri i
int L, R; // cua trai va phai cua window

const char * fn = "BEAUTY.INP";

// doc du lieu
void read() {
  ifstream f(fn);
 f >> n; // so phan tu
 int m;
 for (int i = 0; i < n; ++i) {
       f >> m;
       pos[m] = i; // so m dat tai vi tri pos[m]
 }
 f.close();
} // read

int beauty() {
 L = R = pos[0]; // Dat window D[L;R] chua doan dep dau tien (0)
  int bt = 0; // dem so doan dep
  int m; // so 1 nam ngoai doan dep dau tien (0)
  int len = 1; // chieu dai window
  for (m = 1; m < n; ++m) {
              if (pos[m] < L || pos[m] > R) {
                    // m nam ngoai D[L;R]
```

```
                    if (m == len)
                           ++bt; // them 1 doan dep
                    // mo rong D[L[R] de chua them m
              L = min(L,pos[m]);
              R = max(R,pos[m]);
              // chinh len
              len = R-L+1;
       } // if m nam ngoai doan Di
  } // for
  // Toan hoan vi la doan dep
  ++bt;
 return bt;
 } // beauty

main() {
 read();
  cout << "\n result = " << beauty();
  return 0;
  cout << "\n T h e   E n d ";
} // main
```

Seats

IOI 2018, Japan

Bạn đang tổ chức kỳ thi lập trình quốc tế trong một hội trường có dạng một hình chữ nhật gồm HW ghế ngồi, được bố trí thành H hàng và W cột. Các hàng được đánh số từ 0 đến H–1 và các cột được đánh số từ 0 đến W–1. Ghế ở hàng r và cột c được ký hiệu là (r,c). Bạn mời HW thí sinh được đánh số từ 0 đến H×W–1 vào phòng thi. Bạn tạo sẵn một sơ đồ chỗ ngồi, xếp cho mỗi thí sinh thứ i ($0 \le i \le HW-1$) vào ghế (R_i,C_i). Trong sơ đồ chỗ ngồi, mỗi thí sinh ngồi ở đúng một ghế và mỗi ghế có đúng một thí sinh.

Một tập hợp S gồm các ghế trong hội trường được gọi là có dạng hình chữ nhật nếu tồn tại các số nguyên r_1, r_2, c_1, c_2 thỏa mãn các điều kiện sau:

$$0 \le r_1 \le r_2 \le H-1.$$

$$0 \le c_1 \le c_2 \le W-1.$$

Tập S chính là tập hợp tất cả các ghế (r,c) thỏa mãn $r_1 \le r \le r_2$ và $c_1 \le c \le c_2$.

Một tập hợp có dạng hình chữ nhật gồm k ghế được gọi là đẹp nếu các thí sinh được sắp xếp ngồi trên các ghế thuộc tập này có số thứ tự từ 0 đến k–1. Độ đẹp của một sơ đồ chỗ ngồi là số lượng các tập hợp có dạng hình chữ nhật đẹp trong sơ đồ.

Sau khi tạo ra sơ đồ chỗ ngồi, bạn nhận được một số yêu cầu hoán chuyển ghế ngồi cho một cặp 2 thí sinh. Có Q yêu cầu, đánh số từ 0 đến Q–1 theo thứ tự thời điểm xuất hiện. Yêu cầu thứ j ($0 \le j \le Q-1$) thực hiện việc hoán chuyển chỗ ngồi 2 thí sinh A_j và B_j. Yêu cầu này sẽ được thực hiện ngay và sau đó sơ đồ chỗ ngồi được cập nhật. Sau mỗi lần cập nhật, bạn được yêu cầu tính độ đẹp của sơ đồ chỗ ngồi hiện tại.

Dữ liệu vào được ghi trong file text SEATS.INP

SEATS.INP	Giải thích
H W	Số lượng hàng H, cột W
R_0 ... R_{HW-1}	dãy HW số cho mảng R | (R[i],C[i]) cho biết thí sinh mã số i ngồi
C_0 ... C_{HW-1}	dãy HW số cho mảng C | tại hàng R[i], cột C[i], $0 \le i \le HW-1$.
Q	Số test, mỗi test một dòng gồm 2 số a b
a b	Cần đổi chỗ hai thí sinh mang mã số a và b
...	

Ví dụ

SEATS.INP

```
2 3
0 1 1 0 0 1
0 0 1 1 2 2
3
0 5
0 5
4 5
```

Giải thích

0	3	4
1	2	5

Phòng thi có H = 2 hàng, mỗi hàng W = 3 ghế
Thí sinh mã số 0 ngồi tại hàng 0, cột 0: 0(0,0).
Thí sinh 1 (1,0), 2(1,1), 3(0,1), 4(0,2), 5(1,2).

3 test
Test 1: đổi chỗ 2 thí sinh 0 và 5.
Test 2: đổi chỗ 2 thí sinh 0 và 5.
Test 3: đổi chỗ 2 thí sinh 4 và 5.

Ví dụ trên cho ta các kết quả sau đây:

Cấu hình ban đầu:

0	3	4
1	2	5

Test 1: Đổi chỗ 0 và 5. Ba hình đẹp

5	3	4
1	2	0

1

0

2

1	2	0

3

5	3	4
1	2	0

Test 2: Đổi chỗ 0 và 5. Bốn hình đẹp

0	3	4
1	2	5

1

0

2

0
1

3

0	3
1	2

4

0	3	4
1	2	5

Test 3. Đổi chỗ 4 và 5: Bốn hình đẹp

0	3	5
1	2	4

1

0

2

0
1

3

0	3
1	2

4

0	3	5
1	2	4

Algorithm

Bài này có thuật giải giống bài ***đoạn đẹp*** được mở rộng từ không gian một chiều (đoạn thẳng) sang không gian hai chiều (hình chữ nhật).

Sơ đồ tổng quát cho thuật giải là như sau:

1. Xuất phát từ hình *D* ban đầu (gọi là cửa sổ) chỉ chứa ô số (0):
   - Tính diện tích (số ô) của *D*.
2. Lặp đến khi diện tích của *D* = diện tích phòng thi (HW):
   - Tìm số *m* đầu tiên (nhỏ nhất) nằm ngoài hình *D*.

- Nếu *m* = diện tích của *D* thì *D* là hình đẹp,
- Nếu *m* < diện tích của *D* thì nới rộng *D* để kết nạp *m*.

Ta quy ước biểu diễn mỗi hình chữ nhật (HCN) *ABCD* trong sơ đồ phòng thi thông qua tọa độ (chỉ số) của hai ô đối diện *A* là ô trái trên và *C* là ô phải dưới. Chỉ số mỗi ô được xác định theo (dòng, cột).

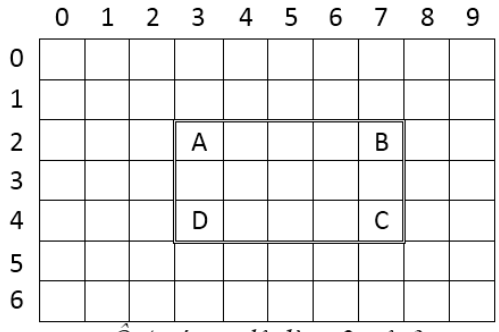
Giả sử ô *A* nằm trên dòng *ar*, cột *ac*; ô *C* nằm trên dòng *cr*, cột *cc*. Theo quy ước về chỉ số các mảng hai chiều ta có:

$ar \le cr$ và $ac \le cc$

tức là hai tọa độ của ô *A* không lớn hơn hai tọa độ tương ứng của ô *C*.

Trong hình dưới ta có *ar* = 2, *ac* = 3, *cr* = = 4, *cc* = 7.

Ký hiệu (*x*) là ô chứa số *x*, ta xuất phát từ HCN *D* chứa duy nhất số (0). Mỗi lần ta mở rộng hình *D* để chứa thêm số nhỏ nhất nằm ngoài *D.* Gọi tọa độ của *D* là *A*(*ar, ac*) và *C*(*cr, cc*). Diện tích của *D* là *s* = (*cr*–*ar*+1)*(*cc*–*ac*+1). Ta thấy *D* là hình đẹp khi và chỉ khi nó chứa đầy đủ các số từ 0 đến *s* – 1. Đặc biệt, khi *s* = H*W ta nhận được toàn bộ phòng thi. Đây cũng là hình đẹp. Thuật toán sẽ kết thúc.



Ô A có tọa độ dòng 2, cột 3
Ô C có tọa độ dòng 4, cột 7

Gọi *m* là số đầu tiên trong khoảng 0 đến *s*–1 chưa xuất hiện trong *D*, $0 \le m \le s-1$. Ta xét hai trường hợp sau:

- *m* = *s*: *D* là hình đẹp. Ghi nhận thêm một hình đẹp.
- *m* < *s*: *D* là hình không đẹp. Ta nới rộng *D* để chứa thêm số *m.*
- Biết *m*, tại bước tiếp theo ta nới rộng HCN *D* bằng cách nạp thêm số *m* và xác định bao lồi là HCN chứa *D* và chứa thêm ô (*m*). Việc này khá dễ: ta chỉ việc cập nhật các tọa độ của các ô *A* và *C* sao cho *D* chứa thêm (*m*), cụ thể là:
- Chỉ số dòng *ar* = min(*ar*, R[*m*])
- Chỉ số cột *ac* = min(*ac*, C[*m*])
- Chỉ số dòng *cr* = max(*cr*, R[*m*])
- Chỉ số cột *cc* = max(*cc*, C[*m*])

trong đó ô (*m*) nằm trên dòng R[*m*], cột C[*m*].

Cập nhật *D* xong ta cần tính lại diện tích của *D*: *s* = (*cr*–*ar*+1)*(*cc*–*ac*+1).

Sau khi được cập nhật, *D* sẽ chứa *m* nên *D* chứa mọi số 0…*m*. Do đó, để xác định tính đẹp của *D* ta chỉ cần xét tiếp các số từ *m*+1 đến diện tích mới của *D*. Ta cũng quy ước là sau khi cập nhật tọa độ của các ô *A* và *C* theo *m*, nếu ta nhận được hình đẹp thì ta gọi *m* là *điểm đẹp*. Vậy *điểm đẹp* là điểm sau khi thêm vào *D* sẽ thu được hình đẹp.

Dễ thấy, mọi sơ đồ phòng thi luôn luôn có hình đẹp đầu tiên chứa duy nhất ghế ngồi của thí sinh số 0. Nếu HW > 1 thì có thêm hình đẹp là toàn bộ phòng thi.

Độ phức tạp tính toán

Thuật toán duyệt mỗi số từ 1 đến HW−1 đúng một lần. Mỗi lần thêm tối thiểu một số nằm ngoài hình *D* sẽ làm tăng diện tích thêm ít nhất là một dòng hoặc một cột, do đó độ phức tạp tính tán là O(HWlog(HW)), trong đó HW = H×W là diện tích của phòng thi.

```
Algorithm Beauty
Input:
        integer HW = H × W ≥ 1.
        R[i], 0 ≤ i < HW: tọa độ dòng của các số 0… HW −1.
        C[i], 0 ≤ i < HW: tọa độ cột của các số 0… HW −1.
Output:  bt số HCN đẹp
begin
   // Khởi trị
   // window D = [A(ar,ac); C(cr,cc)] chứa số 0
    ar ← R[0];
   ac ← C[0];
   cr ← ar;
   cc ← ac;
   bt ← 0; // đếm số hình đẹp D
   s ← 1; // diện tích window D
  // HW là diện tích toàn phòng thi
  for m ← 1 to HW−1 do
      if (m nằm ngoài D) then
          if (m = s) then
              bt ← bt + 1; // thêm một hình đẹp
          end if
        // m đang nằm ngoài hình D
        // Mở rộng window D[A;C] để thêm số m
         s ← updateAC(m); // Cập nhật A, C và tính diện tích của D
      end if m nằm ngoài  D
    end for
    bt ← bt + 1; // Toàn hình là Hình đẹp cuối cùng
  return bt;
end Beauty
```

Ví dụ 1

SEATS.INP
4 5 2 2 1 2 2 1 1 1 0 3 3 0 0 0 1 3 2 0 3 3 1 2 0 3 0 2 1 3 2 0 2 3 4 1 4 3 4 0 4 1 1 3 7

Dữ liệu vào cho biết:

H = 4, W = 5. Tổng cộng có HW = H×W = 20 thí sinh mã số từ 0 đến 19.

Sơ đồ phòng thi lúc đầu:

17	13	8	11	12
2	6	5	7	14
4	0	1	3	16
9	19	10	15	18

1 test: đổi chỗ hai thí sinh 3 và 7.

Thuật toán được thể hiện như sau:

Sơ đồ chỗ ngồi lúc đầu: Thí sinh m ngồi tại dòng R[m], cột C[m]

Thí sinh	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
Dòng R	2	2	1	2	2	1	1	1	0	3	3	0	0	0	1	3	2	0	3	3
Cột C	1	2	0	3	0	2	1	3	2	0	2	3	4	1	4	3	4	0	4	1

Sơ đồ phòng thi lúc đầu:

17	13	8	11	12
2	6	5	7	14
4	0	1	3	16
9	19	10	15	18

Sau khi đổi chỗ hai thí sinh 3 và 7:

1	1	8	1	1
2	6	5	3	1
4	0	**1**	7	1
9	1	1	1	1

Xuất phát từ D = [A(2,1); C(2,1)] = (0).
Hình này *đẹp (thứ nhất)*. Số ngoài hình m = **1**.

1	1	8	1	1
2	6	5	3	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(2,1); C(2,2)] chứa thêm ô (1).
Diện tích s = 2
Hình này *đẹp (thứ 2)*. Số ngoài hình m = **2**.

1	1	8	1	1
2	6	5	**3**	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(1,0); C(2,2)] chứa thêm ô (2).
Hình này không đẹp. Số ngoài hình m = **3**.

1	1	**8**	1	1
2	6	5	3	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(1,0); C(2,3)] chứa thêm ô (3).
Hình này *đẹp (thứ ba)*. Số ngoài hình m = **8**.

1	1	8	1	1
2	6	5	3	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(0,0); C(2,3)] chứa thêm ô (8).
Hình này không đẹp. Số ngoài hình m = **9**.

1	1	8	1	**1**
2	6	5	3	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(0,0); C(3,3)] chứa thêm ô (9).
Hình này không đẹp. Số ngoài hình m = **12**.

1	1	8	1	1
2	6	5	3	1
4	0	1	7	1
9	1	1	1	1

Xét D = [A(0,0); C(3,4)] chứa thêm ô (12).
Diện tích = 20 cực đại. Hình này *đẹp (thứ tư)* .
Thuật toán kết thúc với 4 hình đẹp, sau 7 lần duyệt.

Program

```
// SEATS.CPP
#include <iostream>
#include <fstream>
using namespace std;

const int MAXHW = 1000000;
int H = 0; // Height
int W = 0; // Wide
int R[MAXHW]; // Rows
int C[MAXHW]; // Columns
int ar, ac, cr, cc; // Toa do cac dinh A(ar,ac) C(cr,cc)
int HW; // dien tich phong thi
const char * fn = "Seats.inp";
```

```
ifstream f(fn); // Open file

// doc du lieu
void read() {
  f >> H >> W; // H dong, W cot
  HW = H*W; // Dien tich pHong thi
  // Thi sinh i ngoi tai dong R[i], cot C[i]
 for (int i = 0; i < HW; ++i) f >> R[i];
        for (int i = 0; i < HW; ++i) f >> C[i];
} // read

// Cap nhat A va C: A(ar,ac) <= C(cr,cc)
// return dien tich hinh D sau cap nhat
int updateAC(int m) {
   ar = min(ar,R[m]);
   ac = min(ac,C[m]);
   cr = max(cr,R[m]);
   cc = max(cc,C[m]);
   return (cr-ar+1)*(cc-ac+1);
} // updateAC

// m nam ngoai window[A;C]?
bool outAC(int m) {
 if (R[m] < ar || R[m] > cr)
    return 1;
  if (C[m] < ac || C[m] > cc)
    return 1;
  return 0;
}

// Dem so hinh dep bt
int beauty() {
  int bt = 0; // Dem so hinh beauty
  int m; // so ngoai hinh D
  // Init D[A;C] chua 0
  ar = cr = R[0];
  ac = cc = C[0];
```

```
  int s = 1; // Dien tich hinh D
  int bp = 0; // bp Diem cap nhat beautyPoint
  // HW la dien tich toan phong thi
  for (m = 1; m < HW; ++m) {
    if (outAC(m)) {
        // m nam ngoai D
        if (m == s) {
          ++bt;       // them mot hinh dep
          cout << "\n Cap nhat " << bp
                          << " thu duoc Hinh dep thu " << bt;
        }
        // m dang nam ngoai hinh
        // Mo rong window [A;C] de chua them so m
        bp = m; // ghi nhan dien cap nhat
        s = updateAC(m); // cap nhat [A;C] va tinh dien tich
    } // if outAC
  } // for
 ++bt; // Hinh toan the: dep
 cout << "\n Cap nhat " << bp
      << " thu duoc Hinh dep thu " << bt;
 return bt;
} // beauty

// Doi cho hai thi sinh a va b
void swap(int a, int b) {
  int t;
  cout << " Swap " << a << " " << b;
  t = R[a]; R[a] = R[b]; R[b] = t;
  t = C[a]; C[a] = C[b]; C[b] = t;
} // swap

void run() {
 read();
 int soTest;
 int a, b;
 f >> soTest;
 for (int test = 1; test <= soTest; ++test) {
   cout << "\n Test No " << test << ".";
```

```
   f >> a >> b;
   swap(a,b);
    int bt = beauty();
    cout << " \n result = " << bt;
 }
 f.close();
}  // run

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

Result

```
Test No 1. Swap 3 7
Cap nhat 0 thu duoc Hinh dep thu 1
Cap nhat 1 thu duoc Hinh dep thu 2
Cap nhat 3 thu duoc Hinh dep thu 3
Cap nhat 12 thu duoc Hinh dep thu 4
result = 4
```