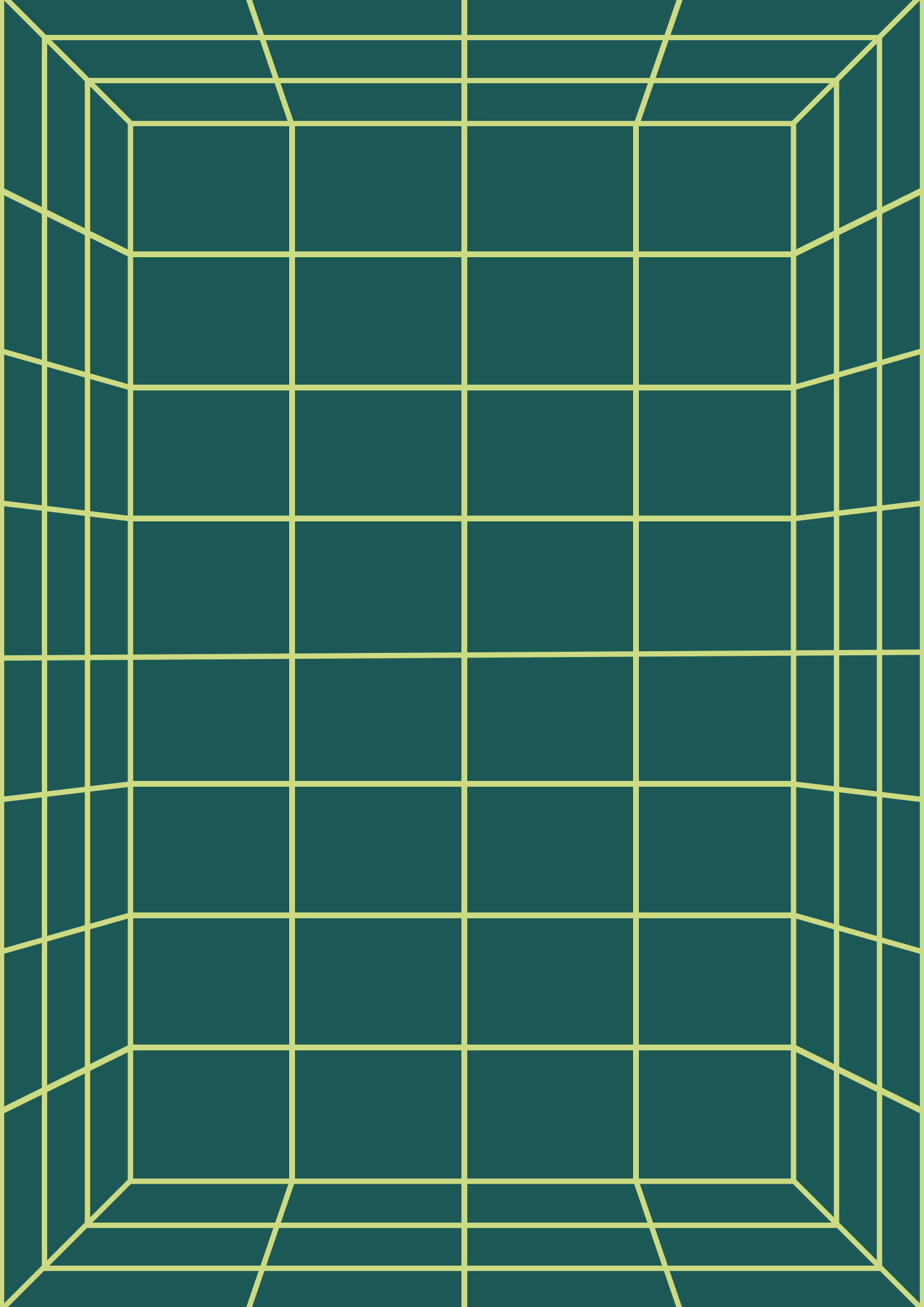


SAT
21/10/23

- Khi gặp 1 bài toán

① Mình làm bằng tay ntn thì code y như thế
② Nếu nhìn mãi ko ra? Học nhiều tập mẫu → Bổ đề
③ Nghĩ xem liệu nếu làm cách đó thì có OK ko? Nếu data lớn? Có CTDL nào dùng đc tốt hơn?

Bài toán lớn chưa làm đc thì giải bài toán con

Telco Data Check & Analyze

- Write a C++ program to perform some queries on a telco data (comming from stdin) with the following format:
- The first block of data consists of lines (terminated by a line containing #), each line (number of lines can be up to 100000) is under the form:

call <from_number> <to_number> <date> <from_time> <end_time>

- which is a call from the phone number <from_number> to a phone number <to_number> on <date>, and starting at time-point <from_time>, terminating at time-point <end_time>
- <from_number> and <to_number> are string of 10 characters (a phone number is correct if it contains only digits 0,1,...,9, otherwise, the phone number is incorrect)
- <date> is under the form YYYY-MM-DD (for example 2022-10-21)
- <from_time> and <to_time> are under the form hh:mm:ss (for example, 10:07:23)
- The second block consists of queries (terminated by a line containing #), each query in a line (number of lines can be up to 100000) and belongs to one of the following types:
- ?check_phone_number: print to stdout (in a new line) value 1 if no phone number is incorrect
- ?number_calls_from <phone_number>: print to stdout (in a new line) the number of times a call is made from <phone_number>
- ?number_total_calls: print to stdout (in a new line) the total number of calls of the data
- ?count_time_calls_from <phone_number>: print to stdout (in a new line) the total time duration (in seconds) the calls are made from <phone_number>

Telco Data Check & Analyze

- Example

stdin	stdout
call 0912345678 0132465789 2022-07-12 10:30:23 10:32:00 call 0912345678 0945324545 2022-07-13 11:30:10 11:35:11 call 0132465789 0945324545 2022-07-13 11:30:23 11:32:23 call 0945324545 0912345678 2022-07-13 07:30:23 07:48:30 # ?check_phone_number ?number_calls_from 0912345678 ?number_total_calls ?count_time_calls_from 0912345678 ?count_time_calls_from 0132465789 #	1 2 4 398 120

Telco Data Check & Analyze - Hint

- Use mapping data structure

  map<string, int>

  to map a phone number to some information

Implementation

```
#include <bits/stdc++.h>
using namespace std;
bool checkPhone (string s){
   if (s.length() != 10) return false;
   for (int i=0; i<s.length(); i++)
      if (!(s[i]>='0' && s[i]<='9')) return false;
   return true;
}
int countTime (string ftime, string etime){
   int startTime = 3600*((ftime[0]-'0')*10 + ftime[1]-'0') + 60*((ftime[3]-'0')*10 + ftime[4]-'0') +
            ((ftime[6]-'0')*10 + ftime[7]-'0');
   int endTime = 3600*((etime[0]-'0')*10 + etime[1]-'0') + 60*((etime[3]-'0')*10 + etime[4]-'0') +
            ((etime[6]-'0')*10 + etime[7]-'0');
   return endTime - startTime;
}
map <string,int> numberCalls, timeCall;
```

Implementation

```
int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(NULL);
    cout.tie(NULL);
    string type;
    int totalCalls = 0;
    int incorrectPhone = 0;
    do {
        cin >> type;
        if (type == "#") continue;
        ++totalCalls;
        string fnum, tnum, date, ftime, etime;
        cin >> fnum >> tnum >> date >> ftime >> etime;
        if (!checkPhone(fnum) || !checkPhone(tnum)) ++incorrectPhone;
        numberCalls[fnum]++;
        timeCall[fnum] += countTime(ftime, etime);
```

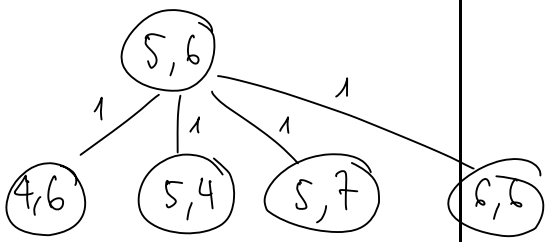
MAZE

- Một mê cung hình chữ nhật được biểu diễn bởi 0-1 ma trận NxM trong đó A[i,j] = 1 thể hiện ô (i,j) là tường gạch và A[i,j] = 0 thể hiện ô (i,j) là ô trống, có thể di chuyển vào. Từ 1 ô trống, ta có thể di chuyển sang 1 trong 4 ô lân cận (lên trên, xuống dưới, sang trái, sang phải) nếu ô đó là ô trống. Xuất phát từ 1 ô trống trong mê cung, hãy tìm đường ngắn nhất thoát ra khỏi mê cung.
- **Input**
- Dòng 1: ghi 4 số nguyên dương n, m, r, c trong đó n và m tương ứng là số hàng và cột của ma trận A (1 <= n,m <= 999) và r, c tương ứng là chỉ số hàng, cột của ô xuất phát.
- Dòng i+1 (i=1,...,n): ghi dòng thứ i của ma trận A
- **Output**
- Ghi giá số bước cần di chuyển ngắn nhất để thoát ra khỏi mê cung, hoặc ghi giá trị -1 nếu không tìm thấy đường đi nào thoát ra khỏi mê cung.

MAZE

- Example

stdin	stdout
8 12 5 6 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1	7



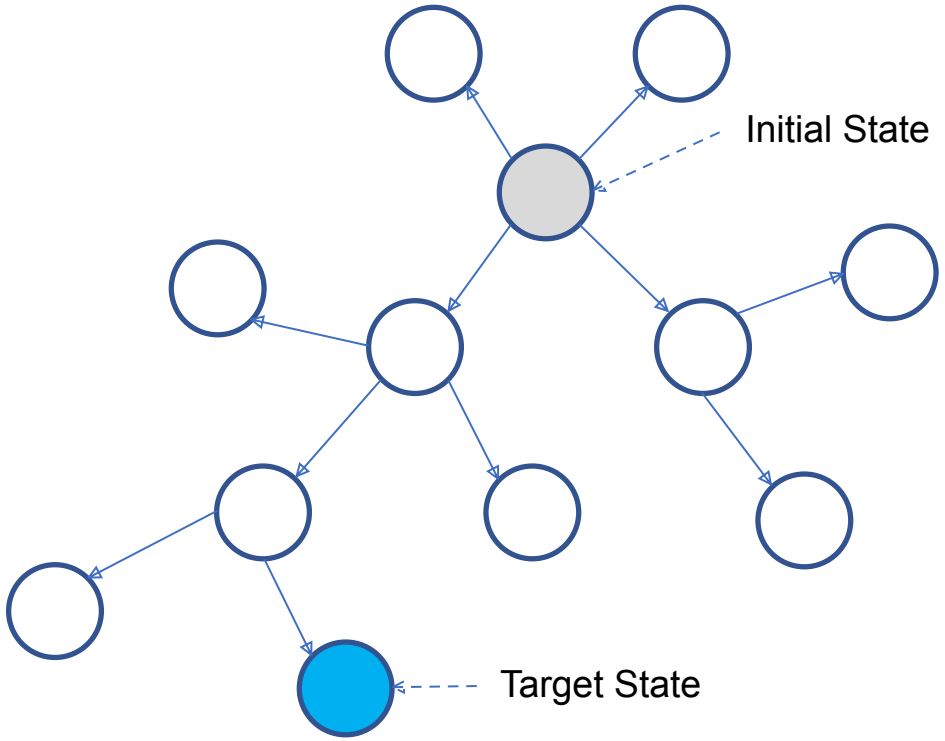
1 2 3 4 5 6 7 8 9 10 11 12

① Tại sao duyệt ?

② Tại sao dùng BFS ? Vì DFS ko đảm bảo tìm ra đường đi ngắn nhất

Implementation

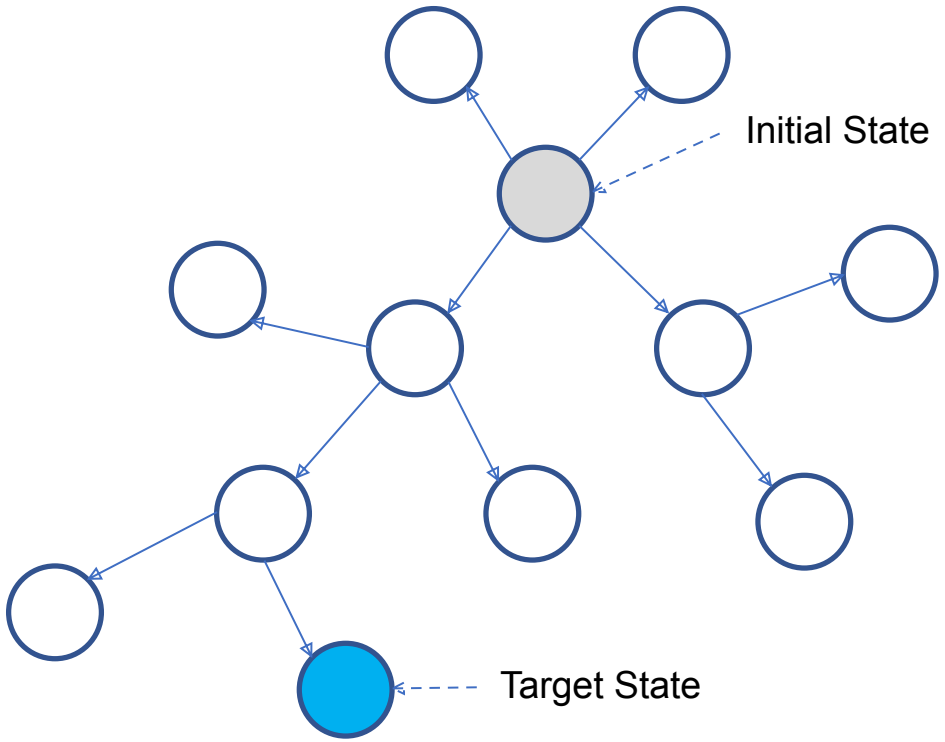
- Find the shortest path in the state transition diagram (graph)



Implementation

- Find the shortest path in the state transition diagram (graph)

```
findPath(s0, N){// N(s): set of neighboring states of s
 Initialize a Queue
 Queue.PUSH(s0); visited[s0] = true;
 while Queue not empty do{
  s = Queue.POP();
  for x ∈ N(s) do{
    if not visited[x] and check(x) then{
        if target(x) then return x;
        else{
          Queue.PUSH(x);  visited[x] = true;
        }
    }
  }
 }
}
```



Implementation

```
#include <bits/stdc++.h>

using namespace std;

typedef pair<int,int> ii;

const int maxN = 999 + 100;

const int oo= 1e9 + 7;

int a[maxN][maxN] , m , n , r, c , d[maxN][maxN];

int dx[] = {1 , 0, -1 ,  0} ,

   dy[] = {0 , 1,  0 , -1};

queue<ii> qe;
```

Implementation

```
int solve(){
  qe.push(ii(r,c));  d[r][c] = 0;   a[r][c] = 1;
  while(!qe.empty()){
    ii u = qe.front(); qe.pop();
    for(int i = 0 ; i < 4 ; i++){
      int x = dx[i] + u.first;        int y = dy[i] + u.second;
      if(x < 1 || x > m || y < 1 || y > n) return d[u.first][u.second] + 1;
      if(a[x][y] != 1){
          d[x][y] = d[u.first][u.second] + 1;
          qe.push(ii(x,y));
          a[x][y] = 1;
      }
    }
  }
  return -1;
}
```

Implementation

```
int main(){
  ios_base::sync_with_stdio(false);cin.tie(0);
  cin >> m >> n >> r >> c;
  for(int i = 1 ; i <= m ; i++) for(int j = 1 ; j <= n ; j++) cin >> a[i][j];
  int ans = solve();
  cout << ans;
  return 0;
}
```

Largest Black SubRectangle

- Một hình chữ nhật kích thước *n* x *m* được chia thành các ô vuông con 1 x 1 với 2 màu đen hoặc trắng. Hình chữ nhật được biểu diễn bởi ma trận *A*(*n* x *m*) trong đó *A*(*i, j*) = 1 có nghĩa ô hàng *i*, cột *j* là ô đen và *A*(*i, j*) = 0 có nghĩa ô vuông hàng *i* cột *j* là ô trắng.
- Hãy xác định hình chữ nhật con của bảng đã cho bao gồm toàn ô đen và có diện tích lớn nhất.
- **Dữ liệu**
- · Dòng 1: chứa số nguyên dương *n* và *m* (1 <= *n, m* <= 1000)
- · Dòng i+1 (i = 1,…, *n*): chứa hàng thứ i của ma trận *A*
- **Kết quả**
- · Ghi ra diện tích của hình chữ nhật lớn nhất tìm được

Largest Black SubRectangle

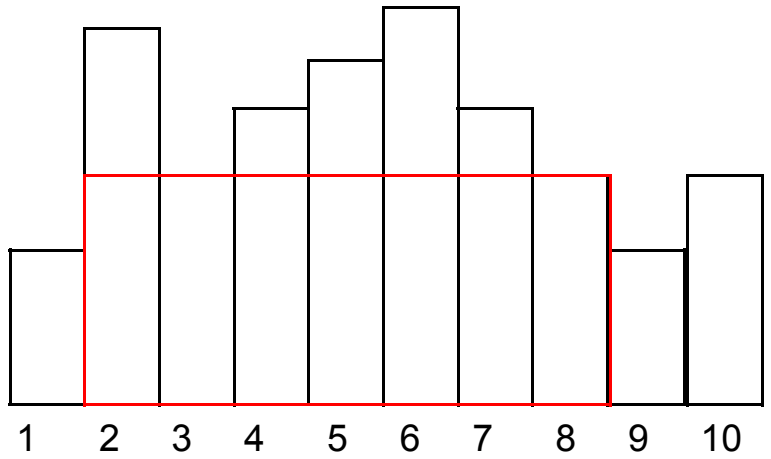
- Example

stdin	stdout
4 4 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0	6

Hint

dãy liên (ô có ô trống)

- Solve sub-problem: Column i have height $h[i]$ (i = 1, 2, …, n). Find the way to cut out the largest-area rectangle from the given configure
- For each index i :
  - Move left and move right as far as possible to cut out the largest rectangle having height $h[i]$
  - $R[i]$: the index j such that $h[i] > h[j]$ and j ($i < j$) is the nearest to i
  - $L[i]$: the index j such that $h[i] > h[j]$ and j ($i > j$) is the nearest to i
  - The largest are built from column i is: $(R[i] – L[i] - 1)*h[i]$
- Use a Stack S for storing indices




```
h[0] = -1, h[n+1] = -1
S □ empty stack
for i = 1 □ n+1 do{
   while S not empty and h[i] < h[S.top] do{
        R[S.top] = i; S.pop();
   }
   S.push(i);
}
```

```
h[0] = -1, h[n+1] = -1
S □ empty stack
for i = n □ 0 do{
   while S not empty and h[i] < h[S.top] do{
        L[S.top] = i; S.pop();
   }
   S.push(i);
}
```

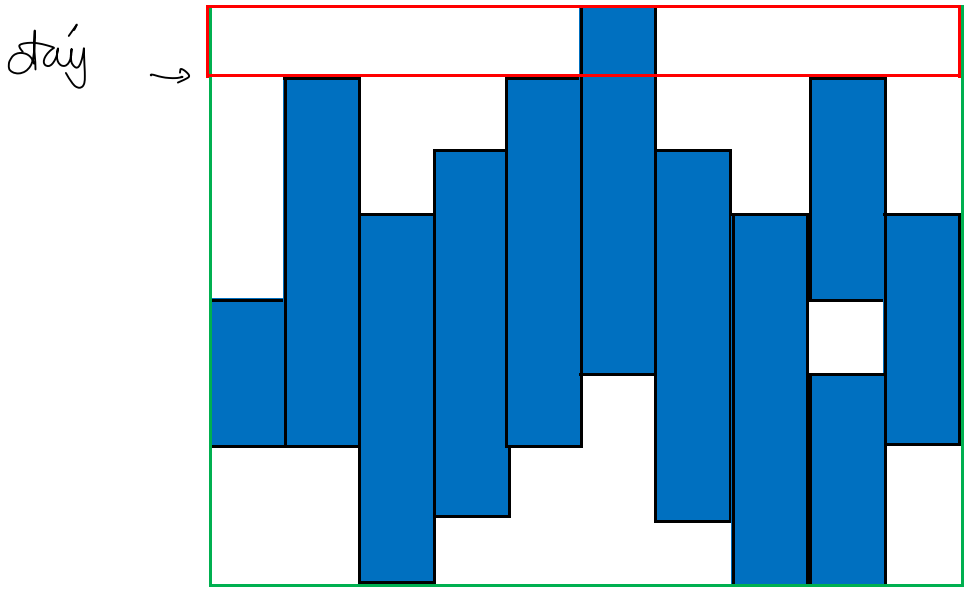
Hint

- Let a horizontal line moving down

↳ Chuyển thành bài toán con:

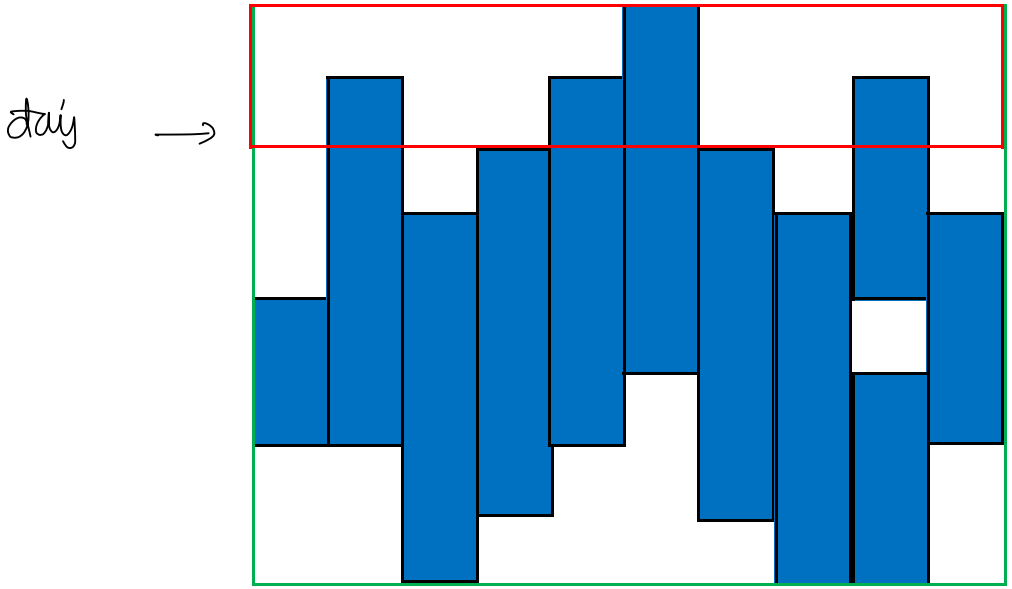
Trượt xuống tới đâu thì coi đó là đáy mới:

- Nếu ở 1 cột, đáy rỗng: coi như độ cao cột đó = 0



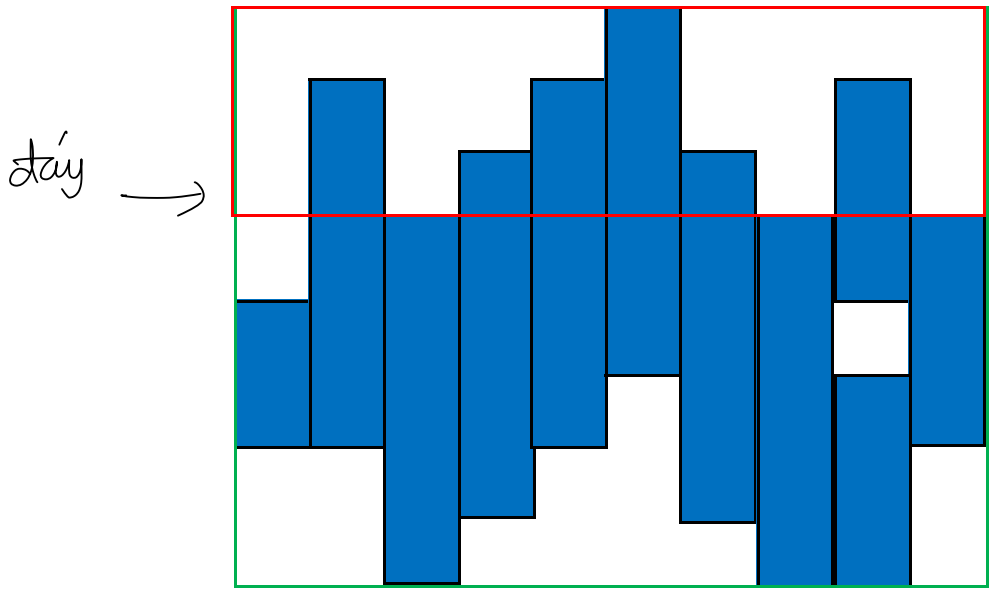
Hint

- Let a horizontal line moving down



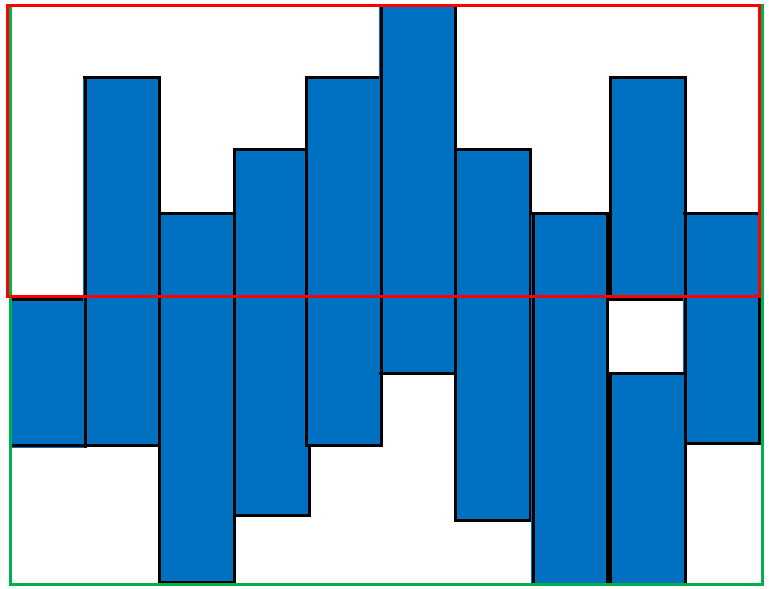
Hint

- Let a horizontal line moving down



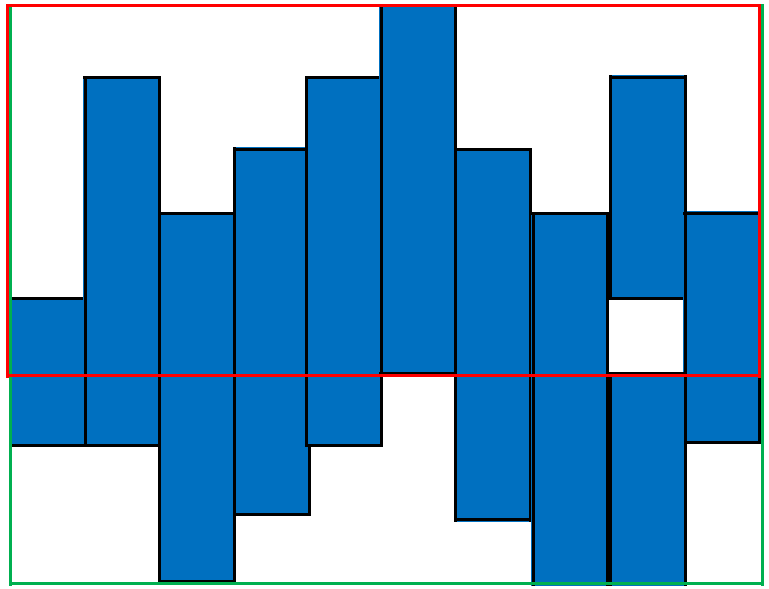
Hint

- Let a horizontal line moving down



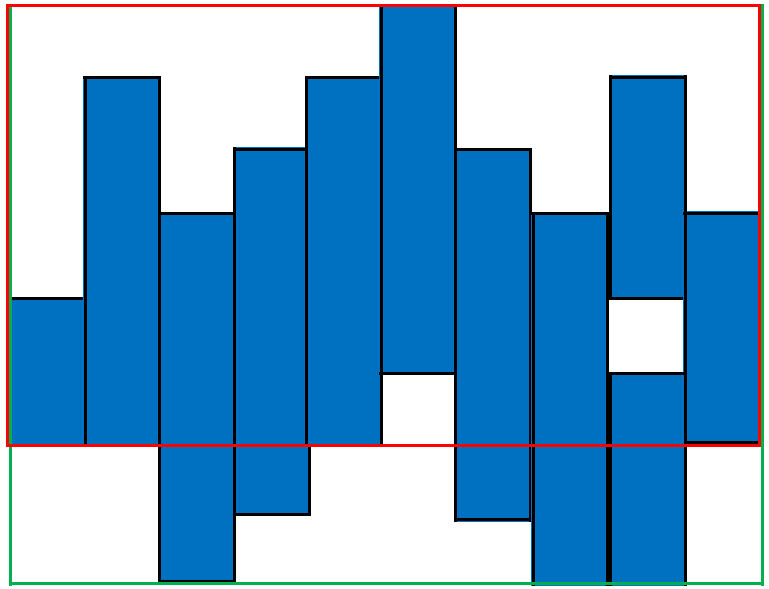
Hint

- Let a horizontal line moving down



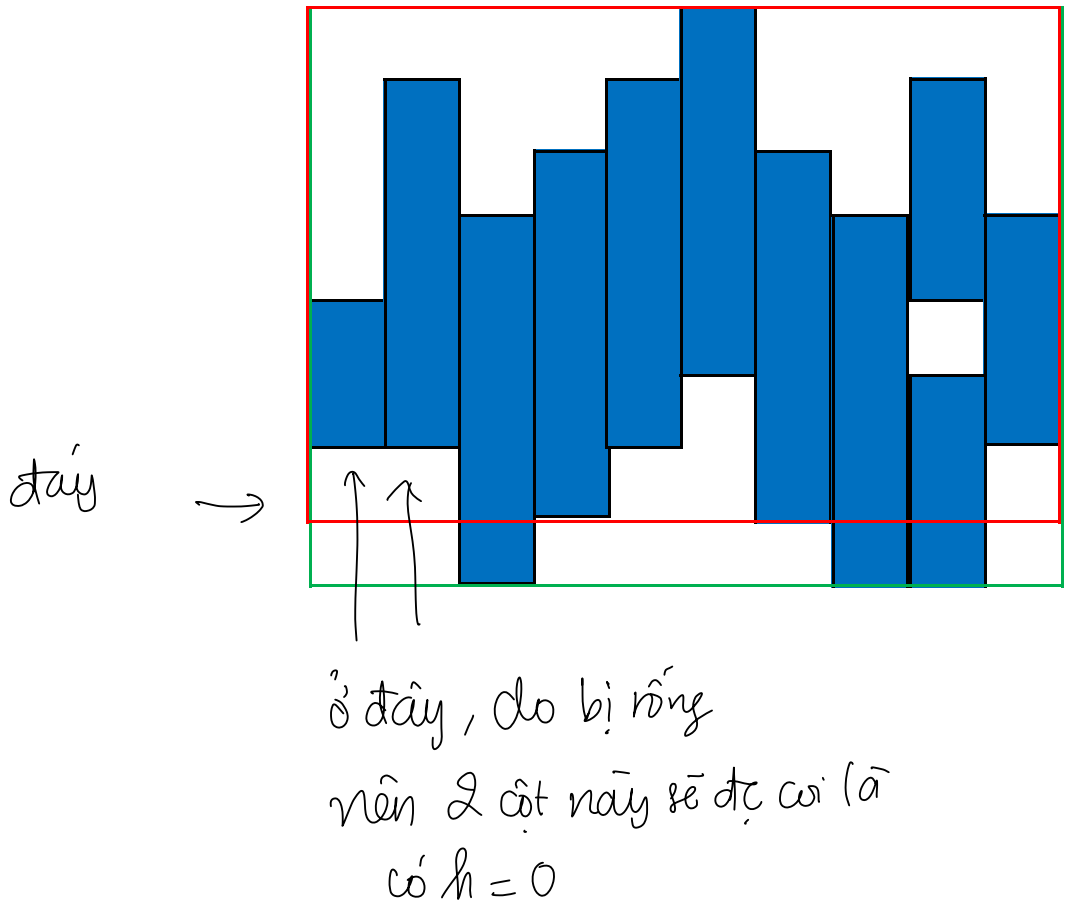
Hint

- Let a horizontal line moving down



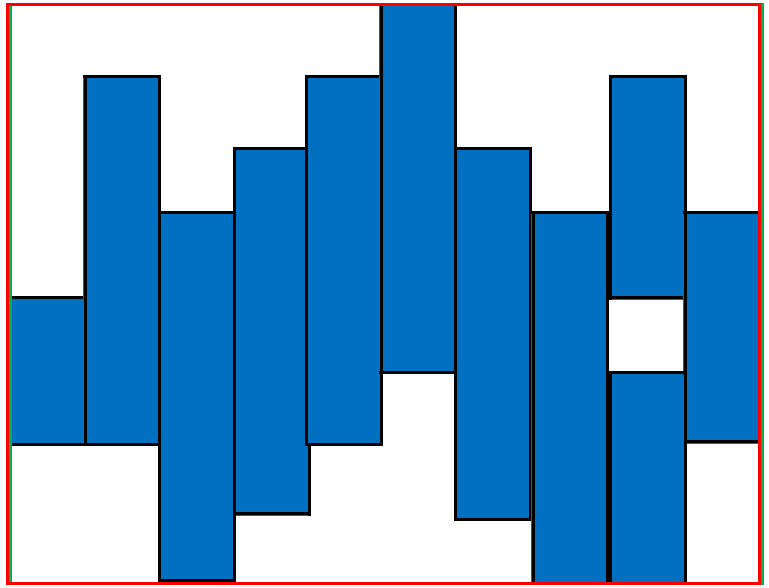
Hint

- Let a horizontal line moving down



Hint

- Let a horizontal line moving down



Implementation

```
#include <bits/stdc++.h>

using namespace std;

const int N = 1e3+1;

int a[N][N];

int n,m;

int ans;

long long h[N];

stack<long long> S;

vector<long long> V;

long long L[N],R[N] ;

void input(){
  cin >> n >> m;
  for(int i = 1; i <= n; i++){
     for(int j = 1; j <= m; j++) cin >> a[i][j];
  }
}
```

Implementation

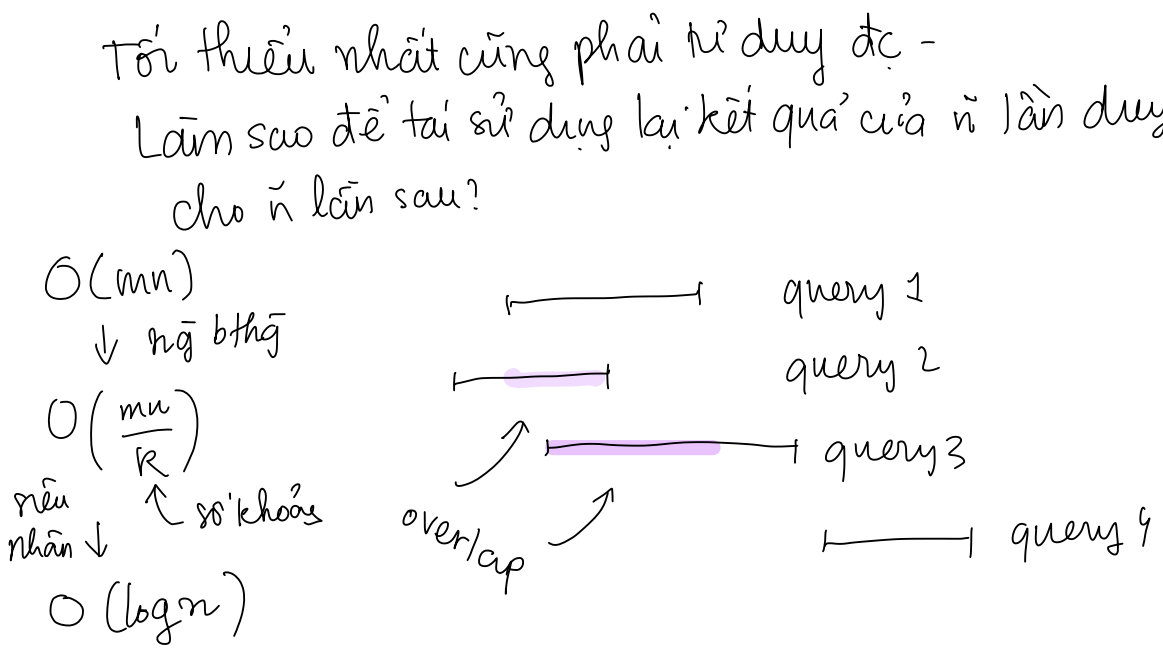
```
long long compute(){
        h[0] = -1; h[m+1] = -1;  V.clear();
        for(int i = 1; i <= m+1; i++){
            while(!V.empty() && h[i] < h[V[V.size()-1]]){      R[V[V.size()-1]] = i; V.pop_back();     }
            V.push_back(i);
        }
        for(int i = m; i >= 0; i--){
           while(!V.empty() && h[i] < h[V[V.size()-1]]){   L[V[V.size()-1]] = i; V.pop_back();   }
           V.push_back(i);
        }
        unsigned long long ans = 0;
        for(int i = 1; i <= m; i++){
           unsigned long long c = (R[i] - L[i] - 1)*h[i];     ans = ans < c ? c : ans;
        }
        return ans;
}
```

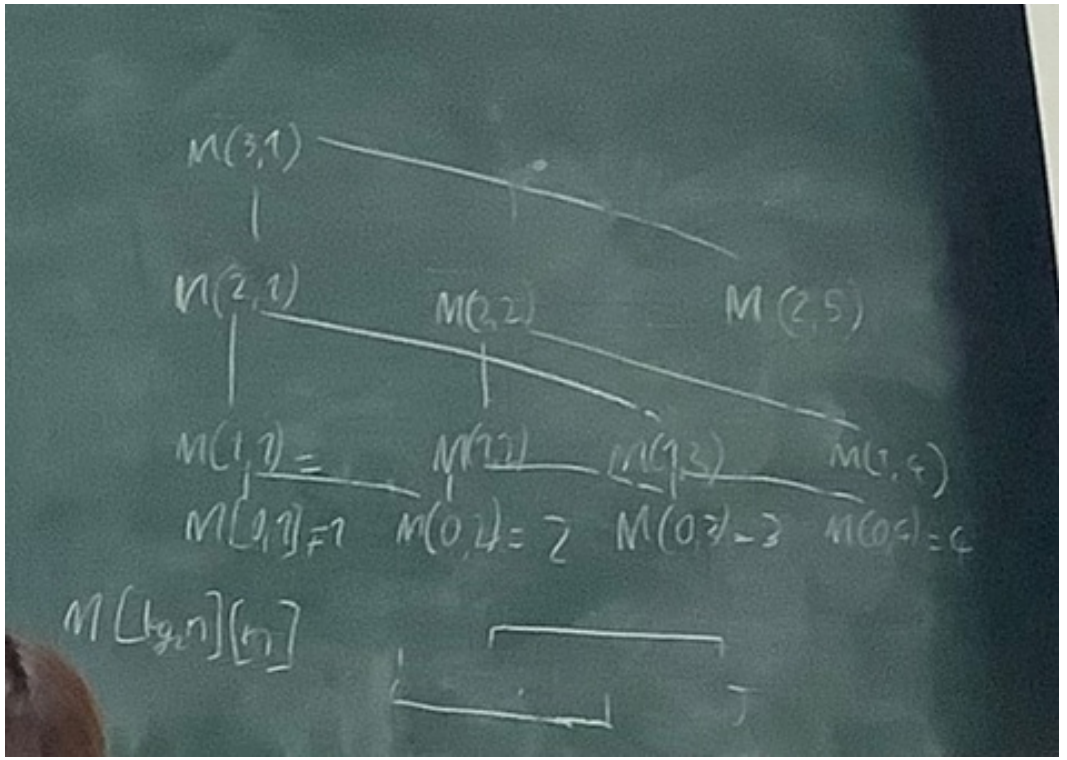
Implementation

```
void solve(){
   long long ans = 0;
   for(int i = 1; i <= m; i++) h[i] = 0;
   for(int i =1 ; i <= n; i++){
      for(int j = 1; j <= m; j++){  if(a[i][j] == 0)   h[j] = 0; else h[j] += 1;  }
      long long t = compute();
      if(t > ans) ans = t;
   }
   cout << ans;
}
int main(){
   input();
   solve();
   return 0;
}
```

Range Minimum Query

- Given a sequence of *n* integers *a*0,. . ., *an*-1. We denote rmq(i, j) the minimum element of the sequence *ai*, *ai*+1, . . ., *aj*. Given *m* pairs (*i*1, *j*1),. . ., (*im*, *jm*), compute the sum Q = rmq(*i*1, *j*1) + . . . + rmq(*im*, *jm*)
- **Input**
- Line 1: contains an integer *n* (1 <= *n* <= 106)
- Line 2: contains *a*0, . . . , *an*-1 (1 <= *ai* <= 106)
- Line 3: contains *m* (1 <= *m* <= 106)
- Line *k*+3 (k = 1, . . ., *m*): contains *ik*, *jk* (0 <= *ik* < *jk* < *n*)
- **Output**
- Write the value Q





Range Minimum Query

- Example

stdin	stdout
16 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 2 4 6 1 6 8 7 3 3 5 8 9 1 2 6 4 4 1 5 → 1 0 9 → 1 1 15 → 1 6 10 → 3	6

Hint

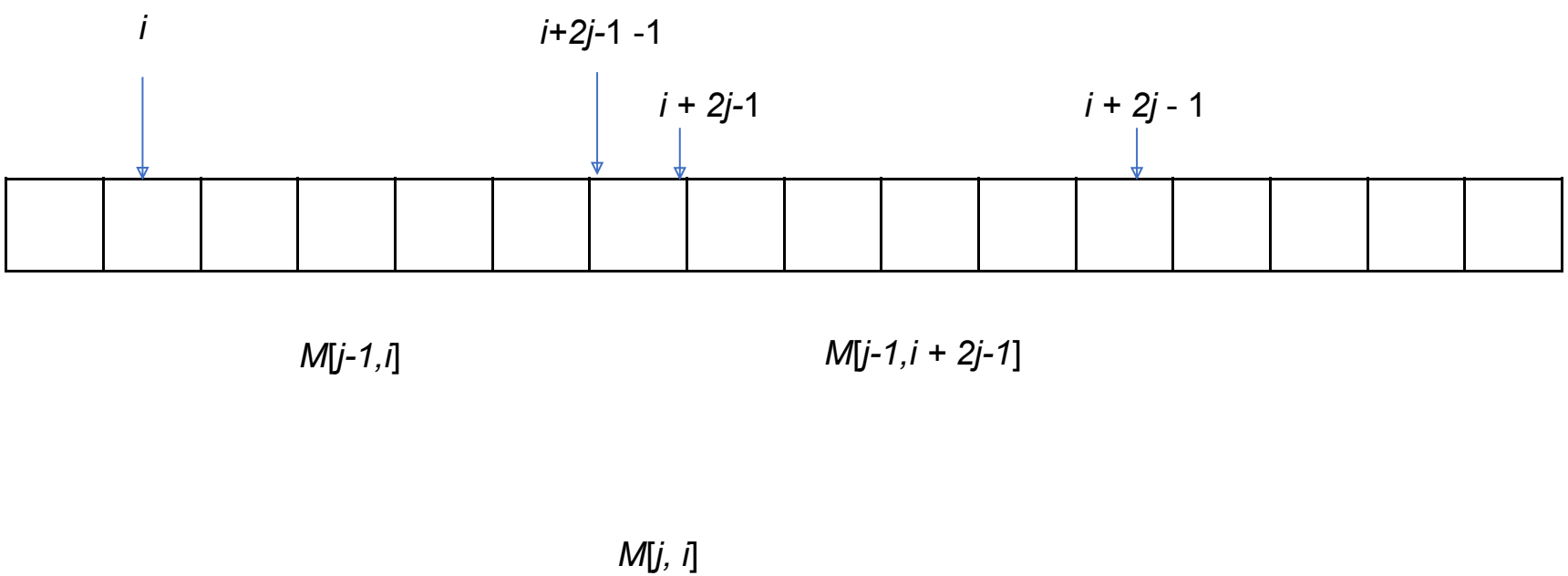
- Denote $M[j, i]$ the index of the smallest element of $a[i]$, $a[i+2]$,…, $a[i+2j-1]$ (the sequence from index i and has the length $2j$).

data struct : segment tree

1	0	1	3	3	4	6	7	8	8	9	10	12	12	13	15	-
2	3	3	3	3	7	8	8	8	8	12	12	12	12	-	-	-
3	3	3	3	3	8	12	12	12	12	-	-	-	-	-	-	-
4	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Hint

- $M[0,i] = i,\ i = 0,\ldots,\ N-1$
- Recurrence relation:
- $M[j,i] = M[j-1,i]$ if $a[M[j-1,i]] < a[M[j-1,i+2j-1]$
  $M[j-1,i+2j-1]$, otherwise

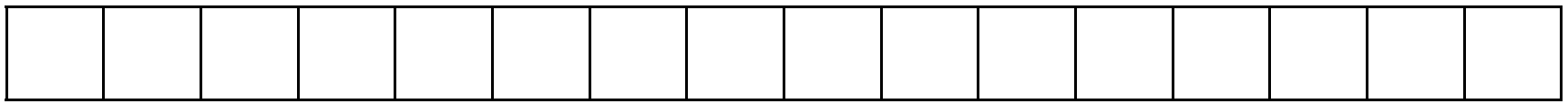


Hint

- Query RMQ(i,j): the index of the smallest element of the sequence $a[i]$, $a[i+1]$, . . ., $a[j]$
- $k = [\log(j-i+1)]$
- RMQ(i,j) = $M[k,i]$ if $a[M[k,i]] \leq a[M[k, j-2k+1]]$

  $M[k, j-2k+1]]$, otherwise
- RMQ(4,14) = ?
  - $k = [\log(14-4+1)]=3$
  - $a[7] > a[12]$ □ RMQ(4,14) = 12

M[3,7] = 12



M[3,4] = 7

Implementation

```
#include <bits/stdc++.h>

using namespace std;

int n;

int M[30][1000000];

int A[1000000];

void preprocessing(){
  for(int j = 0; (1 << j) <= n; j++){
    for(int i = 0; i < n; i++) M[j][i] = -1;
  }
  for(int i = 0; i < n; i++) M[0][i] = i;
  for(int j = 1; (1 << j) <= n; j++){
    for(int i = 0; i + (1 << j) - 1 < n; i++){
      if(A[M[j-1][i]] < A[M[j-1][i+(1 << (j-1))]])  M[j][i] = M[j-1][i];  else  M[j][i] = M[j-1][i + (1 << (j-1))];
    }
  }
}
```

Implementation

```
int rmq(int i, int j){
   int k = log2(j-i+1);
   int p2k = (1 << k);//pow(2,k);
   if(A[M[k][i]] <= A[M[k][j-p2k+1]]){
      return M[k][i];
   }else{
      return M[k][j-p2k+1];
   }
}
```

Implementation

```
int main(){
  scanf("%d",&n);
  for(int i = 0; i < n; i++)   scanf("%d",&A[i]);


  preprocessing();


  int ans = 0; int m;
   scanf("%d",&m);
   for(int i = 0; i < m; i++){
       int I,J;  scanf("%d%d",&I,&J);
       ans += A[rmq(I,J)];
   }
   cout << ans;
   return 0;
}
```

Implementation

```
do {
    cin >> type;
    if (type == "#") continue;
    if (type == "?check_phone_number") {
        if (incorrectPhone == 0) cout << 1 << endl;  else cout << 0 << endl;
    } else if (type == "?number_calls_from") {
        string phone;   cin >> phone;
        cout << numberCalls[phone] << endl;
    }else if (type == "?number_total_calls")
        cout << totalCalls << endl;
    else if (type == "?count_time_calls_from") {
        string phone;  cin >> phone;
        cout << timeCall[phone] << endl;
    }
}while (type!="#");
return 0;
```