Week 3 Programming Assignment

Due on 2019-02-23, 23:59 IST

- Select your language (C/C++/Java/Python2/Python3)
- Paste your code into the submission window.
- There are some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases and report a score on 100. There are 20 private testcases in all, each with equal weightage. You will only get a score on 100. You will not get feedback on which private testcases passed or failed.
- Ignore warnings about "Presentation errors".

Prisoner Escape

(Baltic Olympiad in Informatics, 2009)

A group of war prisoners are trying to escape from a prison. They have thoroughly planned the escape from the prison itself, and after that they hope to find shelter in a nearby village. However, the village (marked as B, see picture below) and the prison (marked as A) are separated by a canyon which is also guarded by soldiers. These soldiers sit in their pickets and rarely walk; the range of view of each soldier is limited to exactly 100 meters. Thus, depending on the locations of soldiers, it may be possible to pass the canyon safely, keeping the distance to the closest soldier strictly larger than 100 meters at any moment.

You are to write a program which, given the width and the length of the canyon and the coordinates of every soldier in the canyon, and assuming that soldiers do not change their locations, determines whether prisoners can pass the canyon unnoticed.

Solution Hint

The soldiers can be modelled as an undirected graph G. Let each soldier be represented by a vertex. Add an edge between two vertices if and only if the range of view of the corresponding soldiers overlaps. Add two additional vertices s and t, representing the northern and southern side of the canyon, respectively. Connect s and t to those vertices representing soldiers who range of view includes the respective side of the canyon. Use depth-first-search or breadth-first-search to check whether there is a path between s and t in G.

Input format

The first line contains three integers L, W, and N – the length and the width of the canyon, and the number of soldiers, respectively. Each of the following N lines contains a pair of integers Xi and Yi – the coordinates of i-th soldier in the canyon (0 ≤ Xi ≤ L, 0 ≤ Yi ≤ W). The coordinates are given in meters, relative to the canyon: the southwestern corner of the canyon has coordinates (0, 0), and the northeastern corner of the canyon has coordinates (L,W), as seen in the picture above. Note that passing the canyon may start at coordinate (0,ys) for any 0 ≤ ys ≤ W and end at coordinate (L,ye) for any 0 &\le; ye ≤ W. Neither ys nor ye need to be integer.

Faceb
Twitte
Youtu
Linke
Googl plus

Output format

Output a single integer: 0 if the prisoners can escape, 1 if they cannot.

Test data

1 ≤ W ≤ 50,000; 1 ≤ L ≤ 50,000; 1 ≤ N ≤ 250.

Example

Sample input 1

```
130 340 5
10 50
130 130
70 170
0 180
60 260
```

Sample output 1

```
1
```

Sample input 2

```
500 300 4
250 0
250 300
100 150
400 150
```

Sample output 2

```
0
```

Sample Test Cases

	Input	Output

Test Case 1	130 340 5
	10 50
	130 130
	70 170
		1

```
884 1461
984 1461
1084 1461
1184 1461
1284 1461
1384 1461
1484 1461
1584 1461
739 231
739 331
739 431
739 531
739 631
739 731
739 831
739 931
739 1031
739 1131
1211 95
1311 95
1411 95
1511 95
```

Due Date Exceeded.
20 out of 20 tests passed.
You scored 100.0/100.

Your last recorded submission was :

```
#include<iostream>
#include<vector>
#include<cmath>
#include<map>
#include<queue>
#include<algorithm>
#include<iterator>
#include<list>
#include<cstdlib>
using namespace std;

int l,w,n;
map<int,pair<long long int,long long int> >mp;
bool dist(int i,int j){
    long long int y=pow((mp[i].second-mp[j].second),2);
    long long int x=pow((mp[i].first-mp[j].first),2);
    if(x+y<=pow(200,2))
//      cout<<"x+y "<<x+y<<" "<<pow(200,2)<<endl;
        return true;

    else
    return false;
}


class Graph{
    public:
    int v;
    list<int >*adj;

        Graph(int v);
        void addedge(int v,int w);
    bool ispath(void);

};

Graph::Graph(int v){
    this->v=v;
    adj=new list<int>[v];
}
void Graph::addedge(int v,int w){
    adj[v].push_back(w);
    adj[w].push_back(v);
}
```

```

bool Graph::ispath()
{
    list<int> queue;
    bool *visited = new bool[n];
    for(int i = 0; i < n; i++)
        visited[i] = false;

    for(int i=0;i<n;++i){
      if(mp[i].second<=100)
        queue.push_back(i);

    }


    list<int>::iterator i;

    while(!queue.empty())
    {
        int s = queue.front();
        queue.pop_front();

        for (i = adj[s].begin(); i != adj[s].end(); ++i)
        {
          if(!visited[*i]){
                 if(abs(w-mp[*i].second)<=100)
                   return true;
             else {
                     visited[*i]=true;
               queue.push_back(*i);
                   }
            }
        }
    }
    return false;
}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin>>l>>w>>n;
    Graph g(n);
    for(int i=0;i<n;++i) cin>>mp[i].first>>mp[i].second;

    for(int i=0;i<n-1;++i)
        for(int j=i+1;j<n;++j)
            if(dist(i,j))
                g.addedge(i,j);


if(g.ispath())
   cout<<"1"<<endl;
else
    cout<<"0"<<endl;


}
```

Faceb

Twitte

Youtu

Linke

Googl
plus

End