Queen's Attack II

Editorial by bishop15

In the problem there is a chessboard of dimension $n \times n$. In one cell there is a queen, also there are k obstacles that block the queen's path. You have to determine the number of **free cells** the queen can attack.

$O(n)$ Approach:

You can iterate over the cells the queen can attack and stop when there is an obstacle or you reach the edge of the board. To do that, you need to iterate horizontally, vertically and diagonally. The moves can be like this:

- Assume, (x, y) is the current position.
- $(x+1, y)$: one step horizontal move to the right.
- $(x-1, y)$: one step horizontal move to the left.
- $(x+1, y+1)$: one step diagonal move up-right
- $(x-1, y-1)$: one step diagonal move down-left
- $(x-1, y+1)$: one step diagonal move left-up
- $(x+1, y-1)$: one step diagonal move right-down

Tester's solution code is based on this approach.

$O(K)$ Approach:

The key idea for this approach is that you can iterate over the obstacles and for those that are in the queen's path, calculate the free cells up to that obstacle. If there is no obstacle in the path you have to calculate the number of free cells up to the edge of the board in that direction.

For any (x_1, y_1) and x_2, y_2:

- If they are in the same row: abs($x_1 - x_2 - 1$) is the number of free cells between them.
- If they are in the same column: abs($y_1 - y_2 - 1$) is the number of free cells between them.
- If they are diagonal, either abs($x_1 - x_2 - 1$) or abs($y_1 - y_2 - 1$) is the number of free cells between them.

Setter's solution code is based on this approach.

Set by bishop15

Problem Setter's code:

```
#include <bits/stdc++.h>
using namespace std;

#define IN(a,x,y) (a>=x && a<=y)

const int MAXOBS = 100000;

int N, K;
int ox[MAXOBS+10], oy[MAXOBS+10];

int val(int x, int y)
{
    int i;

    int d11,d12,d21,d22,r1,r2,c1,c2;

    d11 = min( x-1, y-1 );
    d12 = min( N-x, N-y );
    d21 = min( N-x, y-1 );
    d22 = min( x-1, N-y );

    r1 = y-1;
    r2 = N-y;
    c1 = x-1;
    c2 = N-x;
    for(i=0; i<K; i++)
    {
```

STATISTICS

Difficulty:	Medium
Time Complexity:	O(N) or O(K)
Required Knowledge:	AdHoc, Basic Geometry, Slope
Publish Date:	Nov 30 2016

Originally featured in World CodeSprint 9

NEED HELP?

View discussions

View top submissions

```
        if( x>ox[i] && y>oy[i] && x-ox[i] == y-oy[i] ) d11 = min( d11, x-ox[i]-1 );
        if( ox[i]>x && oy[i]>y && ox[i]-x == oy[i]-y ) d12 = min( d12, ox[i]-x-1 );
        if( ox[i]>x && y>oy[i] && ox[i]-x == y-oy[i] ) d21 = min( d21, ox[i]-x-1 );
        if( x>ox[i] && oy[i]>y && x-ox[i] == oy[i]-y ) d22 = min( d22, x-ox[i]-1 );

        if( x == ox[i] && oy[i]<y ) r1 = min( r1, y-oy[i]-1 );
        if( x == ox[i] && oy[i]>y ) r2 = min( r2, oy[i]-y-1 );
        if( y == oy[i] && ox[i]<x ) c1 = min( c1, x-ox[i]-1 );
        if( y == oy[i] && ox[i]>x ) c2 = min( c2, ox[i]-x-1 );
    }

    return d11 + d12 + d21 + d22 + r1 + r2 + c1 + c2;
}
int main(void)
{
    int i,j,k,kase=0;

    int x,y;
    scanf("%d %d",&N, &K);
    scanf("%d %d",&x, &y);

    assert(IN(N,1,100000));
    assert(IN(K,0,100000));

    for(i=0; i<K; i++)
    {
        scanf("%d %d",&ox[i], &oy[i]);
        assert(IN(ox[i],1,N) && IN(oy[i],1,N) && (ox[i]!=x || oy[i]!=y));
    }

    printf("%d\n",val(x, y));
    return 0;
}
```

Tested by allllekssssa

Problem Tester's code:

```
#include<stdio.h>
#include<map>
#include<iostream>
#include<assert.h>

using namespace std;

map <pair<int,int>,int> mp;

int ans,n,x,y,x1,y1,k;

const int maxi=1e5;

int range(int x,int y)
{
    return(x<=n && x>0 && y<=n && y>0);
}

void check(int x,int y,int xx, int yy)
{
    while(range(x,y) && !mp[{x,y}])
     {
        x+=xx;
        y+=yy;
        ans++;
     }
}

int main()
{
    scanf("%d%d",&n,&k);

    assert(0<n && n<=maxi);
    assert(0<=k && k<=maxi);

    cin>>x>>y;
    assert(x>0 && x<=n);
    assert(y>0 && y<=n);


    while(k--)
```

```
    {
        scanf("%d%d",&x1,&y1);

        assert(x1!=x || y1!=y);
        assert(x1>0 && x1<=n);
        assert(y1>0 && y1<=n);

        mp[{x1,y1}]=1;
    }

     check(x+1,y,1,0);
     check(x-1,y,-1,0);
     check(x,y+1,0,1);
     check(x,y-1,0,-1);
     check(x+1,y+1,1,1);
     check(x+1,y-1,1,-1);
     check(x-1,y+1,-1,1);
     check(x-1,y-1,-1,-1);

     printf("%d\n",ans);
    return 0;
}
```

Feedback

Was this editorial helpful?

Yes No