hackerrank.com/challenges/queens-attack-2/problem?isFullScreen=true

HackerRank | Prepare > Algorithms > Implementation > Queen's Attack II

Exit Full Screen View

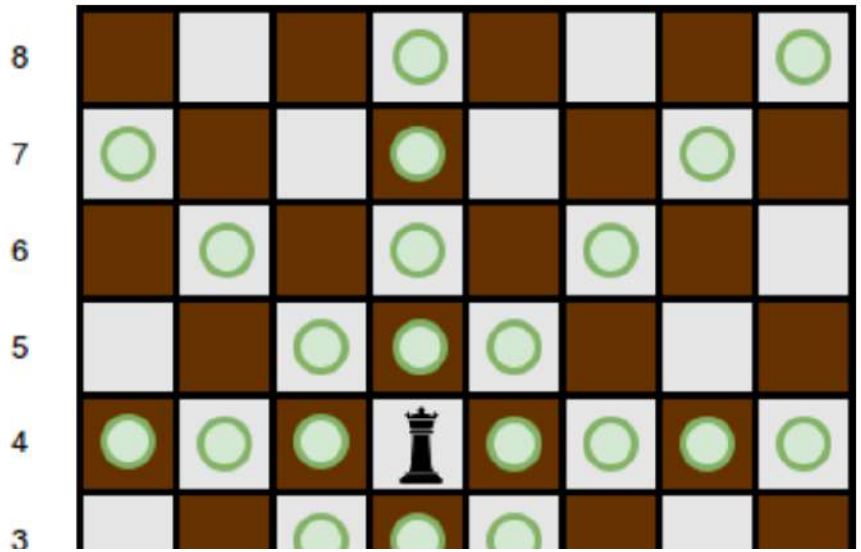
Problem | Submissions | Leaderboard | Discussions

You will be given a square chess board with one queen and a number of obstacles placed on it.

Determine how many squares the queen can attack.

A queen is standing on an $n \times n$ chessboard. The chess board's rows are numbered from 1 to n, going from bottom to top. Its columns are numbered from 1 to n, going from left to right. Each square is referenced by a tuple, (r, c), describing the row, r, and column, c, where the square is located.

The queen is standing at position (r_q, c_q). In a single move, she can attack any square in any of the eight directions (left, right, up, down, and the four diagonals). In the diagram below, the green circles denote all the cells the queen can attack from $(4, 4)$:



Change Theme Language C++11

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

int queensAttack(int n, int k, int r_q, int c_q, vector<vector<int>> obstacle
{
    int up = n - r_q;
    int down = r_q - 1;
    int left = c_q - 1;
    int right = n - c_q;
    int upLeft = min(n - r_q, c_q - 1);
    int upRight = min(n - r_q, n - c_q);
    int downLeft = min(r_q - 1, c_q - 1);
    int downRight = min(r_q - 1, n - c_q);

    for (const auto& obs : obstacles) {
        int r_obs = obs[0];
        int c_obs = obs[1];

        if (r_obs == r_q) {
            if (c_obs < c_q) left = min(left, c_q - c_obs - 1);
            else right = min(right, c_obs - c_q - 1);
        } else if (c_obs == c_q) {
            if (r_obs < r_q) down = min(down, r_q - r_obs - 1);
            else up = min(up, r_obs - r_q - 1);
        } else if (abs(r_obs - r_q) == abs(c_obs - c_q)) {
```

Line: 52 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

hackerrank.com/challenges/queens-attack-2/problem?isFullScreen=true

HackerRank | Prepare > Algorithms > Implementation > Queen's Attack II

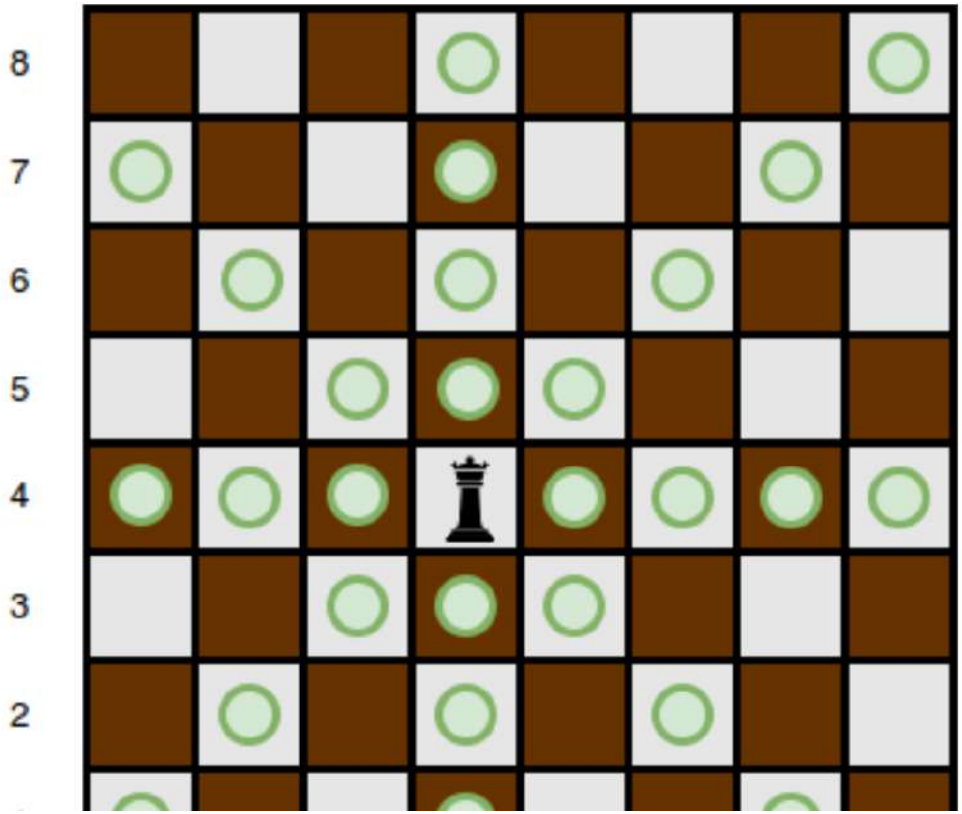
Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions | Editorial

You will be given a square chess board with one queen and a number of obstacles placed on it. Determine how many squares the queen can attack.

A queen is standing on an $n \times n$ chessboard. The chess board's rows are numbered from 1 to n, going from bottom to top. Its columns are numbered from 1 to n, going from left to right. Each square is referenced by a tuple, (r, c), describing the row, r, and column, c, where the square is located.

The queen is standing at position (r_q, c_q). In a single move, she can attack any square in any of the eight directions (left, right, up, down, and the four diagonals). In the diagram below, the green circles denote all the cells the queen can attack from $(4, 4)$:




```
            else if (r_obs < r_q && c_obs < c_q) downLeft = min(downLeft, r_q - r_obs - 1);
            else if (r_obs < r_q && c_obs > c_q) downRight = min(downRight, r_q - r_obs - 1);
        }
    }

    return up + down + left + right + upLeft + upRight + downLeft + downRight;
}

int main() {
    int n, k;
    cin >> n >> k;

    int r_q, c_q;
    cin >> r_q >> c_q;

    vector<vector<int>> obstacles(k, vector<int>(2));
    for (int i = 0; i < k; i++) {
        cin >> obstacles[i][0] >> obstacles[i][1];
    }

    int result = queensAttack(n, k, r_q, c_q, obstacles);
    cout << result << endl;

    return 0;
}
```

Line: 52 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) — Download

```
4 0
4 4
```

Your Output (stdout)

```
9
```

Expected Output — Download

```
9
```