We can put obstacles around (x1,y1)(x1,y1) or (x2,y2)(x2,y2) and the better one is the answer. More formally, let's define a function ff:

f(x,y)=⎧⎩⎨2,3,4,(x,y) is on the corner(x,y) is on the border(x,y) is in the middlef(x,y)={2,(x,y) is on the corner3,(x,y) is on the border4,(x,y) is in the middle

Then the answer is min{f(x1,y1),f(x2,y2)}min{f(x1,y1),f(x2,y2)}.

Without loss of generality, assume that f(x1,y1)≤f(x2,y2)f(x1,y1)≤f(x2,y2). As the method is already given, the answer is at most f(x1,y1)f(x1,y1). Let's prove that the answer is at least f(x1,y1)f(x1,y1).

If (x1,y1)(x1,y1) is on the corner, we can always find two paths from (x1,y1)(x1,y1) to (x2,y2)(x2,y2) without passing the same cell (except (x1,y1)(x1,y1) and (x2,y2)(x2,y2)).

|  |
| --- |
|  |

Similarly, we can always find three or four paths respectively if (x1,y1)(x1,y1) is on the border or in the middle.

|  |
| --- |
|  |

As the paths have no common cell, we need to put an obstacle on each path, so the answer is at least f(x1,y1)f(x1,y1).

In conclusion, the answer is exactly f(x1,y1)f(x1,y1). As we assumed that f(x1,y1)≤f(x2,y2)f(x1,y1)≤f(x2,y2), the answer to the original problem is min{f(x1,y1),f(x2,y2)}min{f(x1,y1),f(x2,y2)}.

Time complexity: O(1)O(1).

Code

#include <bits/stdc++.h>

int T,n,m,x1,x2,y1,y2;

using namespace std;

int f(int x,int y){

    if((x == 1 || x == n) && (y == 1 || y == m)) return 2; *//in the corner*

    if(x == 1 || x == n || y == 1 || y == m) return 3;

    return 4;

}

int main() {

    for(scanf("%d", &T);T;T--) {

        scanf("%d%d%d%d%d%d", &n, &m, &x1, &y1, &x2, &y2);

        printf("%d\n", min(f(x1, y1), f(x2, y2)));

    }

    return 0;

}