ETH Zurich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger
Prof. Dr. Emo Welzl
Prof. Dr. Peter Widmayer

HS 2012
Exercise Sheet ACM Week II

Algorithms Lab

Exercise 2 – *Race Tracks*

Many boring math classes have been spent playing Race Tracks, where two players have to maneuver their cars on a race track drawn on a piece of paper, while their cars can only accelerate by a limited (positive or negative) amount per move.

A variant of Race Tracks involves Hoppers. Hoppers are people on a jump stick who can jump from one square to another, without touching the squares in between (a bit like a knight in chess). Just like the aforementioned cars, they can pick up speed and make bigger hops, but their acceleration per move is limited, and they also have a maximum speed.

Let's be a bit more formal: our variant of Race Tracks is played on a rectangular grid, where each square on the grid is either empty or occupied. While hoppers can fly over any square, they can only land on empty squares. At any point in time, a hopper has a velocity (x, y), where x and y are the speed (in squares) parallel to the grid in x- and y-direction respectively . Thus, a speed of $(2, 1)$ corresponds to a knight jump (as does $(-2, 1)$ and 6 other speeds).

Whenever a hopper stands on a square he can change its speed in each direction separately by -1, 0, or 1. Thus, while having speed $(2, 1)$, the hopper can change to speeds $(1, 0)$, $(1, 1)$, $(1, 2)$, $(2, 0)$, $(2, 1)$, $(2, 2)$, $(3, 0)$, $(3, 1)$ and $(3, 2)$. It is impossible for the hopper to obtain a velocity of 4 in either direction, so the x and y component will **always** stay between -3 and 3 inclusive.

The goal of Hopping Race Tracks is to get from start to finish as quickly as possible (i.e. in the least number of hops), without landing on occupied squares. You are to write a program which, given a rectangular grid, a start point S, and a finish point F, determines the least number of hops in which you can get from S to F. The hopper starts with initial speed $(0, 0)$ and he does not care about his speed when he arrives at F.

Input The first line contains the number of test cases N your program has to process. Each test case consists of a first line containing the width X (where $1 \leq X \leq 30$) and height Y (where $1 \leq Y \leq 30$) of the grid. Next is a line containing four integers separated by blanks, of which the first two indicate the start point (x_1, y_1) and the last two indicate the end point (x_2, y_2) (where $0 \leq x_1, x_2 < X$ and $0 \leq y_1, y_2 < Y$). The third line of each test case contains an integer $P \leq 10$ indicating the number of obstacles in the grid. Finally, the test case consists of P lines, each specifying an obstacle.

Each obstacle consists of four integers x_1, y_1, x_2, y_2 (where $0 \leq x_1 \leq x_2 < X$ and $0 \leq y_1 \leq y_2 < Y$) meaning that all squares (x, y) with $x_1 \leq x \leq x_2$ and $y_1 \leq y \leq y_2$ are occupied.

The start point will never be occupied.

Output The string '`No solution.`' if there is no way the hopper can reach the finish point from the start point without hopping on an occupied square. Otherwise, the text '`Optimal solution takes` N `hops.`', where N is the number of hops needed to get from start to finish point.

Sample input

```
2
5 5
4 0 4 4
1
1 2 4 3
3 3
0 0 2 2
2
1 0 1 2
0 1 2 1
```

Sample output

```
Optimal solution takes 7 hops.
No solution.
```