



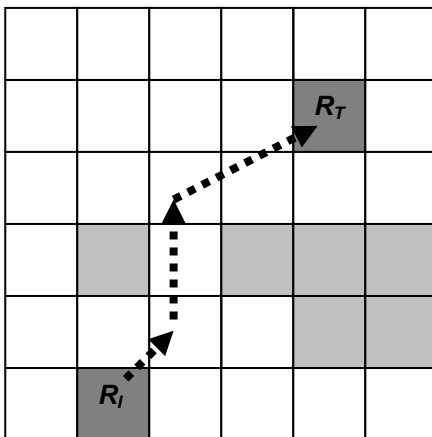
## PROBLEM 8 – ROBOTS

A popular Acopian strategy game consists of moving a robot on a terrain with fixed obstacles. The objective is to move the robot from a given initial position to a given target position.

In our case, the terrain is defined by a planar rectangular map. The robot and the obstacles have rectangular shapes, with corners on integer coordinates. All rectangles are proper, i.e., each side is at least 1 unit long. All obstacles and robot positions are completely within the rectangle bounding the map. The initial and target positions may overlap each other, but do not overlap existing obstacles. Obstacles themselves may overlap. Here, boundaries are not considered as belonging to the objects – thus, the robot, the obstacles and the map may share some of their boundaries.

The robot has a fixed shape, cannot rotate, but can move by translations in any direction, within the map borders, as long as it avoids collisions with the obstacles and with the map limits. The robot is allowed to slide alongside the borders of the obstacles or of the map.

The following diagram gives an example of this game, where  $R_I$  is the initial robot position,  $R_T$  the target position, the other shaded areas are obstacles, and the dotted line suggests the shortest robot path.



Your task is to write a program that determines the length of a shortest robot path on such a map with obstacles, if such a path exists.

For this problem, where necessary, use `double` floating point numbers, floating point additions, the `sqrt()` function, and the output display facilities, such as `printf()`, provided by your system. The final results must be displayed accurate to exactly 2 decimals. For example, in the above diagram scenario, report that the length of the shortest path is 5.65:

$$\text{Sqrt}(2) + 2 + \text{Sqrt}(5) = 1.4142... + 2.0 + 2.2360... \approx 5.65$$



## INPUT FORMAT

Input for this problem consists of a sequence of one or more scenarios. Each scenario is described by several lines.

- The first line defines, in order, a numeric scenario label,  $l$ ,  $1 \leq l \leq 100$ , the number of obstacles,  $m$ ,  $1 \leq m \leq 10$ , and the  $x$  and  $y$  coordinates of the top-right corner of the rectangle bounding the map,  $1 \leq x, y \leq 1000$ , separated by single spaces.
- The second line defines the initial and target robot positions, specifically, in order, the initial  $x$  and  $y$  coordinates of its bottom-left corner, the initial  $x$  and  $y$  coordinates of its top-right corner, and the target  $x$  and  $y$  coordinates of its bottom-left corner, separated by single spaces.
- The next lines contain  $4 \times m$  integers, separated by single spaces or newlines. These integers define the  $m$  obstacles, by giving in order, for each obstacle, the  $x$  and  $y$  coordinates of its bottom-left corner, and the  $x$  and  $y$  coordinates of its top-right corner.
- All coordinates are bounded within the map rectangle, including the inferred  $x$  and  $y$  coordinates of the robot's top-right corner target position.

The input will be terminated by a line consisting of four zeros (0), separated by single spaces. This line should not be processed.

## OUTPUT FORMAT

Output will be a sequence of lines, one for each input scenario. Each line will start with the scenario label, followed by the length of the shortest path, accurate to exactly 2 decimals, separated by a single space. If no such path exists, print the character minus (-) in place of the path length.

## SAMPLE INPUT:

```
10 3 6 6
1 0 2 1 4 4
1 2 2 3 3 2 5 3 4 1 6 3
11 4 6 6
1 0 2 1 4 4
1 2 2 3 3 2 5 3 4 1 6 3 3
4 4 5
12 3 6 6
1 0 3 1 4 4
1 2 2 3 3 2 5 3 4 1 6 3
0 0 0 0
```

## SAMPLE OUTPUT:

```
10 5.65
11 6.41
12 -
```