The Twentieth Annual ACM International Collegiate

Programming Contest Finals

sponsored by

Microsoft®

Problem

Cutting Corners

Input file: corner.in

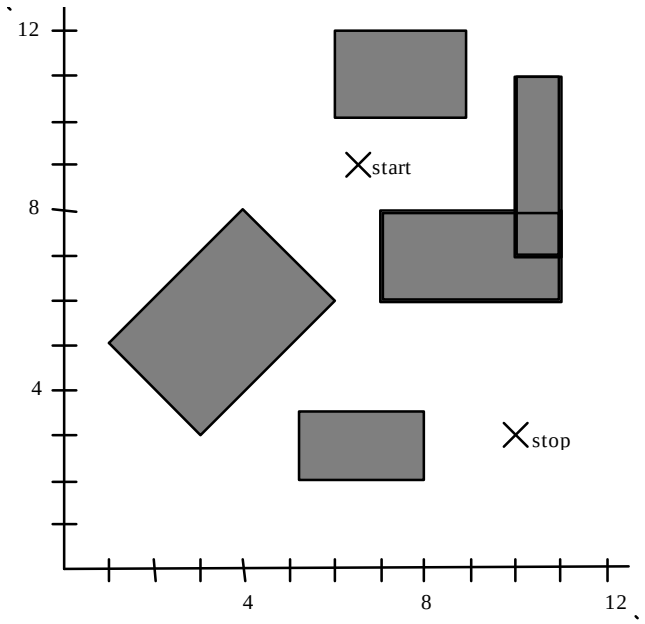
Bicycle messengers who deliver documents and small items to businesses have long been part of the guerrilla transportation services in several major U.S. cities. The cyclists of Boston are a rare breed of riders. They are notorious for their speed, their disrespect for one-way streets and traffic signals, and their brazen disregard for cars, taxis, buses, and pedestrians.

Bicycle messenger services are very competitive. Billy's Bicycle Messenger Service is no exception. To boost its competitive edge and to determine its actual expenses, BBMS is developing a new scheme for pricing deliveries that depends on the shortest route messengers can travel. You are to write a program to help BBMS determine the distances for these routes.

The following assumptions help simplify your task:

- Messengers can ride their bicycles anywhere at ground level except inside buildings.
- Ground floors of irregularly shaped buildings are modelled by the union of the interiors of rectangles. By agreement any intersecting rectangles share interior space and are part of the same building.
- The defining rectangles for two separate buildings never touch, although they can be quite close. (Bicycle messengers— skinny to a fault—can travel between any two buildings. They can cut the sharpest corners and run their skinny tires right down the perimeters of the buildings.)
- For a given route, the starting and stopping points are never inside buildings.
- Their is always some route from a starting point to a stopping point.

Your program must be able to process several scenarios. Each scenario is a bird's-eye snapshot showing the buildings and the starting and ending points for a delivery route. The picture below is a typical scenario.



The input file represents several scenarios. Input for each scenario consists of lines as follows:

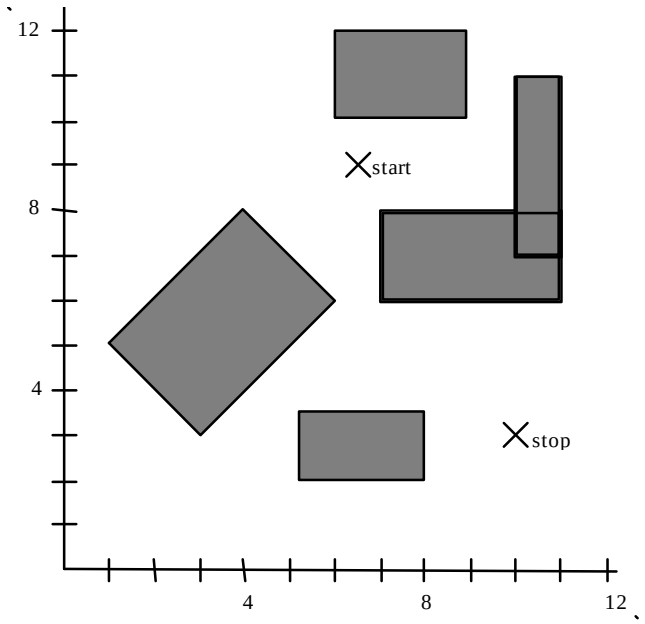
First line:	n	The number of rectangles describing the buildings in the scenario. There are from 0 to 100 such rectangles inclusive.
Second line:	$x_1\ y_1\ x_2\ y_2$	The x- and y-coordinates of the starting and stopping points of the route.
Remaining n lines:	$x_1\ y_1\ \ x_2\ y_2\ \ x_3\ y_3$	The x- and y-coordinates of three vertices of a rectangle.

The x- and y-coordinates of all input data are real numbers between 0 and 1000 inclusive. Successive coordinates on a line are separated by one or more blanks. The integer -1 follows the data of the last scenario.

To avoid problems with real precision, the input data set restricts all coordinates to be between 0 and 1000 inclusive. The interior enclosed by any two intersecting rectangles will be at least large enough to contain a square of .01 unit on a side. In addition, two buildings that do not intersect will be at least .01 unit apart.

Output should number each scenario (Scenario #1, Scenario #2, etc.) and give the distance of the shortest route from starting to stopping point as illustrated in the Sample Output below. The distance should be written with two digits to the right of the decimal point. Output for successive scenarios should be separated by blank lines.

The following scenario is described by the Sample Input below.



Sample Input

```
5
6.5   9     10  3
1     5     3   3     6  6
5.25  2     8   2     8  3.5
6     10    6   12    9  12
7     6     11  6     11 8
10    7     11  7     11 11
-1
```

Sample Output

```
Scenario #1
   route distance: 7.28
```