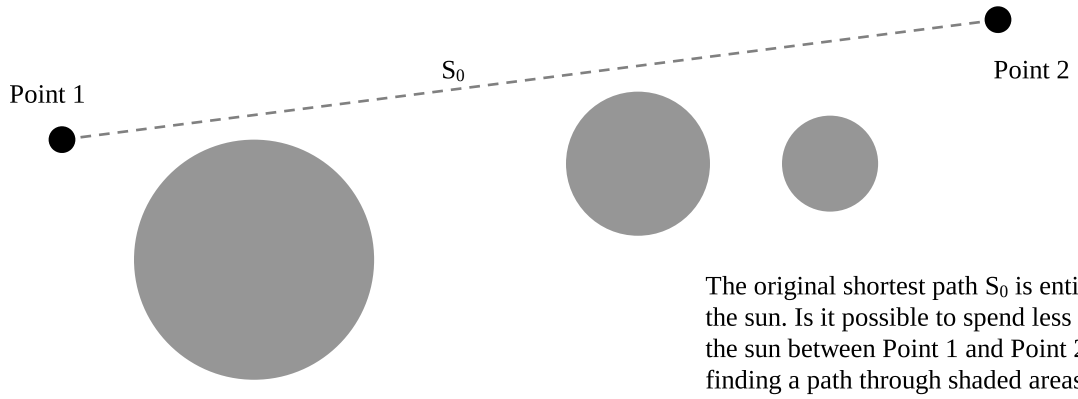
UCF "Practice" Local Contest — Aug 23, 2014

Walking in the Sun

filename: `sunwalk`
(Difficulty Level: `Medium`)

You have calculated all the shortest distances between points on campus, but have found that this strategy is not optimal in the summer. Some of these shortest paths are in the sun, and it's far more annoying to be sweating than to walk a few extra steps in the shade. Your goal is to recalculate shortest distances in the sun on campus, given information about where shade is located.



The original shortest path S_0 is entirely in the sun. Is it possible to spend less time in the sun between Point 1 and Point 2 by finding a path through shaded areas?

The Problem:

We simplify the problem by specifying all locations on campus by 2-D Cartesian coordinates, and by specifying all areas of shade as circular areas with a given center and radius. We also assume that any straight line between two Cartesian points can be walked, i.e., there are no objects blocking any possible straight line paths. Instead of calculating the shortest distance between locations on campus, your goal will be to calculate the least amount of walking that needs to be done in the sun to get between those two points.

The Input:

There will be multiple test cases in the input file. The first input line contains a positive integer, indicating the number of test cases.

The first line of each test case, s *$(0 \le s < 50)$*, contains the number of locations of shade on campus for that test case. Each of the next s lines contains the x-coordinate, y-coordinate, and radius, respectively, of a shade location, separated by spaces.

The next line of each test case, q $(0 < q < 100)$, contains the number of distance queries for the test case. Each of the next q lines will contain four numbers representing two points on campus, in the form x_1 y_1 x_2 y_2, where (x_1,y_1) are the coordinates of the first point, and (x_2,y_2) are the coordinates of the second point.

All x-coordinates, y-coordinates, and radii will be real numbers between -1000 and 1000, inclusive.

The Output:

At the beginning of each test case, output "`Campus #`*`c`*`:`", where *c* is the test case number (starting from 1) on the first line.

For the subsequent q lines, begin the output with the header " `Path #`*`p`*`:`", where p is the distance query number (starting from 1) for that case. Follow each of these headers with the statement of the form: "`Shortest sun distance is D.`", where D is the desired shortest sun distance rounded to one decimal place. To clarify "rounded to one decimal place": the output for 1.74 should be 1.7, for 1.75 should be 1.8, and for 1.76 should be 1.8.

Leave a blank line after the output for each test case. Follow the format illustrated in Sample Output.

Sample Input:

```
2
3
5.2 3.3 4.7
-8.8 -6.1 3.1
18.5 6.1 2.2
6
1.1 20.2 6.1 18.1
1.1 20.2 3.3 -2.5
0.4 -2.7 3.3 -2.5
6.1 18.1 -5.5 -9.2
3.3 -2.5 0.4 -2.7
1.1 20.2 0.4 -2.7
1
0.0 0.0 20.0
1
3.1 2.2 7.7 8.1
```

Sample Output:

```
Campus #1:
  Path #1: Shortest sun distance is 5.4.
  Path #2: Shortest sun distance is 14.1.
  Path #3: Shortest sun distance is 2.9.
  Path #4: Shortest sun distance is 20.6.
  Path #5: Shortest sun distance is 2.9.
  Path #6: Shortest sun distance is 15.7.

Campus #2:
  Path #1: Shortest sun distance is 0.0.
```