Problem 152: Around the Town

Difficulty: Hard

Author: Anthony Vardaro, Dallas, Texas, United States

Originally Published: Code Quest 2021

Problem Background

Public transportation systems often consist of a variety of transportation methods designed to make it easier for people or groups to move around a city. Whatever the method of travel, any public transportation system will have a finite number of "stops," where patrons gather to wait for the arrival of a bus or train. It's crucial that patrons should not have to walk too far to reach a stop, and so stops are often strategically placed near tourist attractions, residential areas, and other high population areas where a stop would be useful.

Lockheed Martin has been contracted to help a city design their first public transportation system. They are able to provide your team with the locations of several major landmarks around the city, but need your help making sure that their proposed bus stops are in the best possible locations.

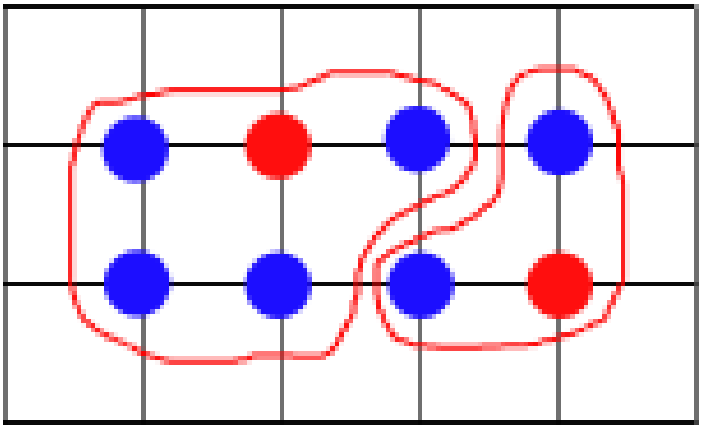
Problem Description

The city council has a rough idea of where they'd like bus stops to be located, but are concerned that they may not be in the best possible location. Your lead engineer suggests using a type of heuristic algorithm called "k-means" to optimize the stop locations.

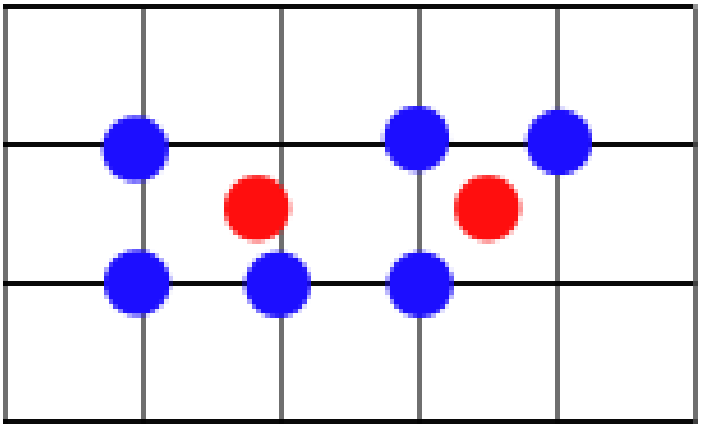
The k-means algorithm works by collecting a series of points into clusters. Each cluster has a "centroid;" a point which is the minimum possible distance from each data point within its cluster. Normally, the algorithm starts by choosing a set of centroids at random, then works to optimize them; in this case, we'll start with the city council's proposed bus stops as our starting set of centroids.

The algorithm works over several iterations; in each iteration, points (landmarks) are sorted into clusters by identifying which candidate centroid (bus stop) is closest to them. (In the event of a tie, landmarks should prefer the bus stop with the lowest-value X coordinate; if still tied, the bus stop with the lowest-value Y coordinate.) Each cluster then identifies a new centroid by averaging the coordinates of all points within the cluster. The clusters are then disbanded and the process repeats, stopping when none of the centroids change position following an iteration.

For example, let's look at the coordinate grid below. The blue points represent landmarks, and the red points represent the city council's proposed bus stops.



The wavy red lines indicate which landmarks have been grouped into each cluster; again, this is done by identifying the closest candidate bus stop to each landmark. With the clusters formed, we can average the coordinates of each landmark within the cluster to identify the new bus stop locations. The left-hand cluster's bus stop would move down and slightly to the left as a result; the right-hand cluster's bus stop would move directly between its two landmarks, as shown below.



The process then repeats, forming new clusters and moving the bus stops further. You'll notice that the top center landmark is now closer to the right-hand bus stop than it is the left-hand one; it will switch to the right-hand cluster as a result. These new clusters are used to identify new bus stop locations, until we complete an iteration without moving the locations at all.

The following formula for determining the straight-line distance between two points will be helpful as you solve this problem:

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing two positive integers, separated by spaces:
  - L, the number of landmarks requiring access to public transportation

  - S, the number of candidate bus stops identified by the city council
- L lines containing two integers, representing the X and Y coordinates (respectively) of a landmark in the city
- S lines containing two integers, representing the X and Y coordinates (respectively) of a candidate bus stop

```
1
6 2
0 1
2 1
3 1
0 0
1 0
2 0
1 1
3 0
```

Sample Output

For each test case, your program must print the X and Y coordinates of the optimum location of each bus stop. Round coordinates to one decimal place, include any trailing zeroes, and separate them by spaces; each bus stop should be printed on a separate line, in the order provided.

```
0.3 0.3
2.3 0.7
```