The University of Newcastle
School of Electrical Engineering and Computer Science

COMP2230/6360 Algorithms

Assignment

Total mark: 100
Due 24 October 2020
11:59pm, via Blackboard

You have been contracted by the Department of Fire and Emergency Services to develop an algorithm they can use to place temporary fire stations in the case of extensive bushfires. They would provide you with a map of hotspots affected by a bushfire and a number of temporary fire stations they are able to support at that time.

The temporary fire stations will each be responsible several hotspots, with no double-handling of hotspots (i.e., each hotspot will be handled by only one temporary fire station).

The fire department have two important considerations:

- On one hand, they want to ensure that hotspots handled by different fire stations are as far from each other as possible. This is important as fire can easily spread and in general it is not possible to fully synchronise the efforts of all neighbouring fire stations.
- On the other hand, it is also important that hotspots handled by the same fire station are as close together as possible so that fire brigade can move from one hotspot to another.

Therefore your program should suggest the number of fire stations to maximize the ratio of distances between hotspots handled by different fire stations and those handled by the same station.

Your task is to solve this problem with an efficient algorithm for solving a clustering problem.

CLUSTERING BACKGROUND

The clustering problem is defined as follows:

Given a set of items, $a_1, a_2, \ldots, a_n$*, and a set of distances between those items,*
$$d_{ij} = d(a_i, a_j) \text{ where } i, j \in [1, n]$$
then find the optimal clustering of the points.

We explain/define the important concepts as follows.

- Distance between two items is a measure of dissimilarity between the two items. If the items are points in a two-dimensional plane, distance can be defined as the Euclidean distance.

- Inter-clustering distance (InterCD) is the minimum distance between any two items belonging to different clusters. Formally:
$$\text{InterCD} = \min\{d(a_i, a_j) \mid i, j \in [1, n] \text{ and } a_i \text{ is in a different cluster to } a_j\}$$

- Similarly, intra-clustering distance (IntraCD) is the maximum distance between any two items belonging to the same cluster. Formally:

$$\text{IntraCD} = \max\{d(a_i, a_j) \mid i, j \in [1, n] \text{ and } a_i \text{ is in the same cluster as } a_j\}$$

- A clustering is optimal if it maximizes the inter-clustering distance for the given number k of clusters. (Note that, in general, there are many different ways to define an optimal clustering, and we are using this particular one).

- The centroid of the cluster is the "average" point in the cluster. The x-coordinate of the centroid is the average of the x-coordinates of all the points in the cluster; similarly, the y-coordinate of the centroid is the average of the y-coordinates of all the points in the cluster.

TASKS

Task 1 – Individual and pairs assignment: Use Kruskal's algorithm for finding a minimum spanning tree to find an optimal clustering (clustering that maximizes the inter-clustering distance) of the given set of *n* points in a two-dimensional plane, for the given number *k* of clusters.

Task 2 – Pairs assignment only: Among the optimal clusterings obtained by Kruskal's algorithm for different values of k (where $2 \leq k \leq n-1$), find the one that maximizes the ratio $\frac{\text{InterCD}}{\text{IntraCD}}$ between the inter-clustering distance and intra-clustering distance.

INPUT

1. Your program must be executable from the command line. It must accept input as command line arguments (a.k.a., parameters), and write output to standard output (e.g., `cout` in C++, `System.out` in Java). Errors should be printed to standard error (e.g., `cerr` in C++, `System.error` in Java).

2. Your program executable (or your main Java class) must be called `kcluster`. The first command line arguments it accepts must be a path to the input file, and the second must be the number of temporary fire stations.

   For example, to run the program using a file named "input.txt" in the current folder, and 2 temporary fire stations, we would run:
   ```
   kcluster ./input.txt 2
   ```
   or
   ```
   java kcluster ./input.txt 2
   ```

3. If you are working in pairs, then the program must allow for the 2nd argument to be the word "`auto`". In this case the program will automatically calculate number of clusters so as to maximize the inter-clustering to intra-clustering distance ratio.

4. The input file will be a comma separated value file, with the first column being the ID of the hotspot, the 2nd column is the x-coordinate of the hotspot, and the 3rd column is the y-coordinate of the hotspot. The ID will be a unique integer, and the x- and y-coordinates will be floating point numbers.

For example, an input file might look like:

```
1,1.0,1.0
2,2.0,2.0
3,3.0,5.0
4,7.0,8.0
5,8.0,7.0
```

5. The program should handle obvious error conditions (e.g., incorrectly formatted or empty input files, negative numbers of temporary fire stations, etc) by printing an error message to standard error and immediately stopping execution.

OUTPUT

The output must look like the following example.

INDIVIDUAL ASSIGNMENT:

Suppose the program is run for an input file named "input.txt", requesting two stations by running `java kcluster ./input.txt 2` then the output would be:

```
Hello and welcome to Kruskal's Clustering!

The weighted graph of hotspots:

 0     1.41  4.47  9.22  9.22
 1.41  0     3.16  7.81  7.81
 4.47  3.16  0     5     5.39
 9.22  7.81  5     0     1.41
 9.22  7.81  5.39  1.41  0


There are 5 hotspots.
You have requested 2 temporary fire stations.

Station 1:
Coordinates: (2.00, 2.67)
Hotspots: {1,2,3}

Station 2:
Coordinates: (7.50, 7.50)
Hotspots: {4,5}

Inter-clustering distance: 5.00

Thank you for using Kruskal's Clustering. Bye.
```

PAIRS ASSIGNMENT:

The assignment should look the same as the individual assignment if the number of requested temporary fire stations is a number.

In addition: if the number of fire stations is given as `auto`, then the output should be as below. For example, suppose the program is run for an input file named "input.txt", requesting automatic station calculation by running `java kcluster ./input.txt auto` then the output would be:

```
There are 5 hotspots.
Automatically calculating number of temporary fire stations.

The weighted graph of hotspots:

 0     1.41  4.47  9.22  9.22
 1.41  0     3.16  7.81  7.81
 4.47  3.16  0     5     5.39
 9.22  7.81  5     0     1.41
 9.22  7.81  5.39  1.41  0


Number of stations: 3

Station 1:
Coordinates: (1.50, 1.50)
Hotspots: {1,2}

Station 2:
Coordinates: (3.00,5.00)
Hotspots: {3}

Station 3:
Coordinates: (7.50, 7.50)
Hotspots: {4,5}

InterCD/IntraCD = 2.24

Thank you for using Kruskal's Clustering. Bye.
```

ASSESSMENT CRITERIA

Your code must compile and run. A program that does not compile or run can only earn marks from the "Graph construction" and "Kruskal's Algorithm" categories (see below). The marker will *not* modify your code to make it work.

Marks awarded for the assignments depend on whether the assignment is an individual or pairs assignment, as follows.

INDIVIDUAL ASSIGNMENT

1. Graph construction: ***12 marks***.
   1.1. *(10 marks)* for the correct algorithm and graph
   1.2. *(2 marks)* for the well-commented and clear implementation

2. Kruskal's Algorithm: ***45 marks***.
   2.1. *(40 marks)* for the correct algorithm
   2.2. *(5 marks)* for a well-commented and clear implementation

3. Positioning of fire stations: ***15 marks***.

4. Calculation of inter-clustering distance: ***10 marks***.

5. Input and output as specified: ***18 marks***.
   5.1. *(4 marks)* Program accepts command line parameters.
   5.2. *(4 marks)* Program outputs to standard output and writes errors to standard error.
   5.3. *(10 marks)* Output formatted as requested.

PAIRS ASSIGNMENT

1. Graph construction: ***7 marks***.
   1.1. *(5 marks)* for the correct algorithm and graph
   1.2. *(2 marks)* for the well-commented and clear implementation

2. Kruskal's Algorithm: ***35 marks***.
   2.1. *(30 marks)* for the correct algorithm
   2.2. *(5 marks)* for a well-commented and clear implementation

3. Positioning of emergency stations: ***10 marks***.

4. Calculation of inter-clustering distance: ***10 marks***.

5. Finding the number of clusters to maximize $\frac{\text{InterCD}}{\text{IntraCD}}$: ***20 marks***.

6. Input and output as specified: ***18 marks***.
   6.1. *(4 marks)* Program accepts command line parameters correctly.
   6.2. *(4 marks)* Program outputs to standard output and writes errors to standard error.
   6.3. *(10 marks)* Output formatted as requested.