CMPE 224
Fall 2021
Programming Homework 4

This assignment is due by 23:59 on Sunday, 19 December 2021.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE224-HW4-OfficeHour1: 10 Dec, 06:00-08:00 PM, Zoom ID: 968 7701 3315
   - CMPE224-HW4-OfficeHour2: 15 Dec, 06:00-08:00 PM, Zoom ID: 930 3298 7261

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK(s)

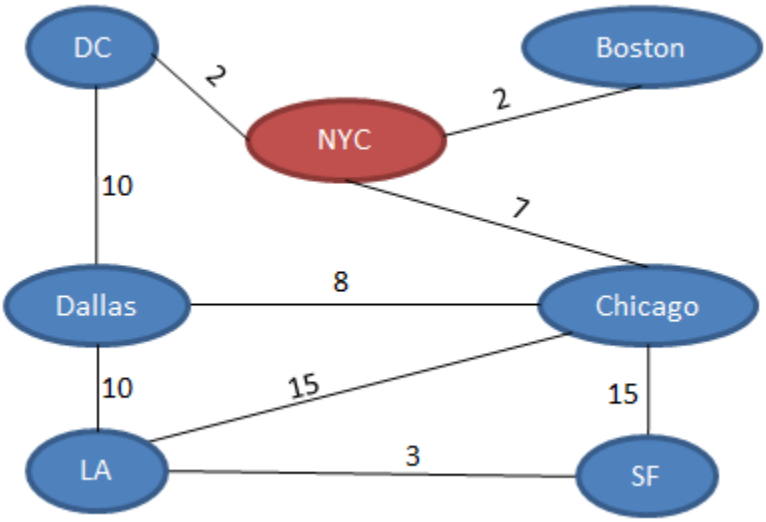
Task 1 (30 points)

In this part of the homework, you are to build high-speed railroad networks in the USA, so that traveling from NYC to DC in the United States becomes only 2 hours! To achieve that goal, assuming all cities are connected, you need to plan a railroad network and calculate the time between the cities, for which we need to solve the minimum spanning tree (MST) problem (by using Prim's or Kruskal's algorithm). To make things easier we only care about the distances from a home city to all other cities. For this task, you must use your own implementation of graphs by taking inspiration from your textbook. You are not allowed to use any external library or .jar file.

Sample input (`sample_input1.txt`) starts with a line of cities separated by blanks; the first city on that line is considered as the home (root node) city where you would grow your MST from. Each of the following lines is triple, where two cities and their distance (hours on planned high-speed rail). The distance is symmetric.

```
NYC Chicago LA Boston DC SF Dallas
NYC Boston 2
NYC DC 2
NYC Chicago 7
Chicago Dallas 8
DC Dallas 10
Chicago SF 15
Chicago LA 15
SF LA 3
LA Dallas 10
```

Here is the visualization according to sample input:



For the `sample_input1.txt` file, the sample output is given below. The output starts with a line of the total of the MST, followed by the lines is triple, where two cities and their distance. Please note that the output should follow a level-order traversal of the MST and within each level, use lexicographical order.

```
> java MST sample_input1.txt
32
NYC Boston 2
NYC DC 2
NYC Chicago 7
Chicago Dallas 8
Dallas LA 10
LA SF 3
```

Task 2 (70 points)

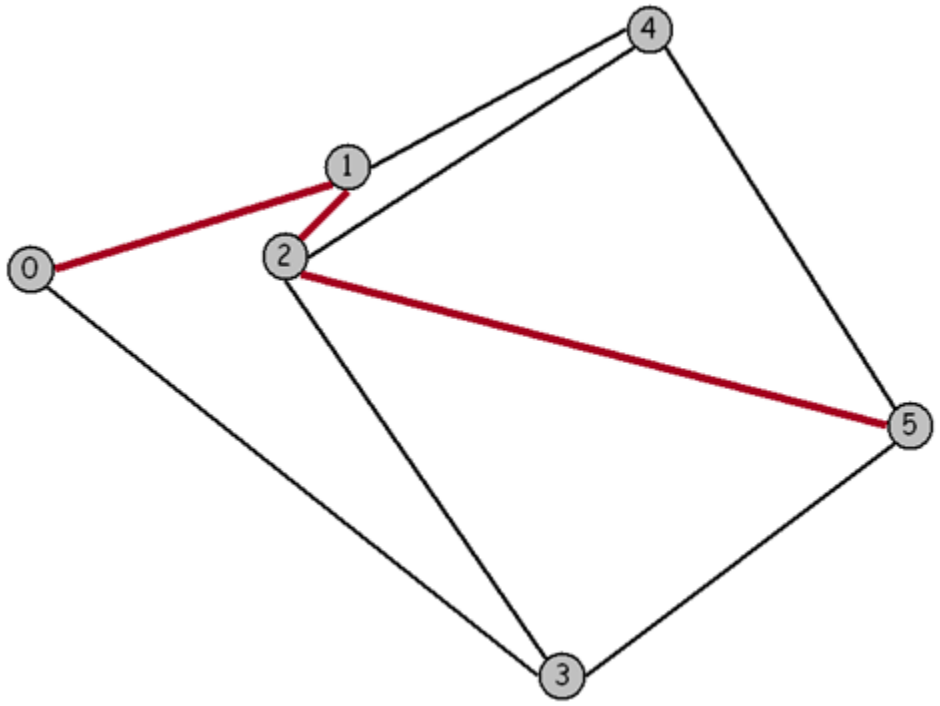
[40 points] In the second task of the homework, you will be working with *maps*, or graphs whose vertices are points in the plane and are connected by edges whose weights are **Euclidean distances**. Think of the vertices as cities and the edges as roads connecting them. The Euclidean distance formula, as its name suggests, gives the distance between two points (or) the straight line distance. Let us assume that (`x1`, `y1`) and (`x2`, `y2`) are the two points in a two-dimensional plane. Here is the Euclidean distance (`d`) formula:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

To represent a map in a file, we list the number of vertices and edges, then list the vertices (index followed by its `x` and `y` coordinates), then list the edges (pairs of vertices). For instance, an example input file (`sample_input2.txt`) is given below. This file contains 6 cities and 9 edges, as indicated its first line. After this first line, there exists information for each city. For example, line 2 of this file indicates that İstanbul; with an id of 0 has the coordinates (1000, 2400) on the map.

```
6 9
0 1000 2400 İstanbul
1 2800 3000 Konya
2 2400 2500 Ankara
3 4000  0   Hatay
4 4500 3800 Trabzon
5 6000 1500 Ağrı
0 1
0 3
1 2
1 4
2 4
2 3
2 5
3 5
4 5
```

Here is the visualization according to `sample_input2.txt` file:



After representing given input file as a graph, you should create a program that computes the shortest paths from a given source city to another target city. For this task, you must use your own implementation of graphs by taking inspiration from your textbook. You are not allowed to use any external library or .jar file. The program should take as input two cities, and output the shortest path (by using Dijkstra algorithm). For the example input file (`sample_input2.txt`) given above, the sample outputs are given such that:

```
> java ShortestPath sample_input2.txt Konya Trabzon
2 cities to be visited:
Konya
Trabzon
Distance: 1878 km

> java ShortestPath sample_input2.txt İstanbul Ağrı
4 cities to be visited:
İstanbul
Konya
Ankara
Ağrı
Distance: 6273 km
```

[30 points] You need also to run empirical studies to evaluate the computational complexity of your program on large map data sets. Two sample large datasets (named as `turkeymap.txt` and `usa.txt`) are available in the course Moodle (LMS) site. You can also download other large map datasets from the web. You can measure time quantity in milliseconds. For this purpose, you should use the `System.currentTimeMillis()` method, which returns a long that contains the current time (in milliseconds). In order to get the most accurate runtimes possible, you may want to close other programs that might be taking a lot of processing power and memory and affecting your program runtimes.

Write additional **max 2-pages into your task's PDF report** explaining your results and any conclusions that might be drawn. You are also expected to add visualization (table, plot, graph etc.) elements showing the time complexity of your program with different size of inputs. You should answer questions such as (but not limited to):

- How does your program scale with changing input size?
- Does it have the expected computational behavior as explained in the textbook?
- Where is the performance bottleneck of your program?
- What can be done to speed it up?

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You should test your Java source files on (if) available Moodle VPL environment to ensure your code solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input. You should pass that task's VPL test case successfully.
- A single **maximum-4 pages** PDF report document that explains your own answers for both tasks in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).
- For both tasks, only code or report submission will not be graded. In other words, you should submit both correct code solution and its related report for that task in order to be graded.

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. Please note that if you do not have correct code solution for the related task, you should not report about that since it will not be graded. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%30): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a *structure chart* that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%40): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the *pseudocode* algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%15): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%10): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Sunday, December 19th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-------------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-----------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
```

```
// Postcondition: The value of the variable is set.
//-------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TAs, Bedrettin Çetinkaya, Deniz Merve Gündüz and İbrahim İleri. Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructors Tolga Kurtuluş Çapın and Ulaş Güleç for help.