Data Structures and Algorithms

Assignment Deadline: Week 13

Semester 1, 2023

Department of Computing, Curtin University

Introduction

In practicals, you have implemented and learned about a number of algorithms and ADTs and will be implementing more of these in the remaining practicals. In this assignment, you will be making use of this knowledge to implement a system to explore and compare a variety of ADT implementations. Feel free to re-use the generic ADTs from your practicals. However, remember to self-cite; if you submit work that you have already submitted for a previous assessment (in this unit or any other), you must specifically state this. Do not use the Java/Python implementations of ADTs – if in doubt, ask.

Problem Description

In this problem, you need to monitor bushfires using unmanned aerial vehicles (UAVs). The UAVs will fly over an area of interest and collect data on temperature, humidity, wind speed, and other relevant factors. The data collected by the UAVs needs to be processed to identify areas where there is a high risk of bushfires and areas that need attention.

Task 1: Creating a Graph

To represent the area of interest, you will need to create a graph. The vertices of the graph will represent the locations in the area, and the edges will represent the distances between the locations. You may use adjacency list to represent the graph.

Input

You have to use the **location.txt file** as input to build the graph.

The first line specifies that the graph has 10 vertices and 15 edges. Each subsequent line represents an edge and first three values in each line: the starting vertex, the ending vertex (both represented by alphabetic characters), and the weight of the edge.

Output

Build the graph and display the adjacency list of the graph.

Task 2: Implementing BFS and DFS

To traverse the graph and gather information about the risk of bushfires, we will use BFS and DFS algorithms. BFS will help to find the shortest path between two locations, and DFS will help to explore the entire graph. For each location, the UAV collects data on temperature (in 25-48 degrees Celsius), humidity (in 15-60 percentage) and wind speed (in 30-100 km/h).

You have to use the **UAVdata.txt file** as input to keep track of the associated data of each location along with **location.txt file**. Each location is identified by a single letter identifier (e.g. "A" for the first location), and the temperature, humidity, and wind speed data are separated by spaces.

Input

Any two locations and the entire graph.

Output

The shortest path between two locations, along with associated data.

Task 3: Insert, Delete, and Searching Operations

To make your program scalable, you will also need to implement insert, delete, and search operations on the graph to add or remove locations and to search for specific locations. You must follow an interactive menu to perform every operation anytime.

Task 4: Hashing the Data

To efficiently store and retrieve the data gathered by the UAVs, you will use a hash table. The hash table will allow us to quickly look up the data associated with a particular location.

You then need to compare and discuss the efficiency of using the hashtable than searching through a list or array.

Task 5: Using a Heap

Finally, to keep track of the areas with the highest risk of bushfires, you will use a heap data structure. The heap will allow us to efficiently extract the maximum risk from a set of risks. To keep track of areas with the highest risk of bushfires using a heap, you can store the risk values of different areas in a heap. Each node in the heap represents an area with its corresponding risk value.

When new data is gathered by the UAVs, you can update the corresponding risk value in the heap. If the new risk value is higher than the current maximum value, we can extract the current maximum value and insert the new value into the heap. This ensures that the heap always contains the areas with the highest risk values.

You have to use some thresholds to mark the risk levels. You may use the following thresholds:

Temperate

25-32 low

33-40 medium

>40 high

Humidity

>50 low

31-50 medium

<30 high

Wind Speed

<40 low

41-55 medium

>55 high

Task 6: Providing an Itinerary

To optimize the UAVs' flight paths, you will provide an itinerary based on the distances and time between the locations with high risk of bushfires. This itinerary will ensure that the UAVs cover as much of the area as possible while minimizing their energy consumption. Here, you have the opportunity to further research than your syllabus. You can use other algorithms than using BFS and DFS.

Task 7: Writing and Testing the Program

Using the above techniques and data structures, you will write a program to process the data collected by the UAVs and identify areas where there is a high risk of bushfires and areas that need attention. The program should take as input the data collected by each UAV and output the areas of high risk and the itinerary for the UAVs. You should use an interactive menu and check for errors and exceptions.

To ensure that your program works correctly, you will test it using different sets of data. You should test it with small and large sets of data to ensure it can handle various inputs. You should also use the given input files to test your program.

Submission

Submit electronically via Blackboard.

You should submit a single file, which should be zipped (.zip) or tarred (.tar.gz). Check that you can decompress it on the lab computers. These are also the computers on which your work will be tested, so make sure that your work runs there. The file must be named DSA_Assignment_<id> where the <id> is replaced by your student id. There should be no spaces in the file name; use underscores as shown.

The file must contain the following:

• Your **code**. This means all .java/.py files needed to run your program. Do include code provided to you as part of the assignment if that is required to run your program.

• **README file** including short descriptions of all files and dependencies, and information on how to run the program.

• Your **unit test harnesses**. One of the easiest ways for us to be sure that your code works is to make sure that you've tested it properly. A test harness for class X should be called UnitTestX.

• **Test data** to validate the correctness of your program.

• **A report** detailing the design of your program, UML diagram, the data structures and algorithms used, description of classes, the testing methodology and results. You may also reflect on your implementation and how it might be improved. What further investigations and/or extensions could be added?

• A **signed and dated cover sheet**. These are available from Blackboard with the assignment specification. You can sign a hard copy and scan it in or you can fill in a soft copy and digitally sign it. • Java Students: o Do not include .class files or anything else that we do not need. We will recompile .java files to ensure that what we're testing is what we're reading. We will use javac *.java to compile your files and run the unit tests by their expected names. Make sure that your file contains what is required. It is your responsibility to make sure that your submission is complete and correct.

Marking

Marks will be awarded to your submission as follows:

- [20] Implementation
- [10] Demonstration
- [10] Project Report

Late Submission

If you need extra time and have a valid reason, you should apply for an extension through Oasis. One week time is allowed for the extension. Late submissions will incur a daily deduction.

Clarifications and Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced via Blackboard. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks.