Software Fundamentals

Assignment 3 - Minimum Spanning Tree

Due: October 29, 2025
Mark: 10

Problem statement
This assignment involves implementing the Prim and Kruskal algorithms for finding a Minimum Spanning Tree (MST) of an undirected, weighted graph.

Procedure
Here are the steps you should follow for this assignment:

- Prim's algorithm: Start with the pseudo code given in the slides Our aim is to implement this, in a programming language of your choice. Rather than just taking a cost matrix as input and printing a set of edges as output, you need to take as input and return as output instances of a GRAPH data structure. Here is how you should proceed (please follow these steps carefully): '
    - First of all, you need to implement the GRAPH Abstract Data Type (ADT) as an actual data structure which allows for weighted, undirected graphs (you are NOT permitted to use existing implementations or libraries). Remember, to fully describe a data structure, you need to specify what your cells (atomic data variables) consist of, what the aggregation mechanism over those cells is, and what associated operations or functions are available to manipulate the data structure. [1]
    - Implement a PRIORITYQUEUE data structure. Note that you will need a DECREASE-KEY operation, in addition to the regular INSERT and DELETEMIN operations. [2]
    - Now implement the Prim's algorithm using your two data structures as above. The PRIORITYQUEUE data structure should be used in order to keep track of the lowest-cost edge from the current set U to each vertex outside U. [2]

  The key point to keep in mind is that your overall implementation needs to be modular: separate GRAPH and PRIORITYQUEUE data structures each with a defined set of functions/operations, and the main program (for Prim's algorithm) must interact with the data structure only through these functions.
- Kruskal's algorithm: Start with the pseudocode given in given in the slides. Here is how you should proceed (please follow these steps carefully):
    - Implement a Union-Find data structure. There are multiple ways of implementing it, you may pick one as you prefer but should aim to minimise time complexity. [2]
    - Using all required data structures you have developed as above, implement Kruskal's algorithm. Like Prim's algorithm, your input and output should be in the form of an instance of your GRAPH data structure. [2]
- Write a top-level program for creating a GRAPH instance and running both Prim's and Kruskal's algorithms on it, and printing out the respective MSTs returned, along with the total cost and runtime (in milliseconds) for each one. The input to this top-level program should be an $n \times n$ cost matrix for a graph, n being the number of vertices (assume the diagonal elements will always be 0, and elements corresponding to unconnected vertices will be set to -1); this can be read in from a text file. [1]

- A sample input file which corresponds to the graph depicted in Fig. 7.4 of AHU is as follows:

```
0 6 1 5 -1 -1
6 0 5 -1 3 -1
1 5 0 5 6 4
5 -1 5 0 -1 2
-1 3 6 -1 0 6
-1 -1 4 2 6 0
```

Your code should be well-commented and it should be easy for anyone reading it to understand how each part works.

Output format

Your top-level program should print out the MSTs from the Prim and Kruskal algorithms, in the form of a sequence of n − 1 edges (vertex pairs), where n is the number of vertices in the input graph.

Each MST should be preceded by a line which gives the name of the algorithm that generated it, the total cost of the MST, and the runtime it took your code to generate the given MST.

For the sample input file shown above, the output might look as follows:

Prim's algorithm MST (total cost: 15; runtime: 17ms)
(1,3)
(2,3)
(2,5)
(3,6)
(4,6)
Kruskal's algorithm MST (total cost: 15; runtime: 41ms)
(1,3)
(2,3)
(2,5)
(3,6)
(4,6)

What to submit

You need to submit all your code, properly commented and documented for easy readability as described. Logically, your code should consist of the following 6 components/modules (it is desirable to have each component in a separate file):

1. An implementation of a data structure which implements the GRAPH ADT [1]
2. An implementation of a data structure which implements the PRIORITYQUEUE ADT [2]
3. An implementation of Prim's algorithm, as a function which uses the above two data structures and takes a GRAPH instance as input and returns another GRAPH instance as output [2]
4. An implementation of the Union-Find data structure [2]
5. An implementation of Kruskal's algorithm, as a function which uses the above three data structures, and takes a GRAPH instance as input and returns another GRAPH instance as output [2]
6. A top-level program which is a function that takes as input the cost matrix for a graph in the given format, creates a GRAPH instance using it, calls the Prim and Kruskal functions on this instance and gets the GRAPH instances returned by each of them

(whilst also keeping track of the runtime for each call), and prints out these MSTs in the given format along with the respective total costs and runtimes [1]

Please package everything into a single zip/tar/rar file. You should also include a README file which explains to the user exactly how they should use your code to generate the MSTs for a given graph. Submit the file in Moodle
Any late submissions will be penalized, and may not be evaluated at all, depending on the extent of delay.

Collaboration and re-use policy
This is an individual assignment. You are free to discuss any aspect of the assignment with others; however, your code must be entirely written by you, without copying from anywhere. You may re-use your own code from earlier assignments for this course but should ensure that it is adapted as per the structuring requirements etc. specified here.