CS-2009 Design and Analysis of Algorithms, Fall-2022
Project

Due Date and Time: 4th December 2022 (1130 hrs) **Weight: 10%**

Instructions:

1. *Late submission will not be accepted.*
2. *Project can be done in a group. Max. group size is 3 members. Solve all the problems, in case of group size of 1 or 2 members.*
3. *Groups are required to list their information on the spreadsheet available on GCR latest by 21st November, 2022*
4. *Changes in the groups are possible till 22nd November, 2022, i.e., after 22nd November the groups cannot be changed. The schedule of demonstrations will be made on the basis of groups' information submitted till 22nd November, 2022.*
5. *Only the listed student (on the spreadsheet) will give the demo. related to a problem, i.e., other member(s) of the group (if any) cannot give demo. on behalf of some group member.*
6. *Submit all codes files in a zip folder*
7. *Submit report in doc and pdf format*
8. *File naming format for report should be "Member1-Roll#_Member2-Roll#_Member3-Roll#" e.g. 200123_205478_213254.pdf, 200123_205478_213254.doc*
9. *File naming format for code files should be "ProblemName_Member1-Roll#_Member2-Roll#_Member3-Roll#.cpp" e.g. Dijkstra_200123_205478_213254.cpp, Diameter_200123_205478_213254.cpp, Cycle_200123_205478_213254.cpp*
10. *There will be no credit if the given requirements are changed*
11. *Your solution will be evaluated in comparison with the best solution*
12. *There will be negative marking for not following the submission instructions on GCR.*
13. *Plagiarism may result in zero marks in the whole project regardless of the percentage plagiarized.*

Problem Description:

Provide efficient implementation of the algorithms for the problems given below. Test your algorithms on a large graph dataset containing minimum 1000 nodes. You are also required to provide a detailed analysis of your algorithms.

a. Find single source shortest path using Dijkstra and Bellman Ford algorithm.
b. Find minimum spanning tree using Prims and Kruskals algorithm.
c. Find breadth first and depth first traversal in a graph.
d. Find diameter of a graph. The ***diameter*** of a graph G = (V,E) is defined as $\max_{u,v \in V} \delta(u,v)$, that is, the largest of all shortest-path distances in the graph.
e. Detect cycle in a graph if exists any.

Requirements:

Implementation:

- Use any programming language to implement the algorithms
- Provide complete trace of your algorithm. For example, in case of using stack/queue, write complete trace like node insertion and removal for each step in a separate file.
- Make sure to use efficient implementation of the queue as discussed in the class for finding the shortest path.
- Store shortest paths in a separate file along with the printing on the screen
- Store result of minimum spanning tree on a separate file. Also show MST on the screen.
- Store diameter of a graph in a separate file including complete trace of the algorithm.
- Store execution time of all of the implementations.
- You can use any library for the visualization of your results.

Testing:

- Test your algorithms on a graph dataset available at http://snap.stanford.edu/data/index.html
- Your selected dataset must contain minimum 1000 nodes. For testing, you can pick different dataset for each of the above problems.
- Your code can be tested for any other graph dataset available at http://snap.stanford.edu/data/index.html .
- User can select any node as a source node for problem 'a' and 'c' during testing.
- No group should select same dataset

Report:

Your report must contain the following elements for the analysis;

1. Machine specification on which tests are run.
2. Complete algorithms with time complexity analysis for best/worst/average case.
3. Details of the dataset.
4. Comparison of algorithms using plots(graphs) of your results.
    a. For example, you can draw plots/graphs to compare run time of an algorithm against 100 nodes, 200 nodes and so on. Your graphs must consider different parameters like number of nodes, number of edges(dense/sparse), directed/undirected, weighted/unweighted. Make sure to provide results using large number of nodes to better distinguish run time for comparison.
    b. The execution time of a program can be calculated in C using the code available here: https://www.geeksforgeeks.org/measure-execution-time-function-cpp/
    c. The analysis should be performed showing the execution time of the program in the form of a XY plot showing various input sizes and corresponding execution times. Sample plot is available here. https://chart-studio.plotly.com/~MustafaCulban/11.embed

5. Discuss implementation structure (stack/queue, type of stack/queue), effect on selection of a particular stack/queue on the time complexity of the algorithm

Marks Distribution:

- Single Source Shortest Path (Dijkstra, Bellman Ford) [15+15]
- Minimum Spanning Tree (Prims, Kruskals) [15 + 15]
- Traversal Algorithms (BFS, DFS) [15+15]
- Diameter of a graph [20]
- Cycle detection in a graph [20]
- Average Degree [20 marks]

- For each of the above algorithms correct implementation with correct results (shortest path/MST/diameter/cycle) including traces of the stack/queue contains 60% marks. Write results of all of these algorithms in a separate file for demonstration.
- Analysis part contains 40% marks.

Each group member will do one of the following tasks. Please fill in the sheet which group member is responsible for their task

Member 1	**Member 2**	**Member 3**
Single Source Shortest Path (Dijkstra, Bellman Ford) [15,15] Diameter of a graph [20]	Minimum Spanning Tree (Prims, Kruskals) [15 + 15] Average Degree [20 marks]	Traversal Algorithms (BFS, DFS) [15+15] Cycle detection in a graph [20]