SMU® | LYLE
SCHOOL OF ENGINEERING

CSE 3353 ALGORITHMS: LAB 2

Overview: In this lab you will extend Lab 1 to include a Search class that extends from the Strategy Design pattern single interface for implementing and analyzing the performance of the following search algorithms: BFS, DFS, Dijkstra, A*.

Due: September 20, 2018 @ 4pm.

Code Requirements:

- Use same Strategy Class from Lab 1 with same functionality of member methods (including output stats and timing functionality)
- Create the following search algorithms
  - DFS (both iterative and recursive implementations)
  - BFS (both iterative and recursive implementations)
  - Dijkstra
  - A*
- Each Algorithm should:
  - Accept a two int values as input representation source and destination nodes
  - Return the path from source to destination, with total cost of path
  - Search from source to destination as well as from destination to source
  - Perform search over both adjacency list and adjacency matrix forms of loaded graphs
- Graph sets provided have the following format.
  - Three files per graph:
    - Graph.txt = Connectivity of directed graph
      - Each row is a csv, where first value is node and remaining values are children
        - 1,3,5,9 → Node 1 connects to Nodes 3,5,9
    - Weights.txt = edge weights for directed graph
      - Each row is a csv, where first two values are nodes and remaining value is weight
        - 1,2,3.78 → Cost of moving from Node 1 to 2 is 3.78
    - Positions.txt = position information for each node
      - Each row is a csv, where first value is node ID and reaming values are x,y, z position
        - 1, 3.78, -43.0,0 → Node 1 is located at position (3.78, -43.0, 0)
  - Sample graphs will be provided, but will also be tested on graphs not provided during grading.
- Program should also accept command line arguments for source and destination node as ints, This feature will be used during grading to test on private data set.
  - Example: ~> Myprogram.exe src dest

- Algorithm::Stats should now output the following information in an easy to read format
    - Algorithm Name
    - Returned Path
    - Number of nodes in returned path
    - Total Cost of path
    - Total distance of path
    - Number of nodes explored in path
    - Execution time to find path
- In main.cpp, loop through all algorithms and data sets to collect the performance timing. Your main.cpp should look similar to

```
Algorithm Search;

for (all Algo Search Types)
   for(all Graphs)
      Search.Load(input file type);
      Search.Execute();
      Search.Stats();
```

<u>Report Requirements:</u>

1. Perform the following test and collect results:
    a. Randomly select source and destination nodes from selected graph
    b. Perform search from src → dest and dest → src on both adjacency list and matrix for each of the coded algorithms, but keep results separate as they will compared against each other
    c. Collect following stats:
        i. Number of nodes in path returned
        ii. Number of nodes explored
        iii. Total time of execution
        iv. Total Distance of path
        v. For A* only: execute using a heuristic that combines cost with distance
            1. F(n) = distance (1 + cost)
    d. For each test normalize each collected stat between 0 -1
    e. Perform test 100 times and record both raw and normalized values into output file

2. Average all execution stats for each graph set and place into a summary table similar to:

Adjacency List Source → Destination		Search Algorithm					
		DFS Iterative	DFS Recursive	BFS Iterative	BFS Recursive	Dijkstra	A*
Avg Normalized Results	Nodes in Path						
	Nodes Explored						
	Execution Time						
	Distance						
	Cost						

3. You should have a summary table for the following combinations for each graph set provided
   a. Source → Destination, Adjacency Matrix
   b. Source → Destination, Adjacency List
   c. Destination → Source, Adjacency Matrix
   d. Destination → Source, Adjacency List
4. Create a plot for each of the performance metrics, where X axis is the graph set and Y axis is the metric value. Each plot should have each algorithm plotted. There should be a total of 4 plots (1 for each metric collected (i.e. Nodes in Path, Nodes Explored, Exe time, Distance))
5. Submit all tables and graphs in nicely formatted report in word. Which includes a writeup explaining results and differences between algorithms if any exist. Also provide the raw excel and data files used to create the charts and tables.

Submit inside of your class git repo.