COP5536 - Programming Assignment

Submitted By:
Pranav Achanta
UFID – 15512289
Email – pranav91690@ufl.edu

Program Structure

```
Main(){
        Switch(runMode)
        Case = 'r'
            runInRandomMode(int numOfNodes, float density, int startNodeNumber)
            Graph createRandomGraph(int numofNodes, float density, int
            startNodeNumber)
            computeShortestPathTree(startNodeNumber, char dataStructureType) –
            Run 2 times with dataStructure as 'L'(Leftist tree) and as 'F'(Fibonacci
            Heap).
        Case = 'u'
            runInUserMode(char dataStructureType, String filename)
            <Graph, startNodeNumber> createGraph (String filename)
            computeShorTestPathTree(startNodeNumber, dataStructureType)
}
```

Summary

Dijikstra's algorithm is implemented using a Priority Queue.
These are the complexities for the Priority Queue implemented using:

1. Leftist Tree Data Structure (n is the number of nodes in the data structure)

Operation	Actual Complexity	Ammortized Complexity
Put(Insert)	O(logn)	O(logn)
Remove Min/Remove	O(logn)	O(logn)
Meld	O(logn)	O(logn)

2. Fibonacci Heap Data Structure (n is the number of the nodes in the data structure)

Operation	Actual Complexity	Ammortized Complexity
Insert	O(1)	O(1)
Remove Min	O(n)	O(logn)
Decreasekey	O(n)	O(1)

Expected Result: Given these complexities, the Fibonacci heap is expected to give a better performance than a leftist tree in implementing the priority queue.

Actual Result: For the User input file with 1000 nodes, the priority queue implemented using the Leftist tree Data Structure took 22 ms, while the priority queue implemented using the Fibonacci Heap took 18 ms.

Experiment Results: Run in Random Mode
Test1: n=50

Density(%)	Running Time for Leftist Tree(ms)	Running Time for Fibonacci Heap(ms)
1	1	1
20	1	1
50	2	1
75	2	1
100	3	2

Observations: For very small n values, no considerable change in performance using a Fibonacci Heap

Test1: n=200

Density(%)	Running Time for Leftist Tree(ms)	Running Time for Fibonacci Heap(ms)
1	4	2
20	4	3
50	7	4
75	5	5
100	6	6

Observations: Almost Equal Performance for both the schemes

Test1: n=500

Density(%)	Running Time for Leftist Tree(ms)	Running Time for Fibonacci Heap(ms)
1	4	5
20	8	11
50	16	12
75	20	15
100	25	19

Observations: For this n, the leftist tree performs better until a edge density of 20%, which is unexpected.

Test1: n=1000

Density(%)	Running Time for Leftist Tree(ms)	Running Time for Fibonacci Heap(ms)
1	10	10
20	20	18
50	41	32
75	28	24
100	39	35

Observations: The Fibonacci heap performs at par till density of 20% and then outperforms the Leftist Tree for the higher densities, as expected.