Omid Sharghi

CS 146 Section 1

Programming Project 4

For Project 4, I used an ArrayList named ‘graph’ to represent the adjacency list. ‘graph’ holds ArrayLists of ‘Edge’ objects. As the text file is read, I take the edges and the weights from the file, create ‘Vertex’ objects for each vertex of an edge and add the vertices to another ArrayList named ‘vertices’, if they have not already been added. I also create ‘Edge’ objects and add them to an ArrayList titled ‘edges’.

For the portion of the project that required Prim’s algorithm, I implemented the algorithm detailed in CLRS. I first initialized all the vertex objects by setting their weights to a high value (1000) and setting the parent property to NULL. I then set the weight for vertex 0 to zero and added it to the Minimum Priority Queue along with all the other vertices. I also set vertex 0 as the root. Since vertex 0 has the lowest weight, it is popped and all its neighboring vertices in its adjacency list have their weights updated and their parent property set to vertex 0. I then continue to do this until the queue becomes empty. When the queue no longer holds vertex objects, my vertices have the lowest weights possible and a reference to a parent. For Prims, we can observe a dramatic increase in time between the 10,000EWG (134ms) and the largeEWG (546209ms) files. CLRS states that this implementation of Prim’s has a run time of O(VlgV + ElgV) = O(ElgV) if the priority queue’s implementation is a binary heap.

For finding the new algorithm to detect the heaviest edges in a cycle and remove them, I used both an iterative and recursive version that both utilized Depth First Search. I first implemented the recursive version, but as the running time increased with the file size, I decided an iterative version of Depth First Search would be more appropriate. Unfortunately, I was unable to get the run time for the largeEWG.txt file because it took so long. I tried running it over night, but it was still running in the morning. For the sake of studying for finals and meeting the project deadline, I could not not resolve this issue. The recursive and iterative version successfully remove the same number of edges from the adjacency list. For the smaller text files, the time difference between the recursive and iterative implementations are negligible. However, for the 10,000EWG, the time difference is about 50,000ms. CLRS states the recursive implementation of DFS, which I used, has a run time of O(V+E). My custom iterative approach iterates through all the edges, and for one vertex in the edge, it iterates through all its neighboring vertices, resulting in a O(E\*V) run time.

Prims Results For tinyEWG:

---------------------------

Number of Vertices: 8

Number of Edges: 16

Initial Adjacency List:

Adjacency List:

0 -> (7, 0.16) -> (4, 0.38) -> (2, 0.26) -> (6, 0.58)

1 -> (5, 0.32) -> (7, 0.19) -> (2, 0.36) -> (3, 0.29)

2 -> (3, 0.17) -> (0, 0.26) -> (1, 0.36) -> (7, 0.34) -> (6, 0.4)

3 -> (2, 0.17) -> (1, 0.29) -> (6, 0.52)

4 -> (5, 0.35) -> (7, 0.37) -> (0, 0.38) -> (6, 0.93)

5 -> (4, 0.35) -> (7, 0.28) -> (1, 0.32)

6 -> (2, 0.4) -> (3, 0.52) -> (0, 0.58) -> (4, 0.93)

7 -> (4, 0.37) -> (5, 0.28) -> (0, 0.16) -> (1, 0.19) -> (2, 0.34)

Initial Vertices: Vertices:

Vertex: 0 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 1 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 2 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 3 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 4 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 5 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 6 Min. Weight: 10000.0 Parent Vertex: Null

Vertex: 7 Min. Weight: 10000.0 Parent Vertex: Null

Final Vertices:

Vertex: 0 Min. Weight: 0.0 Parent Vertex: Null

Vertex: 1 Min. Weight: 0.19 Parent Vertex: 7

Vertex: 2 Min. Weight: 0.26 Parent Vertex: 0

Vertex: 3 Min. Weight: 0.17 Parent Vertex: 2

Vertex: 4 Min. Weight: 0.35 Parent Vertex: 5

Vertex: 5 Min. Weight: 0.28 Parent Vertex: 7

Vertex: 6 Min. Weight: 0.4 Parent Vertex: 2

Vertex: 7 Min. Weight: 0.16 Parent Vertex: 0

Total Minimum Weight: 1.8099999999999998

Time to run: 7ms

New Algorithm Recursive Results for tinyEWG:

-----------------------------------------------

Number of Vertices: 8

Number of Edges: 16

Initial Adjacency List:

Adjacency List:

0 -> (7, 0.16) -> (4, 0.38) -> (2, 0.26) -> (6, 0.58)

1 -> (5, 0.32) -> (7, 0.19) -> (2, 0.36) -> (3, 0.29)

2 -> (3, 0.17) -> (0, 0.26) -> (1, 0.36) -> (7, 0.34) -> (6, 0.4)

3 -> (2, 0.17) -> (1, 0.29) -> (6, 0.52)

4 -> (5, 0.35) -> (7, 0.37) -> (0, 0.38) -> (6, 0.93)

5 -> (4, 0.35) -> (7, 0.28) -> (1, 0.32)

6 -> (2, 0.4) -> (3, 0.52) -> (0, 0.58) -> (4, 0.93)

7 -> (4, 0.37) -> (5, 0.28) -> (0, 0.16) -> (1, 0.19) -> (2, 0.34)

Final Adjacency List:

Adjacency List:

0 -> (7, 0.16) -> (2, 0.26)

1 -> (7, 0.19)

2 -> (3, 0.17) -> (0, 0.26) -> (6, 0.4)

3 -> (2, 0.17)

4 -> (5, 0.35)

5 -> (4, 0.35) -> (7, 0.28)

6 -> (2, 0.4)

7 -> (5, 0.28) -> (0, 0.16) -> (1, 0.19)

Edges Removed: 9 Number Of Edges In MST: 7

Time to run: 3ms

New Algorithm Iterative Results for tinyEWG:

----------------------------------------------

Number of Vertices: 8

Number of Edges: 16

Initial Adjacency List:

Adjacency List:

0 -> (7, 0.16) -> (4, 0.38) -> (2, 0.26) -> (6, 0.58)

1 -> (5, 0.32) -> (7, 0.19) -> (2, 0.36) -> (3, 0.29)

2 -> (3, 0.17) -> (0, 0.26) -> (1, 0.36) -> (7, 0.34) -> (6, 0.4)

3 -> (2, 0.17) -> (1, 0.29) -> (6, 0.52)

4 -> (5, 0.35) -> (7, 0.37) -> (0, 0.38) -> (6, 0.93)

5 -> (4, 0.35) -> (7, 0.28) -> (1, 0.32)

6 -> (2, 0.4) -> (3, 0.52) -> (0, 0.58) -> (4, 0.93)

7 -> (4, 0.37) -> (5, 0.28) -> (0, 0.16) -> (1, 0.19) -> (2, 0.34)

Final Adjacency List:

Adjacency List:

0 -> (7, 0.16) -> (2, 0.26)

1 -> (7, 0.19)

2 -> (3, 0.17) -> (0, 0.26) -> (6, 0.4)

3 -> (2, 0.17)

4 -> (5, 0.35)

5 -> (4, 0.35) -> (7, 0.28)

6 -> (2, 0.4)

7 -> (5, 0.28) -> (0, 0.16) -> (1, 0.19)

Edges Removed: 9 Number Of Edges In MST: 7

Time to run: 2ms

Prims Results For mediumEWG:

----------------------------

Number of Vertices: 250

Number of Edges: 1273

Total Minimum Weight: 10.463509999999994

Time to run: 28ms

New Algorithm Recursive Results for mediumEWG:

------------------------------------------------

Number of Vertices: 250

Number of Edges: 1273

Edges Removed: 1024 Number Of Edges In MST: 249

Time to run: 95ms

New Algorithm Iterative Results for mediumEWG:

-----------------------------------------------

Number of Vertices: 250

Number of Edges: 1273

Edges Removed: 1024 Number Of Edges In MST: 249

Time to run: 75ms

Prims Results For 1000EWG:

----------------------------

Number of Vertices: 1000

Number of Edges: 8433

Total Minimum Weight: 20.773199999999992

Time to run: 55ms

New Algorithm Recursive Results for 1000EWG:

----------------------------------------------

Number of Vertices: 1000

Number of Edges: 8433

Edges Removed: 7434 Number Of Edges In MST: 999

Time to run: 740ms

New Algorithm Iterative Results for 1000EWG:

----------------------------------------------

Number of Vertices: 1000

Number of Edges: 8433

Edges Removed: 7434 Number Of Edges In MST: 999

Time to run: 479ms

Prims Results For 10000EWG:

----------------------------

Number of Vertices: 10000

Number of Edges: 61731

Total Minimum Weight: 65.24072000000007

Time to run: 134ms

New Algorithm Recursive Results for 10000EWG:

------------------------------------------------

Number of Vertices: 10000

Number of Edges: 61731

Edges Removed: 51732 Number Of Edges In MST: 9999

Time to run: 82481ms

New Algorithm Iterative Results for 10000EWG:

------------------------------------------------

Number of Vertices: 10000

Number of Edges: 61731

Edges Removed: 51732 Number Of Edges In MST: 9999

Time to run: 29286ms

Prims Results For largeEWG:

----------------------------

Number of Vertices: 1000000

Number of Edges: 7586063

Total Minimum Weight: 647.6630695499884

Time to run: 546209ms