CSCE 221 Cover Page
Homework Assignment #3
Due December 2 at 23:59 pm to eCampus

Oneal Abdulrahim 324007937

oneal.abdulrahim oneal.abdulrahim@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources			
People	Peer TA in class & office		
Web pages (provide URL)			
Printed material	Textbook		
Other Sources	Dr. Leyk slides/tracing demos		

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Oneal Abdulrahim Date 2 December 2016

Homework 3 (120 points)

due December 2 at 11:59 pm to eCampus.

Write clearly and give full explanations to solutions for all the problems. Show all steps of your work.

Reading assignment.

- Heap and Priority Queue, Chap. 8
- Graphs, Chap. 13

Problems.

1. (10 points) R-8.7 p. 361

   An airport is developing a computer simulation of air-traffic control that handles events such as landings and takeoffs. Each event has a *time-stamp* that denotes the time when the event occurs. The simulation program needs to efficiently perform the following two fundamental operations:

   - Insert an event with a given time-stamp (that is, add a future event)
   - Extract the event with smallest time-stamp (that is, determine the next event to process)

   Which data structure should be used for the above operations? Why? Provide big-oh asymptotic notation for each operation.

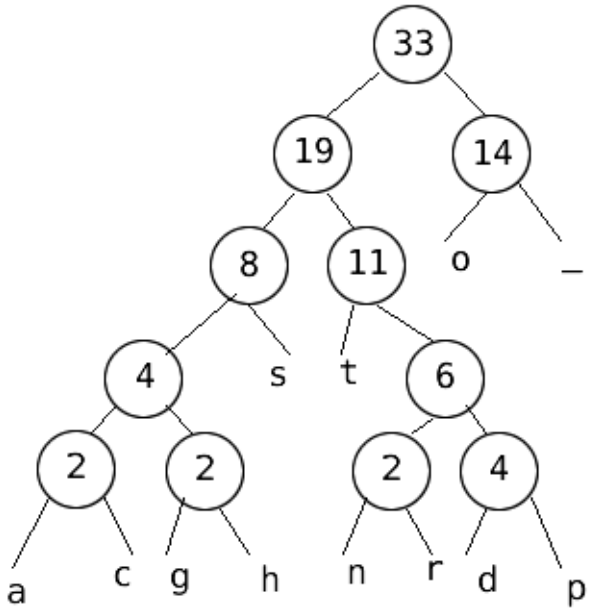
   A Heap would be most efficient, with the time stamps as the keys. A heap would support the insertion and removal operations, as well as provide search functionality. Its expected time complexity is $O(logn)$.

2. (10 points) R-12.14 p. 588

   Draw the frequency array. Use the minimum priority queue based on sorted array to build the Huffman tree for the string below. What is the code for each character and the compression ratio for this algorithm?

   "`dogs do not spot hot pots or cats`".

character	a	c	g	h	n	r	d	p	s	t	o	_
frequency	1	1	1	1	1	1	2	2	4	5	7	7

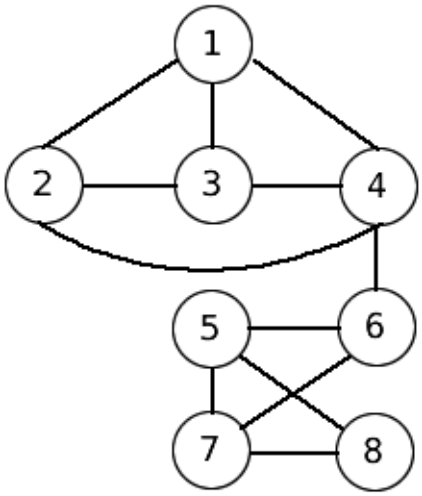


3. (10 points) R-13.5, p. 654

   Bob loves foreign languages and wants to plan his course schedule for the following years. He is interested in the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are given.

   If we visualize this as a directed graph, a possible sequence of courses is: LA15 -> LA16 -> LA22 -> LA31 -> LA32 -> LA127 -> LA141 -> LA169 -> LA126

4. (10 points) R-13.7, p. 655



(a)

(b) DFS traversal yields 1 -> 2 -> 3 -> 4 -> 6 -> 5 -> 7 -> 8
(c) BFS traversal can also yield 1 -> 2 -> 3 -> 4 -> 6 -> 5 -> 7 -> 8

5. (10 points) R-13.8, p. 655

Would you use the adjacency list structure or the adjacency matrix structure in each of the following cases? Justify your choice.
(a) The graph has 10,000 vertices and 20,000 edges, and it is important to use as little space as possible.
Using an adjacency list would take up the least amount of space, where the matrix would yield 10000 X 10000 values, compared to the 20000 in the adjacency list.

(b)The graph has 10,000 vertices and 20,000,000 edges, and it is important to use as little space as possible.
Since the adjacency matrix is dependent on the number of vertices (namely V^2), it is again best to use the list, giving us $V + E$ values in general.

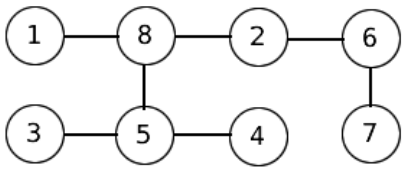
(c) You need to answer the query isAdjacentTo as fast as possible, no matter how much space you use.
If space is not an issue, then the adjacency matrix is best. This would ensure $O(1)$ access to values and adjacent vertices.

6. (10 points) R-13.16, p. 656
Show how to modify Dijkstra's algorithm to not only output the distance from v to each vertex in G, but also to output a tree T rooted at v such that the path in T from v to a vertex u is a shortest path in G from v to u

We can say that whenever we arrive at a vertex u where $(u \neq v)$ with the minimum $D(u)$ and remove it from the queue, we should add the edge closest to $u$to the tree T. That way, if the relax condition holds true, then we set the closest current minimum to u.

7. (10 points) R-13.17, p. 656
   There are eight small islands in a lake, and the state wants to build seven bridges to connect them so that each island can be reached from any other one via one or more bridges. The cost of constructing a bridge is proportional to its length. The distances between pairs of islands are given in the following table.
   Find which bridges to build to minimize the total construction cost.

   We must use a shortest spanning tree algorithm, like Prim's algorithm. This is where the bridges should go.



8. (10 points) R-13.31, p. 657
   A simple undirected graph is complete if it contains an edge between every pair of distinct vertices. What does a depth-first search tree of a complete graph look like?
   It will traverse the vertices in order, possibly looking like a polygon of sorts.

9. (10 points) C-13.10, p. 658
   An Euler tour of a directed graph G with n vertices and m edges is a cycle that traverses each edge of G exactly once according to its direction. Such a tour always exists if G is connected and the in-degree equals the out-degree of each vertex in G. Describe an O(n+m)-time algorithm for finding an Euler tour of such a digraph G.

   In one possible solution, we can visit a vertex and perform DFS traversal trying to find a cycle and close the loop. If we happen to close the loop but we have not visited all edges yet, we go back until we hit a vertex that has untraversed edges (so that the time complexity multiplier stays constant). Then we continue to search for a loop which contains edges that we have not visited yet. If we do find this loop, then we can simply combine the loops.

10. (10 points) C-13.15, p. 659
    Give an example of a weighted directed graph G with negative-weight edges but no negative-weight cycle, such that Dijkstra's algorithm incorrectly computes the shortest-path distances from some start vertex v.

    When following a particular weighted directed graph G with Dijkstra's algorithm, let us suppose that we are moving through vertices x -(1)-> y -(1)-> z. This would be a path of length 2. However, Dijkstra's algorithm would not return the also existing path x -(3)-> w -(-2)-> z as the shortest, whose length is 1, because of the negative edge