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

First Name: Robert Last: Quan UIN 622005281

User Name rhq66 E-mail address rhq66@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, books, etc			
People	Peer Teachers			
Web pages (provide URL)	cplusplues.com			
Printed material				
Other Sources				

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 Robert Quan Date April 23 2017

Homework 3 (100 points)

due April 23 at 11:59 pm to eCampus.

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) For this problem the best Data Structure that should be used is the **Binary Heap** because it enables us to sort entries based on a **Key**, which in this case would be *time-stamp*, that is associated with each flight. A Binary Heap can organize the tree with its root being the minimum key given, and in our case this works very well with the time-stamp. The big-oh notation for each operation in this case would be:

insert(e)	min()
O(logn)	O(1)

   Table 1: Big Oh Notation for Binary Heap

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`’’.

   a) For this problem I found the following frequency table and Tree in Fig 1.
   b) For the Huffman tree look at Fig 2. The Compression ratio is in the figure also.

   Figure 1: Frequency Array for Problem 2

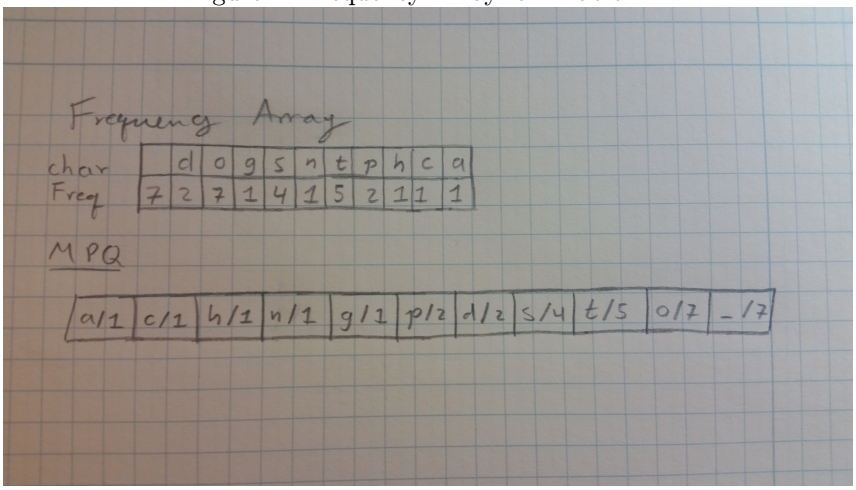
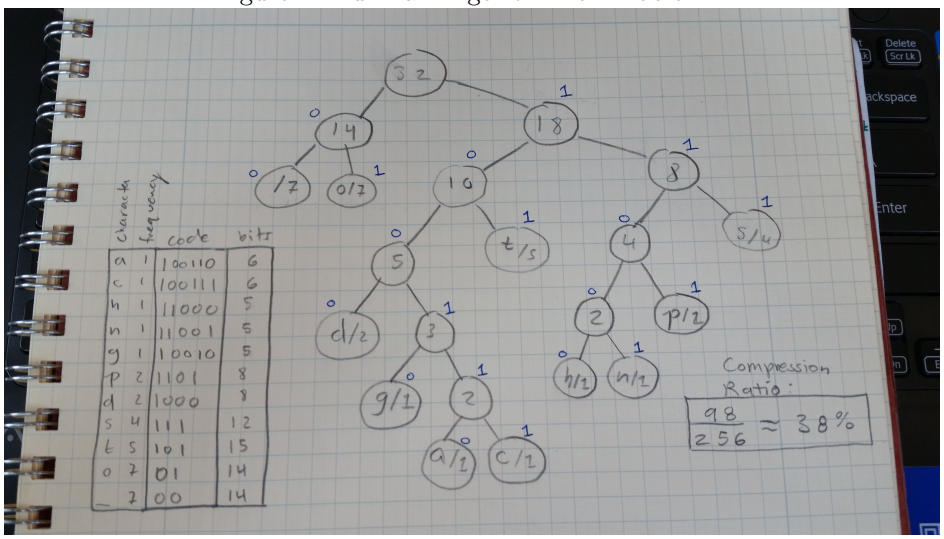


Figure 2: Huffman Algorithm for Problem 2

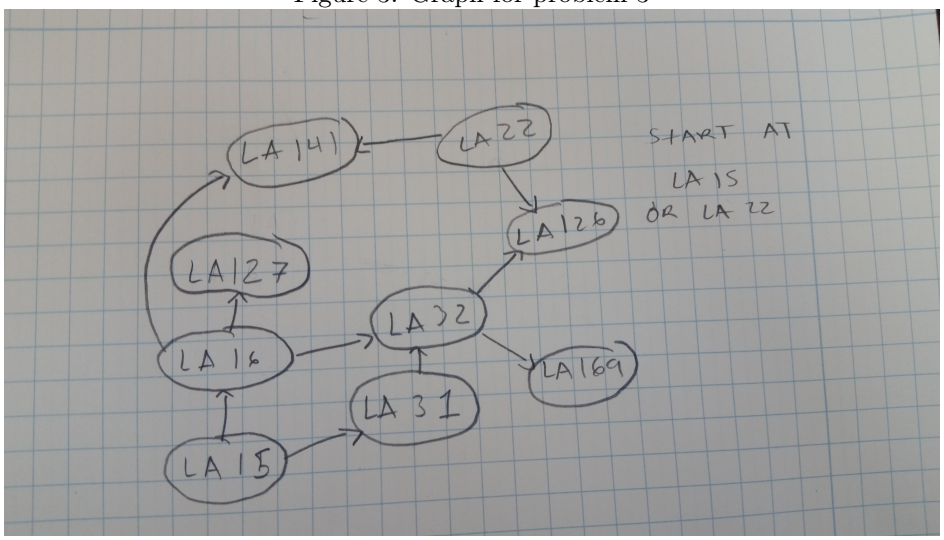


3. (10 points) R-13.5, p. 654.
   Bob loves foreign languages and wants to plan his course schedule for the following years. He is interested in the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are: LA15: (none) LA16: (LA15) LA22: (none) LA31: (LA15) LA32: (LA16, LA31) LA126: (LA22, LA32) LA127: (LA16) LA141: (LA22, LA16) LA169: (LA32) Find the sequence of courses that allows Bob to satisfy all the prerequisites.

   A possible configuration would be:

   LA15→LA31→LA16→LA32→LA22→LA126→LA127→LA141→LA169
   By converting this into a graph I obtain is in Fig 3:

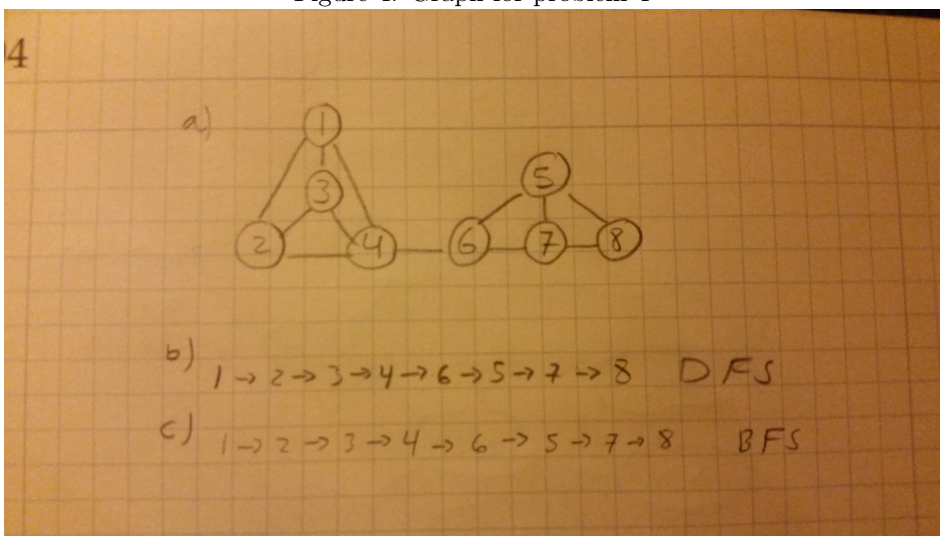
Figure 3: Graph for problem 3



4. (10 points) R-13.7, p. 655
   Let G be a graph whose vertices are the integers 1 through 8, and let the adjacent vertices of each vertex be given by the table below: Vertex 1 2 3 4 5 6 7 8 Adjacent Vertices (2, 3, 4) (1, 3, 4) (1, 2, 4)

(1, 2, 3, 6) (6, 7, 8) (4, 5, 7) (5, 6, 8) (5, 7) Assume that, in a traversal of G, the adjacent vertices of a given vertex are returned in the same order as they are listed in the table above. a. Draw G. b. Give the sequence of vertices of G visited using a DFS traversal starting at vertex 1. c. Give the sequence of vertices visited using a BFS traversal starting at vertex 1.

Figure 4: Graph for problem 4



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. b) The graph has 10,000 vertices and 20,000,000 edges, and it is important to use as little space as possible. c) You need to answer the query isAdjacentTo as fast as possible, no matter how much space you use.

a) We would use the **Adjacency List** structure because the space complexity is **O(n+m)** where n is the number of vertexes and m is the number of edges. In this case it would occupy less space than in a Adjacency Matrix.

b) In this case, we would also apply the **Adjacency List** structure to minimize the space complexity because if we were to use the Matrix form, that would give us a space complexity of $O(n^2)$ and that is bigger than O(n+m).

c) For the isAdjacentTo() function we would use the **Adjacency Matrix Data structure** because the function has a time complexity of O(1).

6. (10 points) R-13.16, p. 656

Show how to modify Dijkstras 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 do this by making two changes to Dijkstras Algorithm and implementing a variable closest[u] that stores its adjacent vertex in the cloud.
1) Whenever a vertex with the minimum distance is removed from the priority queue Q, we add the edge (closest[u],u) to the tree T.
2) if (D[u] + w(u,z)<D[z] holds, set closest[z] to u.

Figure 5: Dijkstras Algorithm to include Tree T

```
Algorithm DijkstraDistances(G, v)
  Input: A simple undirected weighted graph G with nonnegative edge weights,
 and a distinguished vertex s.
  Output: A label D[u], for each vertex u, such that D[u] is the length of a
 shortest path from v to u in G.
 {  for each  u ϵ G  do
       if  ( u = v )
           D[u] = 0;
       else
           D[u] = +∞;
     Create a priority queue Q containing all the vertices of G using the D labels
 as keys;
     Create an empty tree T;
     while  Q is not empty do
        {
         u = Q.removeMin();
         if ( u!=v )
             add the edge (closest[u], u) to T;
         for each  z ϵ Q such that  z is adjacent to u do
             if  ( D[u] + w((u,z)) < D[z] )  // relax edge e
                {  D[z] = D[u] + w((u,z));
                   Change to D[z]  the key of vertex z in Q;
                   closest[z]=u;
                }
        }
     return T and the label D[u] of each vertex u;
 }
```

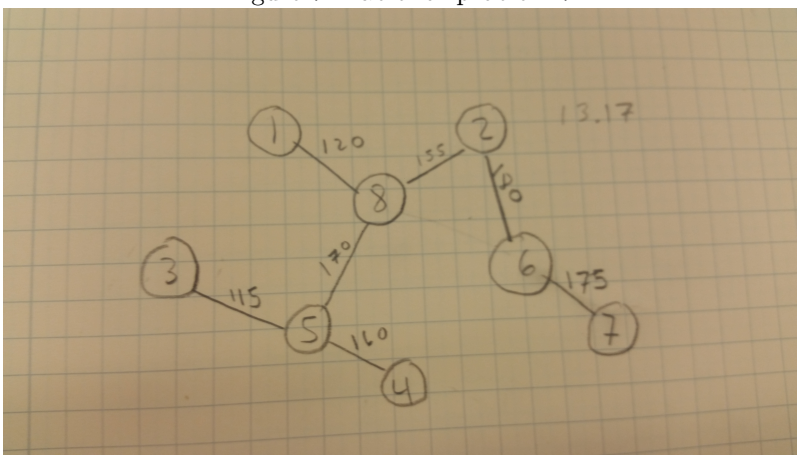
7. (10 points) R-13.17, p. 656
   There are eight small islands in a lake, and the state wants to build seven bridges to connect them so that each island can be reached from any other one via one or more bridges. The cost of constructing a bridge is proportional to its length. The distances between pairs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

Figure 6: Table for problem 7

	1	2	3	4	5	6	7	8
1	-	240	210	340	280	200	345	120
2	-	-	265	175	215	180	185	155
3	-	-	-	260	115	350	435	195
4	-	-	-	-	160	330	295	230
5	-	-	-	-	-	360	400	170
6	-	-	-	-	-	-	175	205
7	-	-	-	-	-	-	-	305
8	-	-	-	-	-	-	-	-

Find which bridges to build to minimize the total construction cost.

Figure 7: Table for problem 7



These are the bridges that should be built. This was found by using the Kruskals algorithm (which employs the greedy method), the cheapest bridge must first be made between island 3 and 5, the second between 1 and 8, then 2,8 etc. This method will find the first best match and use it without waiting for other options.

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?

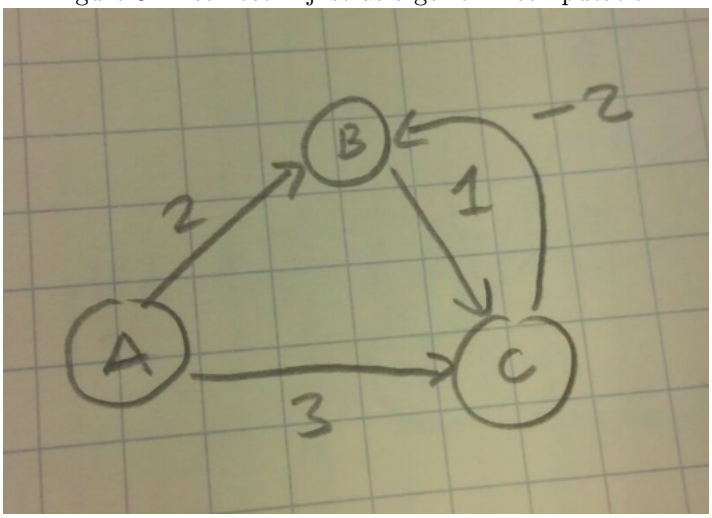
   The depth-first search tree of a complete graph is a path.

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.

   To obtain a time complexity of O(n+m), we must start at a vertex that contains maximum number of out degree and in degree edges. This way, we can traverse a cycle in different parts of the graph where we cover all the vertexes on that side then traverse to the vertex with the max number again to traverse the other side. We should terminate our cycle on the maximum number vertex again. This means we cover all vertexes n and cover all edges m, in degree and out. So out complexity will always be O(n+m), there is no minimum to cover all edges and vetexes.

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 Dijkstras algorithm incorrectly computes the shortest-path distances from some start vertex v.

Figure 8: Incorrect Dijkstras algorithm computation



Dijkstras algorithm, starting at vertex A, will conclude that d[A,B]=2, when in fact it is actually 1 due to the negative-weight edge.