CSCE 221 Cover Page
Homework Assignment #3
Due April 26 to CSNet

First Name Chris Last Name Comeaux UIN 622006681

User Name cmc236 E-mail address cmc236@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 Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources			
People			
Web pages (provide URL)	www.chegg.com		
Printed material	Data Structures and Algorithms in c++ by Goodrich, Tamassia, and Mount		
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 Chris Comeaux Date 4/25/2016

Homework 3

due April 26

1. (10 points) R-9.10 p. 417

   What is the result of Exercise R-9.7,
   when collisions are handled by double hashing using the secondary hash function $h_s(k) = 7-(k \bmod 7)$?

0	1	2	3	4	5	6	7	8	9	10
13	94	23	88	39	44	11	5	12	16	20

$h(K) = h(K) + i\,hs(K) \quad i = 0, 1, 2 \dots 10$

$h(12) = h(12) + 0 \cdot hs(12) = 41 \bmod 11 = 8$

$h(44) = h(44) + 0 \cdot hs(44) = 137 \bmod 11 = 5$

$h(13) = h(13) + 0 \cdot hs(13) = 44 \bmod 11 = 0$

$h(88) = h(88) + 0 \cdot hs(88) = 265 \bmod 11 = \cancel{5} \mid h(88) + 1 \cdot hs(88) = 5 + 1(7 - \overset{4}{88 \bmod 7}) = 5+3 = \cancel{8} \mid h(88) + 2 \cdot hs(88) = 5 + 2(3) = \cancel{11}$

$h(88) + 3 \cdot hs(88) = 5 + 3(3) = 14 \bmod 11 = 3$

$h(23) = h(23) + 0\, hs(23) = 74 \bmod 11 = \cancel{8} \mid h(23) + 1 \cdot hs(23) = 8 + 1(7 - 23 \bmod 7) = 13 \bmod 11 = 2$

$h(94) = h(94) + 0 \cdot hs(94) = 217 \bmod 11 = 1$

$h(11) = h(11) + 0 \cdot hs(11) = 38 \bmod 11 = \cancel{5} \mid h(11) + 1 hs(11) = 5 + 1(7 - \overset{3}{11 \bmod 7}) = \cancel{8} \mid h(11) + 2 hs(11) = 11 \bmod 11 = \cancel{0} \mid h(11) + 3 hs(11) = 14 \bmod 11 = \cancel{3} \mid$

$h(11) + 4 hs(11) = 5 + 4(3) = 17 \bmod 11 = 6$

$h(39) = h(39) + 0 \cdot hs(39) = 122 \bmod 11 = \cancel{1} \mid h(39) + 1 \cdot hs(39) = 1 + 1(7 - 39 \bmod 7) = 4 \bmod 11 = 4$

$h(20) = h(20) + 0\, hs(20) = 65 \bmod 11 = 10$

$h(16) = h(16) + 0\, hs(16) = 9$

$h(5) = h(5) + 0\, hs(5) = 9 \quad hs(5) = (7 - \overset{2}{5 \bmod 11}) \mid h(5) + 1 \cdot hs(5) = \cancel{11} \mid h(5) + 2 \cdot hs(5) = 9 + 4 = \cancel{13} \mid h(5) + 3 \cdot hs(5) = 15 \mid h(5) + 4 \cdot hs(5) = 17$

$13 \bmod 11 = \cancel{2} \mid h(5) + 5 \cdot hs(5) = 9 + 10 = 19 \bmod 11 = \cancel{8} \mid h(5) + 6\, hs(5) = 9 + 12 = 21 \bmod 11 = \cancel{10} \mid h(5) + 7 hs(5) = 9 + 14 = 23 \bmod 11 = \cancel{1} \mid$

$h(5) + 8 hs(5) = 9 + 16 = 25 \bmod 11 = 3 \mid \overset{i=9}{9 + 18} = 27 \bmod 11 = \cancel{5} \mid \overset{i=10}{9 + 20} = 29 \bmod 11 = 7$

2. (10 points) R-8.2 p. 361

   How long would it take toremove $\lceil log\, n \rceil$ smallest elements from a heap that contains n entries using the `removeMin()` operation?

   The remiveMin function takes $log_2 n$. For a heap with n elements, to remove the smallest $log\, n$ elements, with the removeMin function it would take $O(log_2(log_2 n))$.

3. (10 points) R-8.7 p. 361

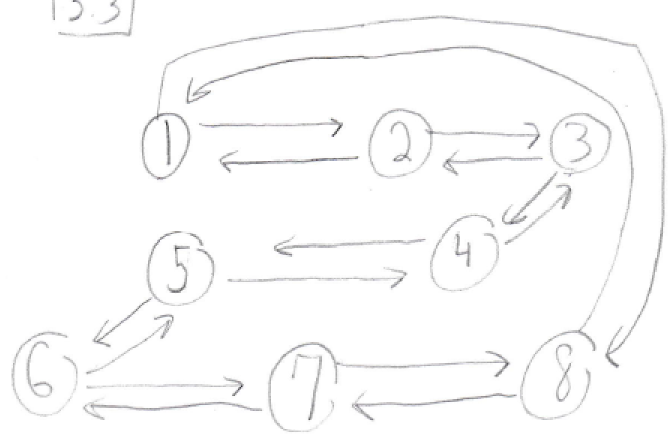
An airport is developing a computer simulation of air-traffic control that handles events such as landings and takeoffs. Each event has a *time-stamp* that denotes the time when the event occurs. The simulation program needs to efficiently perform the following two fundamental operations:

- 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?
A priority queue implemented with a binary hap would be best because it can insert and extract in logarithmic time. Even though an unsorted array can insert in constant time, it take linear time to removeMin. Likewise, a sorted array can removeMin in constant time but is takes linear time to insert a new event.

4. (10 points) R-13-3 and R-13-4, p. 654

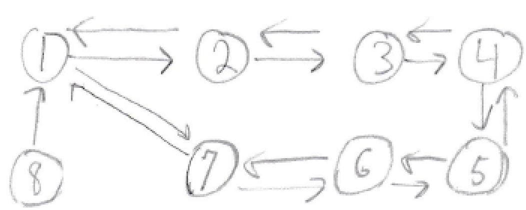
13.3



Euler tour from Vertex 1

1→2→3→4→5→6→7→8→1→8→7→6→5→4→3→2→1

13.4



8→1→2→3→4→5→6→7→2→7→6→5→4→3→2→1

Euler path

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

   A. Adjacency list. This is because it would take $10^8 (10,000 * 10,000)$ Boolean bits to represent the graph as a adjacency matrix. It would take less space to use an adjacency list (enough to hold 20,000 nodes).

   B. Adjacency Matrix. The adjacency matrix would take around $4 * 10^{14}$ bits which is a lot. However, the adjacency list would have to store 20,000,000 edges. Depending on how large each node is in the list, the space being used can grow very fast with that amount of nodes.

   C. Adjacency Matrix. Since space is not a factor, adjacency matrix would be the best choice because it can evaluate isAdjecentTo in constant time. On the other hand, the adjacency list would have to iterate through each edge adjacent to that vertex.

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

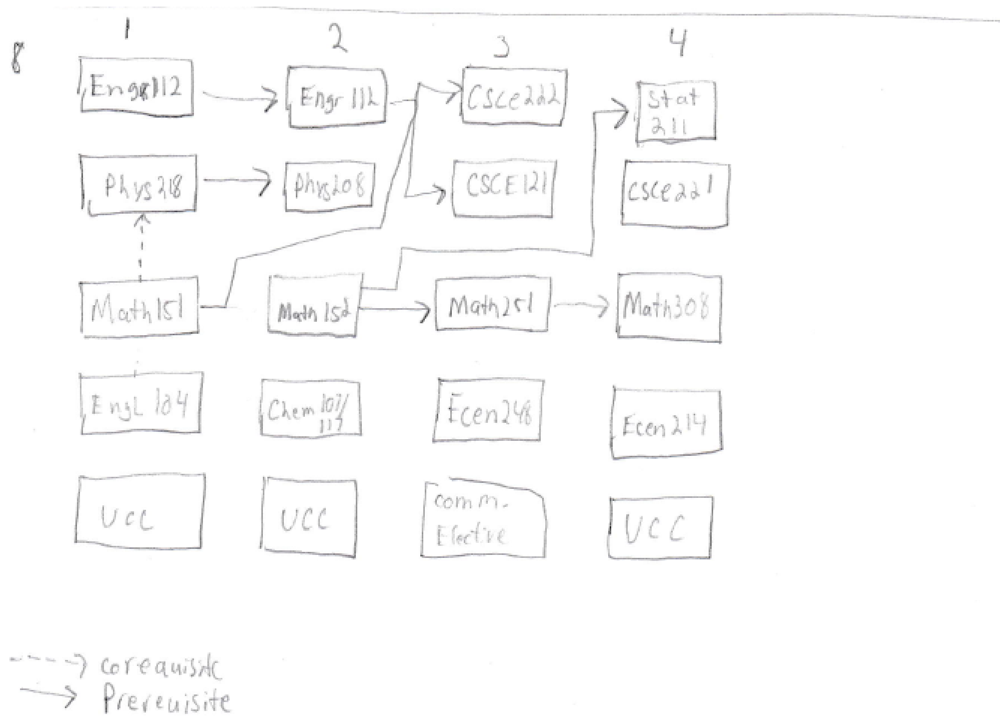
   Algorithm ShortestPath(G,v)

```
Initialize D[v]=0 and D[u] = positive infinity for all u != v.
Priority Queue Que; //contains all vertices of the Graoh G using key as the labels
while(!Queue.empty())
{
u=Que.removeMin() //pull u into cloud
for each vertex z adjacent to u that has not been visited
{
Update D[z] and display its distance and path
if(D[z] > D[u] + w(u,z) )// relax function
{
D[z] = D[u] + w(u,z)
P[z] = u
}
cout<<”Distance from “ <<v <<”to” <<u <<”is” <<D[z]
cout<<”Path from “ <<v <<”to” <<u <<”is v ->” <<z <<”->u”
}
}
```

7. (10 points) R-13.17, p. 656

   3->5 (115); 1->8(120); 2->8 (155); 4->5 (160); 5->8 (170); 2->6 (180)

8. (10 points) You want to help CS/CSE freshman students to prepare their course schedules for the first two years in the lower level division. By building a directed graph suggest order in which they should schedule their courses taking into account their corresponding prerequisites. A set of vertices represents courses and a set of edges represents a dependence of a given course on a course prerequisite.



9. (10 points) R-13.31, p. 656
A tree that contains all vertices in the graph in a straight line. In other words, every node, besides the last node, has exactly one child, and every node, besides the root, has exactly one parent. Like this: a->b->c->d->...->z

10. (10 points) Write what the running time, and provide its justification, of the Dijkstra's algorithm is for a sparse and dense graph and the priority queue implemented based on

   (a) a binary heap
   (b) an unsorted list
   (c) a sorted list

   Dijkstra's algorithm: $O(n * removeMin())$+$(m * DecreaseKey())$ Sparce = n=m. Dense=$n^2 = m$.

   A. Binary Heap: removeMin $= O(log_2 n)$, DecreaseKey $= O(n)$ -> $O(nlog_2 n) + (m * n)$.
   Sparse graph: $O(nlog_2 n) + (n^2)$= $O(n^2)$ without locators and $O(nlog_2 n)$ with locators.
   Dense graph: $O(nlog_2 n) + (n^3) = O(n^3)$ without locators and $O(n^2 log_2 n)$ with locators

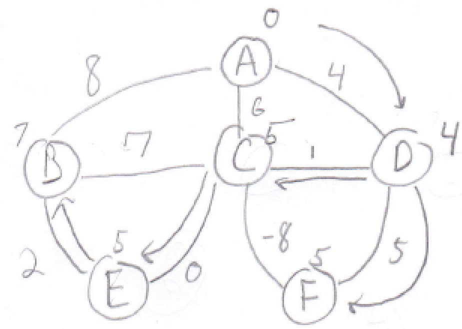
   B. Unsorted Graph: removeMin $= O(n)$ and DecreaseKey $= O(n)$ -> $O(n * n) + (m * n)$.
   Sparse graph: $O(n^2) + (n^2) = O(n^2)$.
   Dense graph: $O(n^2) + (n^3) = O(n^3)$

   C. Sorted Graph: removeMin $= O(1)$ and DecreaseKey $= O(n)$ -> $O(n + m * n)$.
   Sparse graph: $O(n + n^2) = O(n^2)$.
   Dense graph: $O(n + n^3) = O(n^3)$

11. (10 points) C-13.10, p. 658
For an Euler tour, every vertex needs two sets of edges. Two edges that com out from the previous and next vertex in the tour and two edges that does in the previous and next vertex in the tour. This means the each vertex has two edges going in and out of it. Therefore, the number of edges is 2 times the number of nodes meaning m=2n. So, $O(n + m) = O(n + 2n) = O(3n) = O(n)$.

12. (10 points) C-13.15, p. 659

12)



→ : Dijkstra's Path

·this is wrong because it does not take into account the weight of the -8, which should be included in the shortest path.

13. (10 points) C-13.18, p. 659
Using Dijkstra's algorithm we can change the Relax function from if(D[z] > D[u] + w(u,z)) to if(D[z] < D[u] + w(u,z)). This will find the longest path instead of the shortest. The running time based on the implementation with a priority queue using a binary heap and locators would be $O(nlog_2n)$ and $O(n^2)$ without locators.

Algorithm LongestPath(G,v)

Initialize D[v]=0 and D[u] = positive infinity for all u != v.
Priority Queue Que; //contains all vertices of the Graph G using key as the labels
while(!Queue.empty())
{
u=Que.removeMin() //pull u into cloud
for each vetex z adjencent to u that has not been visited
{
if(D[z] < D[u] + w(u,z))// relax function
{
D[z] = D[u] + w(u,z)
P[z] = u
}
}
}