CSCE 221 Cover Page

Homework Assignment #3
Due April 25 at 23:59 pm to eCampus

First Name Hunter Last Name Cleary UIN 625001547

User Name hncleary E-mail address hncleary@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				
Web pages (provide URL)	Listed Below			
Printed material	Data Structures and Algorithms (Textbook)			
Other Sources				

https://en.wikipedia.org/wiki/Eulerian_path
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
https://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/
https://en.wikipedia.org/wiki/Double_hashing

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 Hunter Cleary Date 4/24/18

Homework 3 (100 points)

due April 25 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.

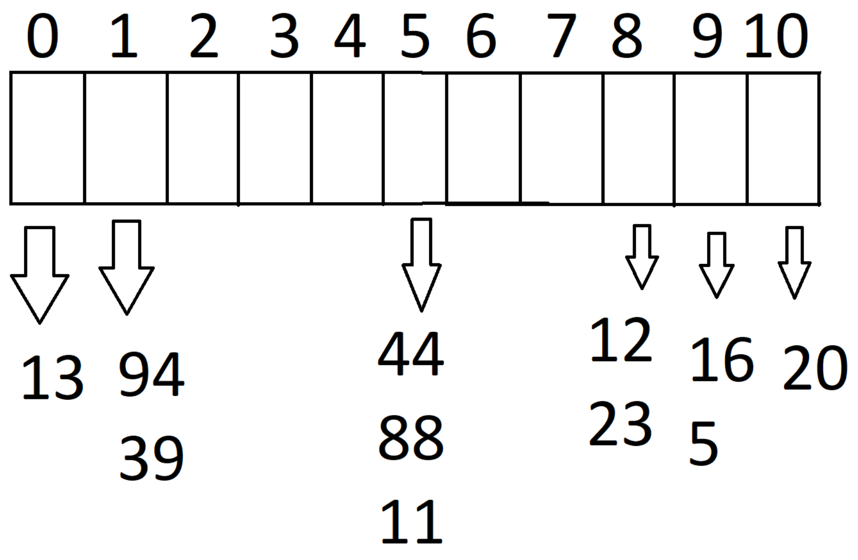
- Hash Tables Chap. 9
- Heap and Priority Queue, Chap. 8
- Graphs, Chap. 13

Problems.

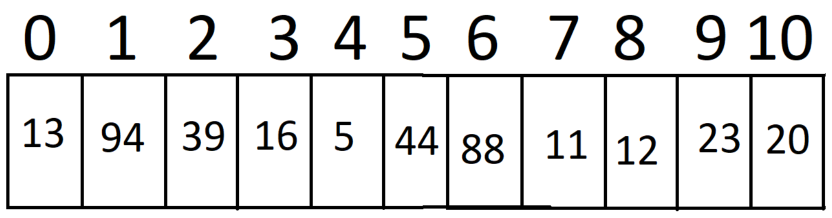
1. (10 points) R-9.7 p. 417

   Draw the 11-entry hash table that results from using the has function, $h(k) = (3k + 5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

   $h(k) = (3k + 5)\%11$
   $h(12) = 8,\ h(44) = 5,\ h(13) = 0,\ h(88) = 5,\ h(23) = 8,\ h(94) = 1,\ h(11) = 5,\ h(39) = 1,\ h(20) = 10,\ h(16) = 9,\ h(5) = 9$



2. (10 points) R-9.8 p. 417 What is the result of the previous exercise, assuming collisions are handled by linear probing?



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

3. (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	11	44	39	5	12	16	20

4. (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)

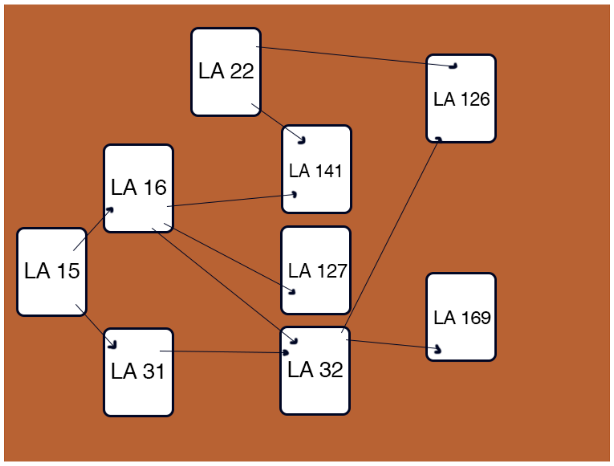
Which data structure should be used for the above operations? Why? Provide big-oh asymptotic notation for each operation.

Solution

The data structure that should be used for this scenario is a binary heap. The heap is made up of nodes that have a key and an element. These will correspond to the time and the event.

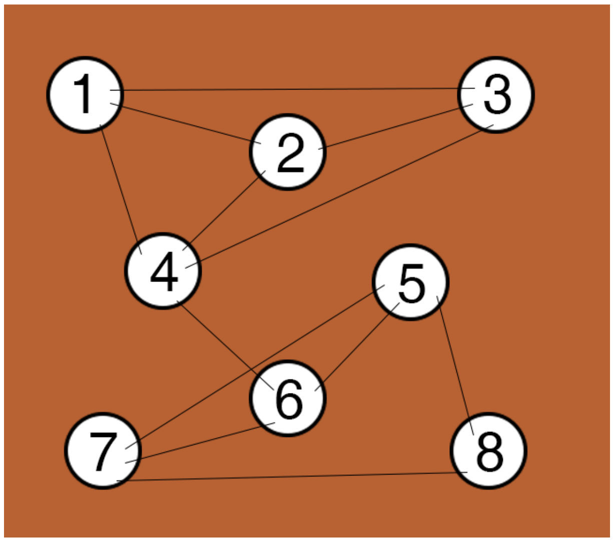
Insertion and search for the binary heap wold both be done in $O(logn)$ time.

5. (10 points) R-13.5, p. 654



$LA15- > LA16- > LA31- > LA22- > LA141- > LA127- > LA32- > LA169- > LA126$

6. (10 points) R-13.7, p. 655



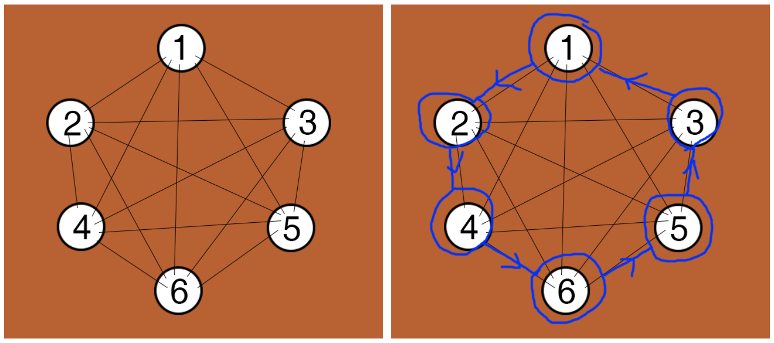
DFS Traversal: 1 -> 2 -> 3 -> 4 -> 6 -> 5 -> 7 -> 8
BFS Traversal: 1 -> 2 -> 3 -> 4 -> 6 -> 5 -> 7 -> 8

7. (10 points) R-13.16, p. 656
Initialize $D[v] = 0$ and $D[u] = \infty$ for every vertex that isn't the root.

Priority queue contains all vertices of the graph G using keys as labels for D.

For each vertex u, there is a closest u. Whenever a vertex u that isn't the root is removed from the priority queue add an edge (closest to u, u) to the tree.

8. (10 points) R-13.31, p. 657



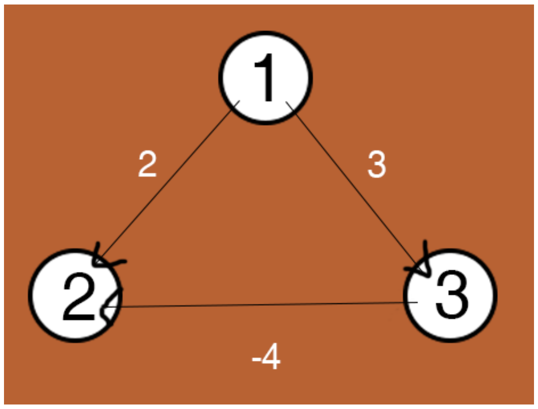
DFS Yields: 1 -> 2 -> 4 -> 6 -> 5 -> 3

9. (10 points) C-13.10, p. 658

   Set a starting vertex v. Find a cycle starting and ending at the starting vertex v so that it contains all edges going into and out of v. Put each edge of the cycle into a list in the order of the cycle. Traverse the list to find the first vertex that has an outgoing edge that is un-visited. If the vertex doesn't have the outgoing edge, then the algorithm is complete. If not, then recursively repeat function with the new edge to create a new cycle.

   The algorithm visits and puts into a list each edge in $O(1)$ time. The total time complexity is then $O(m + n)$. m is the number of edges, and n is the number of vertices in the graph.

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



   The algorithm will start on vertex 1. Edges 1 -> 2 and 1 -> 3 will be relaxed first (order is not consequential). These two will be pulled off of the priority queue. Dijkstra's algorithm will determine 2's shortest path to be 1 -> 2, when it is actually 1 -> 3 -> 2.