Braden Ackles & Alex Wilson

Assignment #7

CS 261

1. Q. In searching for a good hash function over the set of integer values, one student thought he could use the following: int index = (int) cos(value); // Cosine of 'value' What was wrong with this choice?

   A. The index is an integer, but cosine is a function that only has values between 0 and 1 so it only has two valid values, 0 and 1. Because of this if you used this as an index for a heap, the only buckets that can be accessed are 0 and 1.
2. Q. Can you come up with a perfect hash function for the names of days of the week? The names of the months of the year? Assume a table size of 10 for days of the week and 15 for names of the months. In case you cannot find any perfect hash functions, we will accept solutions that produce a small number of collisions (< 3).

   A. I could not work out perfect hash functions but these are easy computable ones and have few collisions.

Day	First letter	Lengths	Hash Value (First Letter+Length-1+length)%10
Sunday	18	6	9
Monday	13	6	4
Tuesday	20	7	3
Wednesday	23	9	0
Thursday	20	8	5
Friday	6	6	7
Saturday	19	8	4

Month	First Letter	Last Letter	Length	Hash Value
January	10	25	7	12
February	6	25	8	9
March	13	8	5	11
April	1	12	5	3
May	13	13	3	14
June	10	5	4	4
July	10	25	4	9
August	1	20	6	12
September	19	18	9	1
October	15	18	7	10
November	14	18	8	10
December	4	18	8	0

3. Q. The function containsKey() can be used to see if a dictionary contains a given key. How could you determine if a dictionary contains a given value? What is the complexity of your procedure?

A. In a dictionary, the key is the value that refers to the definition. To find a given Key it takes a O(N) complexity. This is because it has to search through every key to find the one you are looking for.

4. Q. Represent the following graph as both an adjacency matrix and an edge list

   A.

   Adjacency Matrix

	1	2	3	4	5	6	7	8
1	1	1	0	1	0	0	0	0
2	0	1	1	0	0	0	0	0
3	0	0	1	0	1	1	0	0
4	0	0	0	1	1	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	1	1
7	0	0	0	0	1	0	1	0
8	0	0	0	0	0	0	0	1

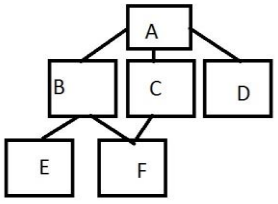
   Edge List:

1	{2,4}
2	{3}
3	{5, 6}
4	{5}
5	{}
6	{7, 8}
7	{5}
8	{}

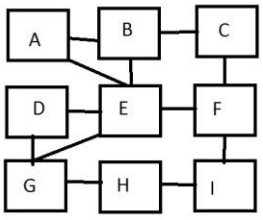
5. Q. Construct a graph in which a depth first search will uncover a solution (discover reachability from one vertex to another) in fewer steps than will a breadth first search. You may need to specify an order in which neighbor vertices are visited. Construct another graph in which a breadth-first search will uncover a solution in fewer steps.

   A.

   Graph A:



   Graph B:



6. Q. Complete Worksheet 41 (2 simulations). Show the content of the stack, queue, and the set of reachable nodes.

   A.

Iteration	Stack (Top - Bottom)	Reachable
0	1	Empty
1	6, 2	1
2	11, 2	1, 6
3	16, 12, 2	1, 6, 11
4	21, 22, 2	1, 6, 11, 16
5	22, 12, 2	1, 6, 11, 16, 21
6	23, 17, 12, 2	1, 6, 11, 16, 21, 22
7	17, 12, 2	1, 6, 11, 16, 21, 22, 23
8	13, 12, 2	1, 6, 11, 16, 21, 22, 23, 17
9	18, 8, 12, 2	1, 6, 11, 16, 21, 22, 23, 17, 12
10	18, 8, 12, 2	1, 6, 11, 16, 21, 22, 23, 17, 12, 13
11	8, 12, 2	1, 6, 11, 16, 21, 22, 23, 17, 12, 13, 18
Iteration	Queue (Front-Back)	Reachable
0	1	Empty
1	2, 6	1
2	6, 3, 7	1, 2
3	3, 7, 11	1, 2, 6
4	7, 11, 4, 8	1, 2, 6, 3
5	11, 4, 8	1, 2, 6, 3, 7
6	4, 8 , 12, 16	1, 2, 6, 3, 7, 11
7	8, 12, 16, 5, 9	1, 2, 6, 3, 7, 11, 4
8	12, 16, 5, 9, 13	1, 2, 6, 3, 7, 11, 4, 8
9	16, 5, 9, 13, 13, 17	1, 2, 6, 3, 7, 11, 4, 8, 12
10	5, 9, 13, 13, 17, 21	1, 2, 6, 3, 7, 11, 4, 8, 12, 16
11	9, 13, 13, 17, 21, 10	1, 2, 6, 3, 7, 11, 4, 8, 12, 16, 5
12	13, 13, 17, 21, 10, 14	1, 2, 6, 3, 7, 11, 4, 8, 12, 16, 5, 9
13	13, 17, 21, 10, 14, 18	1, 2, 6, 3, 7, 11, 4, 8, 12, 16, 5, 9, 13

7. Q. Complete Worksheet 42 (1 simulation). Show the content of the priority queue and the cities visited at each step.

   A.

Iteration	Priority Queue	Reachable
0	Pensacola (0)	empty
1	Phoenix (5)	Pensacola (0)
2	Pueblo (8), Peoria (9), Pittsburg (15)	Phoenix (5)
3	Peoria (9), Pierre (11), Pittsburg (15)	Pueblo (8)
4	Pierre (11), Pittsburg (14), Pittsburg (15)	Peoria (9)

5	Pendleton (13), Pittsburg (14), Pittsburg (15)	Pierre (11)
6	Pittsburg (14), Pittsburg (15)	Pendleton (13)
7	Pittsburg (15)	Pittsburg (14)
8	empty	empty

8. Q. Why is it important that Dijkstra's algorithm stores intermediate results in a priority queue, rather than in an ordinary stack or queue?
   A. Because results that or immediate will be at the shorter part of the queue and there for get removed sooner. A priority queue allows for the item that is the most important to happen next.
9. Q. How much space (in big-O notation) does an edge-list representation of a graph require?
   A. Edge-list representations have a O(V+E) space to represent a graph.
10. Q. For a graph with V vertices, how much space (in big-O notation) will an adjacency matrix require?
    A. Adjacency matrix representation requires O(V^2) of space; for a matrix with V matrices
11. Q. Suppose you have a graph representing a maze that is infinite in size, but there is a finite path from the start to the finish. Is a depth first search guaranteed to find the path? Is a breadth-first search guaranteed to find the path? Explain why or why not.
    A. A breadth-first search will find the solution fastest because it explores every possibility at once not just one path. A depth-first search, as the name says goes to the deepest possible position in a route and then goes back, if the maze is infinite it will never hit the end and there for run for ever. There is a very small chance that the DFS chooses the correct path and finds it fast than the BFS but it is very unlikely.