CS255 HOMEWORK 1 SPRING 2020

Q1: CHARACTERS MEDIAN AND MODE

Given a **n** sized unsorted array, that contains only latin characters, describe an efficient algorithm that finds the median (lower if even n) and the mode (most frequent element). If there are multiple elements that appear maximum number of times, output any one of them.

What is the time and space complexity?
Examples:

Input : array a[] = {F, A, D, A, B, A}

Output : Mode = A Median = A

Input : array a[] = {G, F, R, R, C}

Output : Mode = R Median = G

Q2: WHERE IS THE ZERO?

Alice has a sorted array A[1, . . . , n] with distinct positive numbers. Now Bob has another almost sorted array B[1, . . . , n + 1], which is derived from inserting a zero into A. Help Bob to discover the index of this zero. Design and analyze an efficient algorithm for him to find the index of the zero in B. Hint: use A and B in your solution.

See next example. If

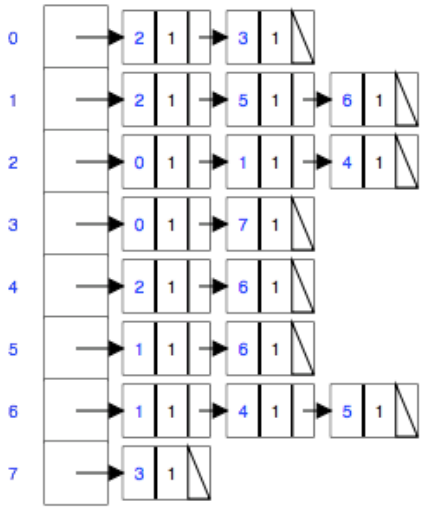
A : 1, 3, 4, 6, 7, 8, 9, 20

B : 1, 3, 0, 4, 6, 7, 8, 9, 20.

Your algorithm should return 3 in this case, which is the index of the zero in B (starting from 1).

Q3: SOCIAL GRAPH

Given the following adjacency list representation of a graph (every node has the label of the neighbor and the weight of the edge, note all have 1 weight), draw the graph, give the visited node order for each type of graph search, starting with 0.



a. Draw the graph

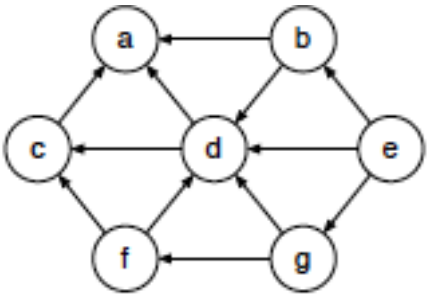
b. Breadth First Search:

c. Depth First Search:

d. Give the length of the shortest path from 0 to 5:

Q4: TOPOLOGICAL SORTING

Run the topological sort algorithm on the graph (Hint: where should you start?)



Q5: MATCHING CLASSES

For each function on the left, give the best matching order of growth of the running time on the right.

public static int f1(int n) {
  int x = 0;
  for (int i = 0; i < n; i++)
  x++;
	A. O(log n)
	B. O(n)
	C. O(n log n)
	D. O(n^2)

return x;
}
 	
public static int f2(int n) {
if (n == 0) return 0;
return f2(n/2) + f1(n) + f2(n/2);
}
 	
public static int f3(int n) {
if (n == 0 || n==1) return 1;
return f3(n-1)+f1(n);
}
 	
public static int f4(int n) {
if (n == 1) return 0;
return 1 + f4(n/3);
}
 	
public static int f6(int n) {
if (n == 1) return 0;
return
1 + f6(n/3) +f6(n/3) +f6(n/3);
}
	E. O(n^3

Q6: MULTIGRAPH TO SIMPLE GRAPH

Given an adjacency-list representation of a multigraph G =(V, E), describe an O(V + E)-time algorithm to compute the adjacency-list representation of the “equivalent” undirected graph G’=(V,E’), where E’ consists of the edges in E with all multiple edges between two vertices replaced by a single edge and with all self-loops removed.

Q7: LOMBARDI GRAPHS

A number of art museums around the country have been featuring work by an artist named Mark Lombardi (1951–2000), consisting of a set of intricately rendered graphs. Building on a great deal of research, these graphs encode the relationships among people involved in major political scandals over the past several decades: the nodes correspond to participants, and each edge indicates some type of relationship between a pair of participants. And so, if you peer closely enough at the drawings, you can trace out ominous-looking paths from a high-ranking U.S. government official, to a former business partner, to a bank in Switzerland, to a shadowy arms dealer. Such pictures form striking examples of social networks, which have nodes representing people and organizations, and edges representing relationships of various kinds. And the short paths that abound in these networks have attracted considerable attention recently, as people ponder what they mean. In the case of Mark Lombardi’s graphs, they hint at the short set of steps that can carry you from the reputable to the disreputable. Of course, a

single, spurious short path between nodes v and w in such a network may be more coincidental than anything else; a large number of short paths between v and w can be much more convincing. So in addition to the problem of computing a single shortest v-w path in a graph G, social networks researchers have looked at the problem of determining the number of shortest v-w paths. This turns out to be a problem that can be solved efficiently. Suppose we are given an undirected graph G = (V , E), and we identify two nodes v and w in G. Give an algorithm that computes the number of shortest v-w paths in G. (The algorithm should not list all the paths; just the number suffices.) The running time of your algorithm should be O(m + n) for a graph with n nodes and m edges.

A Lombardi drawing