CS 251 Homework 3: Data Structures and Algorithms (100 points)

Out: March 1st, 2019 (8:00 pm)
Due: March 22nd, 2019 (8:00 pm)

Purdue ID:

Instructions: Each **multiple choice** question has only one correct answer. Additionally, you must provide an **explanation** for your answer (2-3 short sentences is sufficient). Answers without proper explanations, even though it is correct, will be graded with 0 points.

For **true/false** questions, you must answer true or false AND explain your reasoning.

For all other questions, follow the specific instructions carefully.

For all questions, put your responses in the boxes provided.

1. [5pts] What would be the complexity of Quicksort if the pivot is always the median? (Assume the median can always be found in linear time).

   a. Always O(n log n)

   b. Always $O(n^2)$

   c. Worst case: $O(n^2)$, average case: O(n log n)

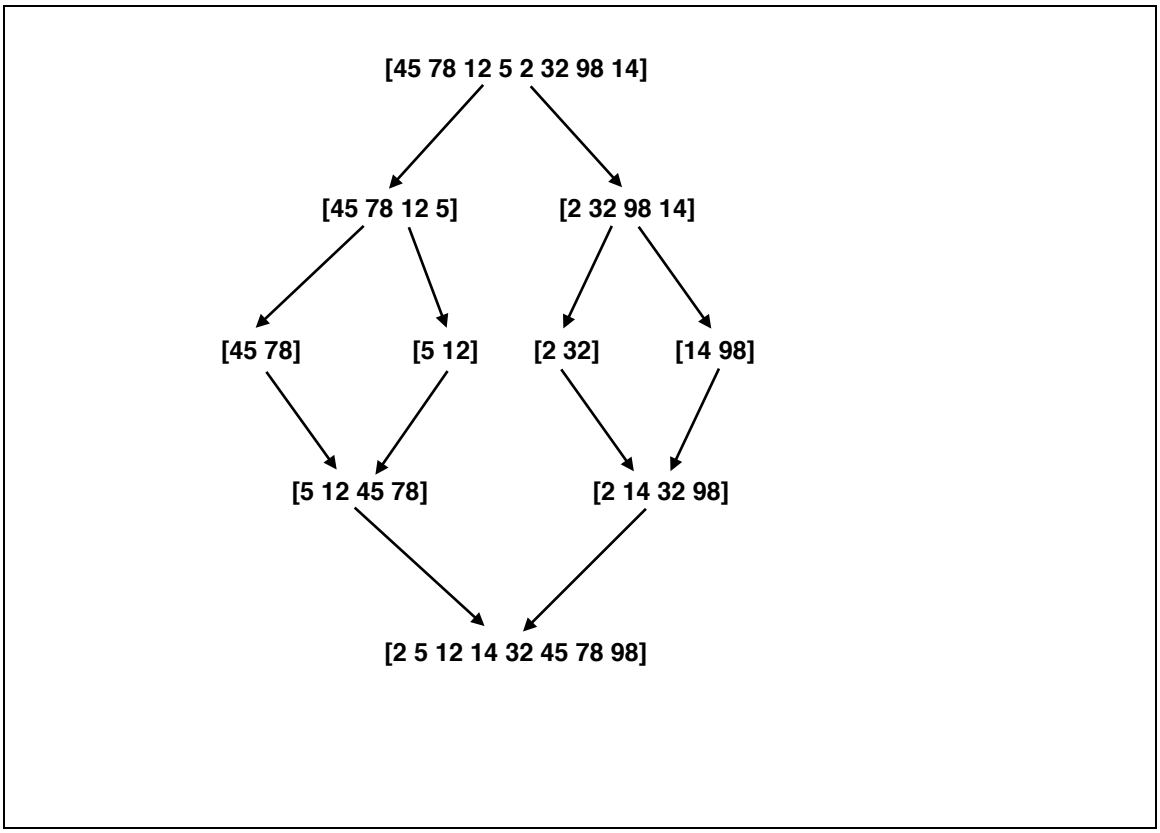
   d. Worst case: $O(n^2 \log n)$, average case: $O(n^2)$

   a

   Quicksort will have the worst case O(n^2) when the pivot is always the max/min. When the pivot is always the median, it's always O(nlogn).

2. [10pts] Visually show what happens to the following list of numbers as it goes through the MergeSort algorithm. You can show it using an upside-down tree structure starting with lists of one element each and showing the merges.

   45, 78, 12, 5, 2, 32, 98, 14



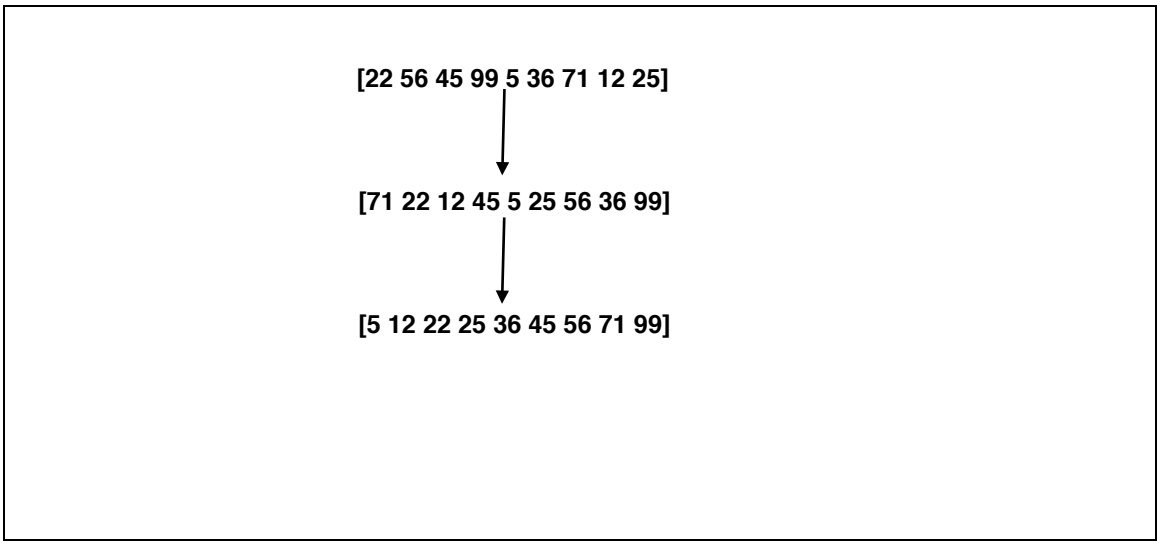
3. [10pts] Alice is participating in a competitive programming contest in which most programs end up with TLE (Time Limit Exceed). In one of the tasks, she has to sort n integers. She can use SelectionSort and/or MergeSort provided in the code library. For inputs of size n, SelectionSort runs in ($8n^2$) steps, while MergeSort runs in (128nlogn) steps on the server. Under what circumstances, if any, is SelectionSort more efficient than MergeSort given these specifications? Show your work and explain your reasoning.

Selectionsort will run in 8n^2 steps. Mergesort will run for (128nlogn) steps.

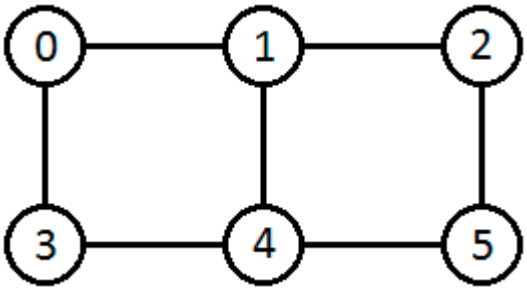
When n/logn < 16, selectionsort is better. It indicates a relatively small n, selection sort is better.

4. [10pts] Visually show what happens to the following list as it goes through RadixSort. You do not need to show the buckets, but you should show the contents of the list after each iteration of the stableSort method used within Radix Sort.

22, 56, 45, 99, 5, 36, 71, 12, 25



5. a) [5pts] Write down the adjacency matrix for the following undirected graph.



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

b) [5pts] Given an undirected graph G = (V, E) where |V| = 1000 and |E| = 1000, what is the space required to represent the graph as (i) an adjacency matrix and (ii) an adjacency list. (iii) Would you describe this graph as **dense** or **sparse**? Explain your reasoning for all parts of the question.

i) Adjacency matrix: 1000^2. It is the square of the number of vertices.

ii) Adjacency list: 1000. It takes the space of the number of edges. |E|=1000.

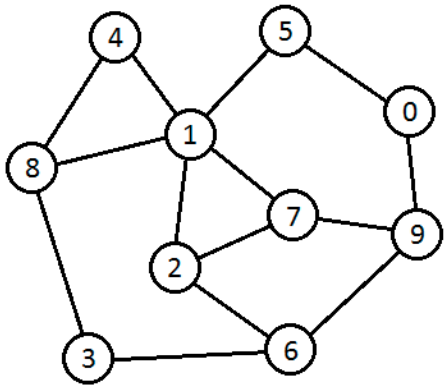
iii) A sparse graph is defined by G=(V, E) where |E|=O(|V|). Since |V|=|E|=1000, we can conclude it is a sparse graph.

c) [5pts] Now consider a simple, undirected graph G = (V, E) where |V| = 128 and G is as dense as it could possible be. What is |E|? Show your work and explain your reasoning.

|E|= (1+127)*127/2 = 8128
For the first node, there are 127 possible edges without repetition.
Second node has 126, third node 125, …

In total, there are 127 + 126 + … + 1 nodes.

6. a) [5pts] Given the graph and associated adjacency list representation here, show the result of a BFS of the graph starting at vertex 7 by (i) listing the vertices in the order in which they are visited and (ii) listing the depth of each vertex in the resulting tree where the root (vertex 7) is at depth 0.



0	5, 9
1	2, 8, 4, 5, 7
2	1, 7, 6
3	6, 8
4	1, 8
5	1, 0
6	3, 2, 9
7	2, 1, 9
8	4, 1, 3
9	0, 7, 6

(i) The order of BFS is:
[7] —0
[2 1 9] —1
[6 8 4 5 0] —2
[3] —3

7 -> 2 -> 1 -> 9 -> 6 -> 8 -> 4 -> 5 -> 0 -> 3

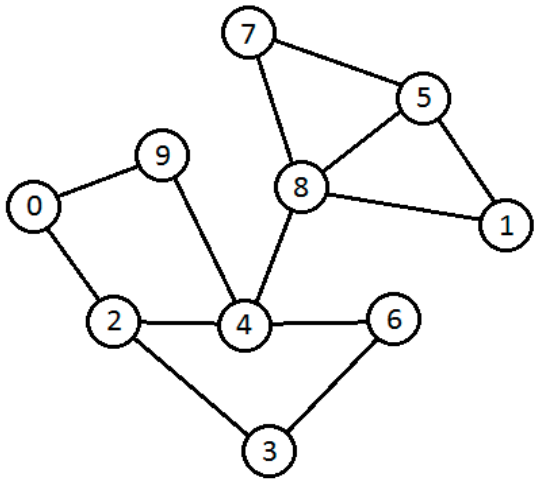
(ii)
7 -> 2 -> 1 -> 9 -> 6 -> 8 -> 4 -> 5 -> 0 -> 3
0 1 1 1 2 2 2 2 2 3

b) [5pts] [True/False] It is possible to use BFS to find the shortest path in a weighted graph in which all the edges have the same weight.

True. The path in BFS has the least number. If the edges have the same weight, it is guaranteed to be the shortest path.

7. a) [5pts] Which of the following is not the result of a DFS starting at vertex 3 of the graph below?

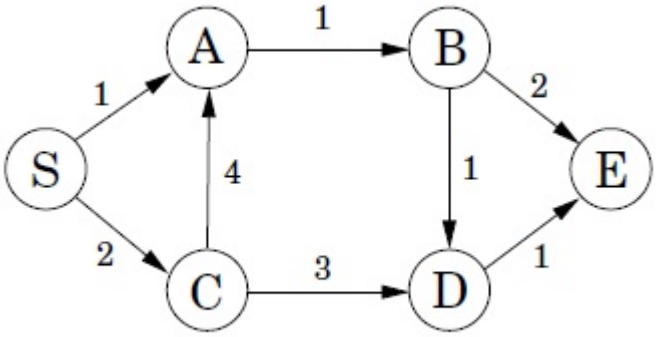
   A. 3, 6, 4, 8, 5, 7, 9, 0, 2, 1
   B. 3, 6, 4, 9, 0, 2, 8, 1, 5, 7
   C. 3, 2, 4, 6, 8, 1, 5, 7, 9, 0
   D. 3, 2, 0, 9, 4, 6, 8, 7, 5, 1



A.
The correct order close to A should be 3, 6, 4, 8, 5, 7, 1, 9, 0, 2.

8. [15pts] Given a weighted directed graph G with V vertices and E edges and non-negative edge weights, write pseudocode to find the *second shortest* path from a given source ***s*** to destination ***t***. If there are multiple shortest paths from the source to the destination, then the total path weight of the second shortest will be the same as the total weight of the shortest path. (e.g. if there are two shortest paths of weight 7, then both the shortest and the second shortest path weights are 7). If there is only one path between the source and the destination, then the second shortest path is null.

For example in the following graph, the shortest path from C to E has a weight of 4 (C-D-E). The second shortest path from C to E has a weight of 7 (either C-A-B-E or C-A-B-D-E).



If you use an existing algorithm (e.g. Dijkstra's Algorithm or Bellman-Ford Algorithm) as a subroutine, then just mention the name of the algorithm. There is no need to show the pseudocode for the algorithm when you are using it as discussed in class. But if you *extend* an existing algorithm (e.g. by adding to it), then you need to show the entire pseudocode.

Psudocode:
Revised from k shortest path code
s: source node
t: destination node

```
P = empty
count = 0, for all u in V
insert path Ps = {s} into B with cost 0
while B is not empty and count < 2:
    let Pu be the shortest cost path in B with cost C
    B = B- {Pu} , count = count + 1
    if u = t then P = P and {Pu}
    if count <= 2 then
        for each vertex v adjacent to u:
            let Pv be a new path with cost C + w(u, v) formed by concatenating edge (u, v) to path P
            insert Pv into B
            return P
return P
```

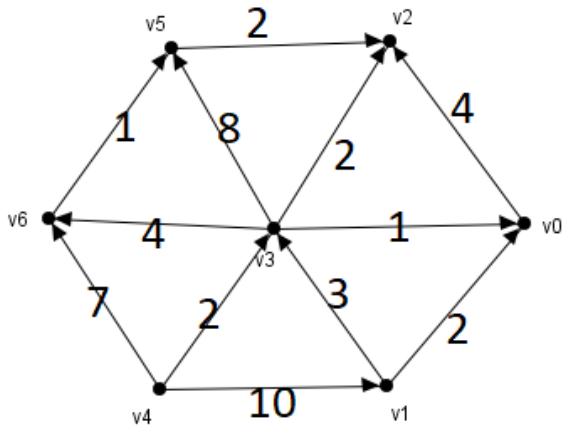
9. a) [5pts] Let **G** be a simple directed graph and let **G*** be the transitive closure of **G.** Given the adjacency matrix representation of **G***, determine if each statement below is **true** or **false** and explain your reasoning.

   i) The original graph **G** can be topologically sorted.

   ii) The original graph **G** is strongly connected.

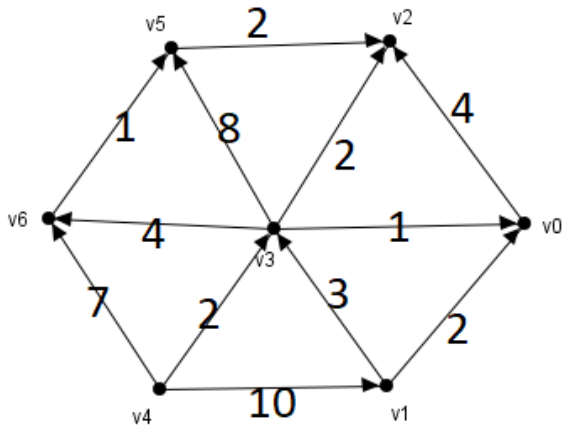
	0	1	2	3	4
0	0	0	0	0	0
1	1	0	0	1	0
2	1	1	0	1	0
3	1	0	0	0	0
4	1	1	1	1	0

i) True. G*, as the transitive closure of G, is able to be topological sorted. Therefore, G can be topological sorted.
ii) False. It can be identified from G* that 4 cannot reach to every node. Therefore, G is not strongly connected.



b) [5pts] What is the total weight of the shortest path from v_4 to v_5? Please also list the vertices along the path.

7: v4 -> v3 -> v6 -> v5



10. a) [5pts] In the above graph, show the results of Dijkstra's shortest path algorithm starting at **v4** by listing the vertices in the order in which they are visited.

v0: 3
v1: 10
v2: 4
v3: 2
v4: 0
v5:7 [10]
v6: 6 [7]

Consider v4, v4 = 0, v1 = 10, v3 = 2, v6 = 7
Consider v3 (2), v0=2+1, v2=2+2, v5=2+8, v6=2+4
Consider v0 (3), v2=3=4
Consider v2 (7).
Consider v6 (6), v5=6+1
Consider v5 (10), v2=12
Consider v1 (10), v0=12, v3=13

b). [5pts] List all the possible topological orderings of the graph above.

[v4, v1, v3, v0, v6, v5, v2]

[v4, v1, v3, v6, v5, v0, v2]

[v4, v1, v3, v6, v5, v0, v2]