Quiz 2

- Do not open this quiz booklet until you are directed to do so. Read all the instructions on this page.
- When the quiz begins, write your name on every page of this quiz booklet.
- This quiz contains 5 problems, some with multiple parts. You have 120 minutes to earn 100 points.
- This quiz booklet contains 11 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your quiz at the end of the examination period.
- This quiz is closed book. You may use one $8\frac{1}{2}'' \times 11''$ crib sheet. No calculators or programmable devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem, since the pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

Problem	Parts	Points	Grade	Grader
1	5	25		
2	3	15		
3	1	20		
4	2	20		
5	4	20		
Total		100		

Name: _____

Circle the name of your recitation instructor:

Yoyo Zhou (10AM) Michael Lieberman (3PM)

Problem 1. True/False [25 points] (5 parts)

Decide whether each statement below is **True** or **False**. You must justify all your answers to receive full credit.

(a) There exists a comparison sort of 5 numbers that uses at most 6 comparisons in the worst case.
True **False**
Explain:

(b) Heapsort can be used as the auxiliary sorting routine in radix sort, because it operates in place.
True **False**
Explain:

(c) If the DFS finishing time $f[u] > f[v]$ for two vertices u and v in a directed graph G, and u and v are in the same DFS tree in the DFS forest, then u is an ancestor of v in the depth first tree.
True **False**
Explain:

(d) The sequence $\langle 20, 15, 18, 7, 9, 5, 12, 3, 6, 2\rangle$ is a max-heap.
True **False**
Explain:

(e) Let P be a shortest path from some vertex s to some other vertex t in a graph. If the weight of each edge in the graph is increased by one, P will still be a shortest path from s to t.

True **False**

Explain:

Problem 2. Short Answer [15 points] (3 parts)

(a) Where in a max-heap can the smallest element reside, assuming all elements are distinct? Include both the location in the array and the location in the implicit tree structure.

(b) What property of the Rubik's cube graph made 2-way BFS more efficient than ordinary BFS?

(c) What is the running time of the most efficient deterministic algorithm you know for finding the shortest path between two vertices in a directed graph, where the weights of all edges are equal? (Include the name of the algorithm.)

Problem 3. Topological Sort [20 points]

Another way of performing topological sorting on a directed acyclic graph $G = (V, E)$ is to repeatedly find a vertex of in-degree 0 (no incoming edges), output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $O(V + E)$. What happens to this algorithm if G has cycles?

Problem 4. Shortest Paths [20 points] (2 parts)

Carrie Careful has hired Lazy Lazarus to help her compute single-source shortest paths on a large graph. Lazy writes a subroutine that, given $G = (V, E)$, a source vertex s, and a non-negative edge-weight function $w : E \to R$, outputs a mapping $d : V \to R$ such that $d[v]$ is supposed to be the weight $\delta(s, v)$ of the shortest-weight path from s to v (or ∞ if no such $s \to v$ path exists) and also a function $\pi : V \to (V \cup \{NIL\})$ such that $\pi[v]$ is the penultimate vertex on one such shortest path (or NIL if $v = s$ or v is unreachable from s).

Carrie doesn't trust Lazarus very much, and wants to write a "checker" routine that checks the output of Lazarus's code (in some way that is more efficient than just recomputing the answer herself).

Carrie writes a "checker" routine that checks the following conditions. (No need for her to check that $w(u, v)$ is always non-negative, since she creates this herself to pass to Lazarus.)

(i) $d[s] = 0$

(ii) $\pi[s] = NIL$

(iii) for all edges $(u, v) : d[v] \leq d[u] + w(u, v)$

(iv) for all vertices v : if $\pi[v] \neq NIL$, then $d[v] = d[\pi[v]] + w(\pi[v], v)$

(v) for all vertices $v \neq s$: if $d[v] < \infty$, then $\pi[v] \neq NIL$ (equivalently: $\pi[v] = NIL \implies d[v] = \infty$)

(a) Show, by means of an example, that Carrie's conditions are not sufficient. That is, Lazarus's code could output some d, π values that satisfy Carrie's checker but for which $d[v] \neq \delta(s, v)$ for some v. (Hint: cyclic π values; unreachable vertices.)

(b) How would you augment Carrie's checker to fix the problem you identified in (a)?

Problem 5. Sorting [20 points] (4 parts)

For each set of data, match the sorting algorithm (that we went over in lecture or recitation) to the appropriate set. You may only use each algorithm once, so be sure that you are picking the best possible algorithm for that set. Not all algorithms will be used. You must justify your responses in one or two short sentences, and answers without justification will not be given credit. You may not assume anything about the sets beyond what you are given.

Insertion Sort	Counting Sort	Radix Sort
Bucket Sort	Merge Sort	

(a) Sorting 24,000,000 evenly distributed real numbers between 1 and 6,006.
Best algorithm for this set:
Explain:

(b) Sorting an array of 32,000,000 integers between 0 and 32,000,000.
Best algorithm for this set:
Explain:

(c) Independently sorting each of 1,000,000 arrays, each with 5 elements.
Best algorithm for this set:
Explain:

(d) Sorting a set of 4,000,000 numbers in worst case $O(n \lg n)$ time.
Best algorithm for this set:
Explain: