Quiz 2 Solutions

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:

Solution: **False.** The number of leaves of a decision tree which sorts 5 numbers is $5!$ and the height of the tree is at least $\lg(5!)$. Since $5! = 120$, $2^6 = 64$, and $2^7 = 128$, we have $6 < \lg(5!) < 7$. Thus at least 7 comparisons are required.

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

Solution: **False.** The auxiliary sorting routine in radix sort needs to be stable, meaning that numbers with the same value appear in the output array in the same order as they do appear in the input array. Heapsort is not stable. It does operate in place, meaning that only a constant number of elements of the input array are ever stored outside the array.

(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:

Solution: **False.** In a graph with three nodes, r u and v, with edges (r, u) and (r, v), and r is the starting point for the DFS, u and v are siblings in the DFS tree, neither as the ancestor of the other.

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

Solution: **True.** For every node with 1-based index $i > 1$, the node with index $\lfloor \frac{i}{2} \rfloor$ is larger.

(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:

Solution: **False.** In a graph where $w(s, v_1) = 1$, $w(v_1, v_2) = 1$, $w(v_2, t) = 1$, and $w(s, t) = 4$, the shortest path would change if 1 was added to every edge weight.

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.

Solution: The smallest element will be a leaf (because if it had a child, that child would have to be smaller). As seen in the quiz review, the leaves are the nodes indexed by $\lfloor \frac{n}{2} \rfloor + 1, \ldots, n$.

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

Solution: The number of nodes at most 7 away from any source was significantly less than the number of nodes at most 14 away from that source.

(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.)

Solution: Run BFS, treating it as an unweighted graph. This takes $O(V + E)$ time.

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?

Solution: First, for each vertex, we need to count how many incoming edges it has. We store these counts in a dictionary keyed by the vertices. This takes $\Theta(V + E)$ time.

For each vertex with 0 incoming edges, store it in a special dictionary. Call that dictionary `zero`.

We repeatedly do the following: Take a vertex out of `zero`. For every edge coming out of that vertex, decrement the count of that edge's target vertex. If you happen to decrement a count to 0, add that vertex to `zero`.

This touches every vertex once and every edge once, so it also takes $\Theta(V + E)$ time.

If there is a cycle, at some time when we try to take a vertex out of `zero`, we won't find any.

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.)

Solution: The following graph is a counterexample to Carrie's checker:
s has no edges, $\pi[s] = NIL$, and $d[s] = 0$
u and v have edges of weight 0 to each other, $\pi[u] = v$, $\pi[v] = u$, $d[u] = 0$, and $d[v] = 0$.
Instead, it should be the case that $d[u] = d[v] = \infty$, and $\pi[u] = \pi[v] = NIL$.

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

Solution: In addition to the given checks, Carrie should also do the following:

1. Check that $\pi[v] \neq NIL \implies (\pi[v], v) \in E$.
2. Run DFS on G from source S. For each unreachable vertex v, check that $d[v] = \infty$ and $\pi[v] = NIL$.
3. Run DFS on $G' = (V, E')$, where $E' = \{(v, \pi[v]) : \pi[v] \neq NIL\}$ to check that G' is acyclic.

These checks take time $\Theta(V + E)$.

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:

Solution: Bucket sort, the input is uniformly distributed, so Bucket Sort runs in linear time.

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

Solution: Counting sort or Radix Sort, both run in linear time on the bounded input.

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

Solution: Insertion sort, due to the very low constant factors involved in running it on small inputs.

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

Solution: Merge Sort, as it has the required running time.