Quiz 2 Practice Problems

1 True/False

Decide whether these statements are True or False. You must briefly justify all your answers to receive full credit.

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

2. Heapsort can be used as the auxiliary sorting routine in radix sort, because it operates in place.
   True  False
   Explain:
3. 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:

4. 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:

5. If an in-place sorting algorithm is given a sorted array, it will always output an unchanged array.
   True  False

   Explain:
6. [5 points] Dijkstra’s algorithm works on any graph without negative weight cycles.
   True    False

   Explain:

   True    False

   Explain:
2 Short Answer

1. What property of the Rubik’s cube graph made 2-way BFS more efficient than ordinary BFS?

2. 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.)

3 Topological Sort

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?
4 Shortest Paths

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 $w : E \to \mathbb{R}$, outputs a mapping $d : V \to \mathbb{R}$ such that $d[v]$ is supposed to be the weight $\delta(s, v)$ of the shortest-weight path from $s$ to $v$ (or $\infty$ 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 $v : \text{if } \pi[v] \neq NIL, \text{ then } d[v] = d[\pi[v]] + w(\pi[v], v)$

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

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

2. How would you augment Carrie’s checker to fix the problem you identified in (a)?
3. You are given a connected weighted undirected graph $G = (V, E, w)$ with no negative weight cycles. The **diameter** of the graph is defined to be the maximum-weight shortest path in the graph, i.e. for every pair of nodes $(u, v)$ there is some shortest path weight $\delta(u, v)$, and the diameter is defined to be $\max_{(u,v)} \{\delta(u, v)\}$.

Give a polynomial-time algorithm to find the diameter of $G$. What is its running time? (Your algorithm only needs to have a running time polynomial in $|E|$ and $|V|$ to receive full credit; don’t worry about optimizing your algorithm.)

4. You are given a weighted directed graph $G = (V, E, w)$ and the shortest path distances $\delta(s, u)$ from a source vertex $s$ to every other vertex in $G$. However, you are not given $\pi(u)$ (the predecessor pointers). With this information, give an algorithm to find a shortest path from $s$ to a given vertex $t$ in $O(V + E)$ time.