CMSC 451 Quiz 1

This quiz is closed-book and closed-notes. You may use any algorithms or results given in class. The total point value is 50 points. Good luck!

Problem 1. (9 points) In this problem, use the definition of graph, digraph, and DAG as given in lecture. Explanations are *not* required, but may be given for partial credit.

(a) (3 points) Which of the following is the *maximum* number of edges in an *undirected graph* with n vertices.

(i) n
(ii) n^2
(iii) $n(n-1)$
(iv) $n(n-1)/2$ (or equivalently $\binom{n}{2}$)
(v) None of the above

(b) (3 points) Repeat (a) for a *directed graph* with n vertices.

(c) (3 points) Repeat (a) for a *directed acyclic graph* (DAG) with n vertices.

Problem 2. (15 points – 5 each) For each of the lists below, put the functions in increasing asymptotic order. When two functions are asymptotically equivalent, indicate this. (Explanations are not needed, but may be given for partial credit.)

(a)	n^2	$n \log n$	$n\sqrt{n}$
(b)	$n^{\lg 2}$	$2^{\lg n}$	$2^{(2 \lg n)}$
(c)	$n \log(\log n)$	$n(\log n)^2$	$n \log(n^2)$

For example, given n^2, n^3, and $n + n^2$, the answer would be $n^2 \approx n + n^2 \prec n^3$.

Problem 3. (6 points) For this problem, assume the version of the Bellman-Ford algorithm given in class (which terminates when distance values converge). Suppose that you are running the Bellman-Ford algorithm on a *directed acyclic graph* (DAG). Which of the following hold?

(i) It is generally more efficient to process vertices/edges in (*forward*) *topological order*.

(ii) It is generally more efficient to process vertices/edges in *reverse topological order*.

(iii) It is generally more efficient to process *negative weight edges* first.

(iv) None of the above. (The efficiency is not better for any of the above orderings.)

Briefly justify your answer. (A couple of sentences. Not a formal proof.)

Problem 4. (10 points)

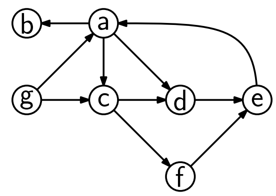


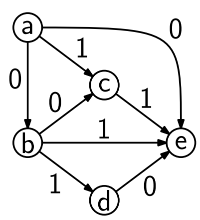
Figure 1: Depth-first search.

(a) (7 points) Show the result of running DFS on the digraph in Fig. 1 using the algorithm given in class. (Start at vertex a. Whenever you have a choice of which vertex to visit next, take the lowest vertex in alphabetical order.)

Label each node u with its discovery and finish times ($d[u]/f[u]$). As in the lecture notes, show tree edges with solid lines and the other edges with dashed lines. Label these other edges as *forward*, *cross*, or *back* edges. (Show only the **final tree**, not the intermediate results.)

(b) (3 points) What are the strong components of this digraph? (No explanation needed. Just list the vertices appearing in each strong component.)

Problem 5. (10 points) You are given a DAG (directed acyclic graph) $G = (V, E)$ in which each edge has a label, label(u, v), which is either 0 or 1. A *1-path* is defined to be any path that has exactly one edge with the label 1 (anywhere along the path), and all other edges have label 0. For each vertex $u \in V$, let one(u) denote the number of 1-paths that start at u (see Fig. 2).



u	one(u)	
a	5	$\longrightarrow \langle a, b, c, e\rangle, \langle a, b, d\rangle, \langle a, b, d, e\rangle, \langle a, b, e\rangle, \langle a, c\rangle$
b	4	$\longrightarrow \langle b, c, e\rangle, \langle b, d\rangle, \langle b, d, e\rangle, \langle b, e\rangle$
c	1	$\longrightarrow \langle c, e\rangle$
d	0	
e	0	

Figure 2: 1-paths in a DAG.

Present an efficient algorithm which, given any DAG $G = (V, E)$ with 0-1 edge labels, computes one(u) for all $u \in V$. (Just the counts, not the actual paths.)

Briefly justify your algorithm's correctness and derive its running time. Ideally, your algorithm should run in $O(n + m)$ time. **Hint:** Use a DFS approach, similar to the alternating paths problem on the practice set. If you are stuck, for half credit, instead count the number of paths that have *at least* one edge with label 1. Be sure to indicate that you are going for partial credit.