Homework No. 4 Answers

CSCI 4470/6470 Algorithms, CS@UGA, Spring 2017

Due Monday March 27, 2017

There are 100 points in total.

1. (15 points) Show how the algorithm RECURSIVE DFS (along with TO-START-DFS) in the Lecture Note 4 works on the graph figure shown below. Assume for the graph, the adjacency list of every vertex is ordered alphabetically. Assume that the for loop of lines 4-5 of algorithm TO-START-DFS considers the vertices in the alphabetical order. Show the resulted depth-first-search tree (or forest).

Answers: Note that this is an algorithm different from the DFS in the text. There is no discover time nor finish time computed for each vertex.

Points are awarded proportionally to how much the search forest is corrected.
2. (25 points) Consider the following Constrained Reachability problem: the input consists of a directed graph $G(V,E)$, two vertices $s,t \in V$, and a subset $U \subseteq V$, where $U \cap \{s,t\} = \emptyset$. The output is “yes” if and only if there is a path from $s$ to $t$ in the graph without using any vertex in $U$. Apparently, the problem is the commonly known Reachability problem when $U = \emptyset$.

(1) Design a recursive DFS type of algorithm to Constrained Reachability solve problem. You may assume that the input graph is already stored in adjacency lists and $v.\mu$ indicates if vertex $v$ is in the set $U$ or not. The information about $\mu$ is also given along with the graph.

(2) Argue that your algorithm still runs in linear time (i.e., $O(|E| + |V|)$-time) though the set $U$ may contain a large number of vertices.

Answers:

(1) (10 points) The student can/should simply modify the existing DFS algorithm to suit the need of this problem. In particular, there are mainly modifications: (1) to avoid a vertex within the set $U$, which is indicated by $v.\mu = TRUE$; that is, never search along edge $(u,v)$ when $v.\mu = TRUE$. (2) If $v = t$ then search should stop (via the return statement).

(2) (5 points) Since $v.\mu$ values are given as a part of the input, there is no needed to search through the set $U$ to identify such values. So the complexity remains the same.
3. **(20 points)** It is known that depth-first-search strategy can be used to determine if a given undirected graph contains a cycle. Argue that the algorithm runs in time $O(|V|)$, independent of $|E|$ before it answers the question correctly. Note that you need to prove that time $O(|V|)$ is enough for the algorithm to answer both “yes” and ”no”.

**Answers:**

Note that this question does not require student to redesign a depth-first-search algorithm. But he/she needs to know that a cycle can be detected by identifying a back edge (i.e., when edge $(u,v)$ is being explored and $v.color$ is GRAY/BLACK).

**(10 points)** Case 1: when the answer is ”yes”. Note that any cycle contains at most $n = |V|$ vertices. Before hitting the first GRAY vertex (“old” vertex), the search algorithm has visited all “new” vertices, whose number is $\leq n$. So it terminates with $O(|V|)$ time.

**(10 points)** Case 2: when the answer is ”no”. If there is no cycle in the graph, it is a tree/forest. There will be at most $|V| - 1$ edges. So the algorithm will stop in at most $O(|E| + |V|) = O(|V|)$ steps.
4. **(15 points)** Show how the procedure **Strongly Connected Components** (page page 617) works on the graph given for question 1. Specifically show the finishing times computed in line 1 and the forest produced in line 3. Assume the **for** loop of lines 5-7 of **DFS** (page 604) considers vertices in alphabetical order and that the vertices in each adjacency list are in alphabetical order.

**Answers:** Points are awarded according to how much the answers are correct. **5 points** for finish times; **10 points** for the forest.
5. **(25 points)** Prove that the Topological Sorting problem can be solved with the reversed order of finished time produced by the DFS algorithm. Note that you will need to argue that, for any two vertices \( u \) and \( v \) in the graph, \( u.f > v.f \) if and only if there is a directed path from \( u \) to \( v \).

**Answers:**

The student needs to prove that

for any two vertices \( u \) and \( v \) for which the sort order is important, \( u.f > v.f \) if and only if there is a directed path from \( u \) to \( v \);  

**Proof ideas:**

Since the sort order is important to \( u \) and \( v \), there must be a path from \( u \) to \( v \) or the other way around.

10 points) Assume \( u.f > v.f \). Then either \( u \) is an ancestor of \( v \) in the DFS tree, which means there is a directed path from \( u \) to \( v \), or \( u \) is finished after \( v \) is finished. Because \( u \) cannot be a descendent of \( v \) in the DFS tree, there must be a directed path from \( u \) to \( v \).

On the other hand, if there is a directed path from \( u \) to \( v \), there are two possibilities:

(a) 10 points) \( u \) is searched before \( v \). Then \( u \) is an ancestor of \( v \). So \( u.f > v.f \);

(b) 5 points) \( v \) is searched before \( v \). Because there is no directed path from \( v \) to \( u \), \( u \) will be discovered after \( v \) is finished. So \( u.f > v.f \)