COMP 3804 — Assignment 3

Due: Sunday March 28, 23:59.

Assignment Policy:

- Your assignment must be submitted as one single PDF file through cuLearn.

Use the following format to name your file:

LastName_StudentId_a3.pdf

- Late assignments will not be accepted. I will not reply to emails of the type “my internet connection broke down at 23:57” or “my scanner stopped working at 23:58”, or “my dog ate my laptop charger”.

- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.

- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams.

- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Question 1: Write your name and student number.

Question 2: Let $G = (V, E)$ be an undirected graph, and let $e$ be an edge in $E$. Describe an algorithm that decides, in $O(|V| + |E|)$ time, whether $G$ has a cycle containing the edge $e$. Justify your answer. You may use any result that was presented in class.

Question 3: Consider the following directed graph:

![Directed Graph](attachment:image.png)
(3.1) Draw the DFS-forest obtained by running algorithm DFS; the pseudocode is at the end of this assignment. Algorithm DFS uses algorithm EXPLORE as a subroutine. The pseudocode for this subroutine is also given at the end of this assignment.

Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the DFS-forest, give the pre- and post-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically first.

(3.2) Draw the DFS-forest obtained by running algorithm DFS. Classify each edge as a tree edge, forward edge, back edge, or cross edge. In the DFS-forest, give the pre- and post-number of each vertex. Whenever there is a choice of vertices, pick the one that is alphabetically last.

**Question 4:** In order to increase revenue, the owner of Michiel’s Taxi Company introduces the policy that all taxi drivers must always take the longest path in the directed graph $G = (V, E)$ representing the roads of Ottawa. For each directed edge $(u, v)$ in $E$, let $wt(u, v) > 0$ denote the length of this edge.

The following approach is suggested to compute the length of the longest path from a source vertex $s$ to each vertex of $V$: For each directed edge $(u, v)$ in $E$, define $wt'(u, v) = -wt(u, v)$. Run a shortest-path algorithm using the new weights $wt'$. In the output, replace each shortest-path length $L$ by $-L$.

In both of the following two parts of this question, assume that the directed graph $G$ is acyclic.

(4.1) Prove or disprove: Algorithm SHORTESTPATHACYCLIC, which we have seen in class, correctly computes, for every vertex $v$, the length of the longest path from $s$ to $v$. (Note: You have to run the algorithm exactly as given at the end of this assignment.)

(4.2) Prove or disprove: Dijkstra’s algorithm correctly computes, for every vertex $v$, the length of the longest path from $s$ to $v$. (Note: You have to run Dijkstra’s algorithm exactly as given at the end of this assignment. After a vertex $v$ has been deleted from the set $Q$, it may happen that the value of $d(v)$ gets decreased later in the algorithm.)

**Question 5:** Let $G = (V, E)$ be a connected undirected graph in which each edge has a positive weight. You may assume that no two edges have the same weight.

(5.1) Prove or disprove: The edge with the second smallest weight is an edge in the minimum spanning tree of $G$.

(5.2) Prove or disprove: The edge with the third smallest weight is an edge in the minimum spanning tree of $G$. 
**Question 6:** In class, we have seen a data structure for the Union – Find problem that stores each set in a linked list, with the header of the list storing the name and size of the set. Using this data structure, any operation Find(x) takes $O(1)$ time, whereas any operation Union(A, B, C) takes $O(\min(|A|, |B|))$ time.

Consider the same data structure, except that the header of each list only stores the name of the set (and not the size). Show that, in this new data structure, any operation Find(x) can be performed in $O(1)$ time, and any operation Union(A, B, C) can still be performed in $O(\min(|A|, |B|))$ time.

**Question 7:** You are given two binary strings $A = a_1a_2\ldots a_m$ and $B = b_1b_2\ldots b_n$. A common substring of $A$ and $B$ is a binary string that occurs consecutively both in $A$ and in $B$.

Give a dynamic programming algorithm (in pseudocode) that computes, in $O(mn)$ time, a longest common substring of $A$ and $B$. Argue why your algorithm is correct.

For example, for the input strings $A = 0001010111$ and $B = 01010101000$, the output will be the string 010101.

Hint: Define $L[i, j]$ to be the length of a longest common substring of $A$ and $B$ that ends at $a_i$ and $b_j$. Observe that $L[i, j] = 0$ if $a_i \neq b_j$. 

3
Algorithm DFS($G$):
for each vertex $v$
do $visited(v) = false$
endfor;
clock = 1;
for each vertex $v$
do if $visited(v) = false$
    then $Explore(v)$
    endif
endfor

Algorithm $Explore(v)$:
$visited(v) = true$;
$pre(v) = clock$;
clock = clock + 1;
for each edge $(v, u)$
do if $visited(u) = false$
    then $Explore(u)$
    endif
endfor;
$post(v) = clock$;
clock = clock + 1
**Algorithm** ShortestPathAcyclic\((G, s, wt')\):
topologically sort \(G\);  
denote the resulting numbering of the vertices by \(v_1, v_2, \ldots, v_n\);  
asume that \(s = v_1\);  
\(d(s) = 0\);  
for \(i = 2\) to \(n\)  
do \(d(v_i) = \infty\);  
endfor;  
for \(i = 1\) to \(n\)  
do \(u = v_i\);  
for each edge \((u, v)\)  
do if \(d(u) + wt'(u, v) < d(v)\)  
then \(d(v) = d(u) + wt'(u, v)\)  
endif  
endfor  
endfor

**Algorithm** Dijkstra\((G, s, wt')\):
for each \(v \in V\)  
do \(d(v) = \infty\)  
endfor;  
\(d(s) = 0\);  
\(S = \emptyset\);  
\(Q = V\);  
while \(Q \neq \emptyset\)  
do \(u = \) vertex in \(Q\) for which \(d(u)\) is minimum;  
delete \(u\) from \(Q\);  
insert \(u\) into \(S\);  
for each edge \((u, v)\)  
do if \(d(u) + wt'(u, v) < d(v)\)  
then \(d(v) = d(u) + wt'(u, v)\)  
endif  
endfor  
endfor