COMP 3804 — Assignment 3

Due: Monday March 28, 23:59.

Assignment Policy:

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

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:** Consider the following directed graph:

![Directed Graph](image)

(2.1) Draw the DFS-forest obtained by running algorithm DFS; the pseudocode is given 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.

(2.2) 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 3:** Let $G = (V, E)$ be a directed graph, in which each edge $(u, v)$ has a positive weight $wt(u, v)$, and let $s$ be a source vertex in $V$. Let $n = |V|$ and $m = |E|$.

Recall that Dijkstra’s algorithm computes for each vertex $v$, the length $\delta(s, v)$ of a shortest path from $s$ to $v$, in total time $O((n + m) \log n)$. The pseudocode for this algorithm is given at the end of this assignment.

Assume that each edge weight $wt(u, v)$ is an integer in the set $\{1, 2, \ldots, 17\}$. Prove that Dijkstra’s algorithm can be implemented such that the running time is $O(m + n)$.

*Hint:* A priority queue is a data structure that stores any finite sequence of numbers and supports the operations INSERT, EXTRACT_MIN and DECREASE_KEY. An example of a priority queue is a min-heap.

A priority queue is called awesome if, during any sequence of operations, the smallest element never decreases. Thus, if at some moment, the smallest element is 45, then later on, the smallest value is always at least 45.

Assume that at any moment, any number stored in an awesome priority queue is an integer belonging to the set $\{0, 1, 2, \ldots, k\}$. Start by showing that any sequence consisting of $n$ INSERT-operations, $n$ EXTRACT_MIN-operations and $m$ DECREASE_KEY-operations (these operations may appear in any order) can be processed in total time $O(m + n + k)$. 

2
Question 4: Lionel Messi is having a difficult time. Not only has his team been kicked out of the Champions League, he is even [booed] by his own fans. Lionel decides to hang up his boots and become a software developer.

On the first day of his new job, Lionel is asked to implement a sorting algorithm, i.e., an algorithm that takes as input an arbitrary sequence of numbers and returns these numbers in sorted order.

Since Lionel has no clue about algorithms, he looks at the cheater website [chegg.com]. Unfortunately, this website does not have implementations of sorting algorithms. However, Lionel does find a highly optimized implementation of Dijkstra’s algorithm. For any directed input graph $G = (V, E)$, in which each directed edge $(u, v)$ has a weight $wt(u, v) > 0$, and for any given source vertex $s$, this algorithm computes for each vertex $v$, the length $\delta(s, v)$ of a shortest path from $s$ to $v$, in total time $O((|V| + |E|) \log |V|)$.

Prove that Lionel can use this implementation of Dijkstra’s algorithm to sort any sequence of $n$ numbers in $O(n \log n)$ time.

Hint: Given a sequence $x_1, x_2, \ldots, x_n$ of numbers, define a (very simple!) directed graph $G$ with positive edge weights. Run Dijkstra’s algorithm on this graph.

Question 5: Let $G = (V, E)$ be a connected undirected graph, in which each edge has a weight. An edge $\{u, v\}$ is called annoying if the graph $G' = (V, E\setminus\{\{u, v\}\})$ is not connected.

Assume there is a unique edge in $E$ with largest weight; denote this edge by $e$.

Prove that $e$ is an edge in every minimum spanning tree of $G$ if and only the edge $e$ is annoying.

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, and each node storing a back pointer to the header. If we start with $n$ sets, each having size one, then we have seen in class that, using this data structure, any sequence of $n - 1$ UNION-operations can be processed in $O(n \log n)$ time.

Give an example of a sequence of $n - 1$ UNION-operations, for which the algorithm takes $\Omega(n \log n)$ time.

Question 7: You are given two strings $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ over some finite alphabet. We consider the problem of converting $X$ to $Y$, using the following operations:

1. Substitution: replace one symbol by another one.
2. Insertion: insert one symbol.
3. Deletion: delete one symbol.

For example, if $X =$ “logarithm” and $Y =$ “algorithm”, we can convert $X$ to $Y$ in the following way:

1 “We offer you a salary of half a million”. “That’s reasonable. I assume this is per week?”
1. Start with “logarithm”.

2. Inserting “a” at the front gives “allogarithm”.

3. Deleting “o” gives “algarithm”.

4. Replacing the second “a” by “o” gives “algorithm”.

The similarity between the strings $X$ and $Y$ is defined to be the minimum number of operations needed to convert $X$ to $Y$. For example, the similarity between $X = “logarithm”$ and $Y = “algorithm”$ is three, because $X$ can be converted to $Y$ using three operations, but not using two operations. If the string $X$ has length $m$ and the string $Y$ is empty, then the similarity between $X$ and $Y$ is equal to $m$.

Give a dynamic programming algorithm (in pseudocode) that computes, in $O(mn)$ time, the similarity between the strings $X$ and $Y$. Argue why your algorithm is correct.
Algorithm DFS($G$):
for each vertex $v$
do $\text{visited}(v) = \text{false}$
endfor;
$\text{clock} = 1$;
for each vertex $v$
do if $\text{visited}(v) = \text{false}$
then Explore($v$)
endif
endfor

Algorithm Explore($v$):
$\text{visited}(v) = \text{true}$;
$\text{pre}(v) = \text{clock}$;
$\text{clock} = \text{clock} + 1$;
for each edge $(v, u)$
do if $\text{visited}(u) = \text{false}$
then Explore($u$)
endif
endfor;
$\text{post}(v) = \text{clock}$;
$\text{clock} = \text{clock} + 1$

Algorithm Dijkstra($G, s$):
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 $d(u) + \text{wt}(u, v) < d(v)$
then $d(v) = d(u) + \text{wt}(u, v)$
endif
endfor
endwhile