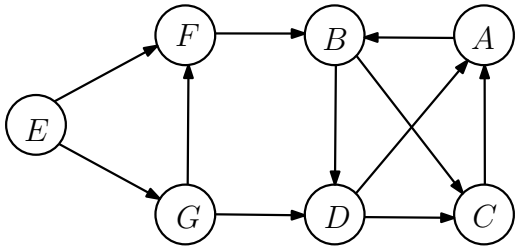
COMP 3804 — Tutorial March 15

Question 1: Consider the following directed graph:



(1.1) Draw the *DFS*-forest obtained by running algorithm DFS; the pseudocode is given on the last page. Algorithm DFS uses algorithm EXPLORE as a subroutine. The pseudocode for this subroutine is also given on the last page.

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.

(1.2) 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 2: 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 on the last page.

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 *monotone* 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 a monotone 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 $O(m + n + k)$.

Question 3: Lionel Messi[1] 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[2].

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 4: 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 5: 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 6: You are given two strings $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ 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] When I made this question, Messi was still playing in Paris.

[2] "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 “alogarithm”.

3. Deleting “o” gives “algarithm”.

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

The *edit distance* between the strings X and Y is defined to be the minimum number of operations needed to convert X to Y. For example, the edit distance 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 edit distance between X and Y is equal to m.

Give a dynamic programming algorithm (in pseudocode) that computes, in $O(mn)$ time, the edit distance between the strings X and Y. Argue why your algorithm is correct.

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 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 $d(u) + wt(u, v) < d(v)$
        then $d(v) = d(u) + wt(u, v)$
        endif
    endfor
endwhile