COMP 3804/MATH 3804
Design and Analysis of Algorithms
Assignment 3

Due Date: November 19th at 11:59PM
Your assignment should be submitted online on Brightspace as a single .pdf file. The filename should contain your name and student number. No late assignments will be accepted.You can type your assignment or you can upload a scanned copy of it. Please, use a good image capturing device. Make sure that your upload is clearly readable. If it is difficult to read, it will not be graded.

Question 1:[20 points]
We are given a directed graph $G = (V, E)$ with $|V| = n$ vertices. Let $goal$ be a vertex of G. We want to compute a shortest path from each of k vertices of G to $goal$, where $k < n$.

- We could solve the problem by applying Dijkstra's algorithm k times, ones for each of the k starting vertices. What is the time complexity (stated in terms of n and k)?
- Alternately, we could start at the vertex $goal$ and somehow go backwards to all k vertices. Describe how this would work, i.e., how would we modify Dijkstra's algorithm and/or its input to achieve this? Then, state the time complexity of this solution to our original problem. (Do not forget to argue why the algorithm, as modified, is correct!)

Question 2:[15 points]

Let $G = (V, E)$ be a graph with vertex set, V, and edge set E. We would like to apply Topological Sort on G. One problem is that we do not know if G is a DAG or not. What will happen if we apply the algorithm for Topological Sorting on G if G is not a DAG?

Question 3:[15 points]

Suppose we consider lattice paths from $(0, 0)$ to (n, n) on an n by n grid. The paths must, at every step, either go up or right. We call lattice path, k-Lpaths, if they have precisely $2k$ path segments on one side of the diagonal and the remaining $2(n - k)$ segments on the other. Argue precisely why the number of k-Lpaths is equal to the number of $(n - k)$-Lpaths.

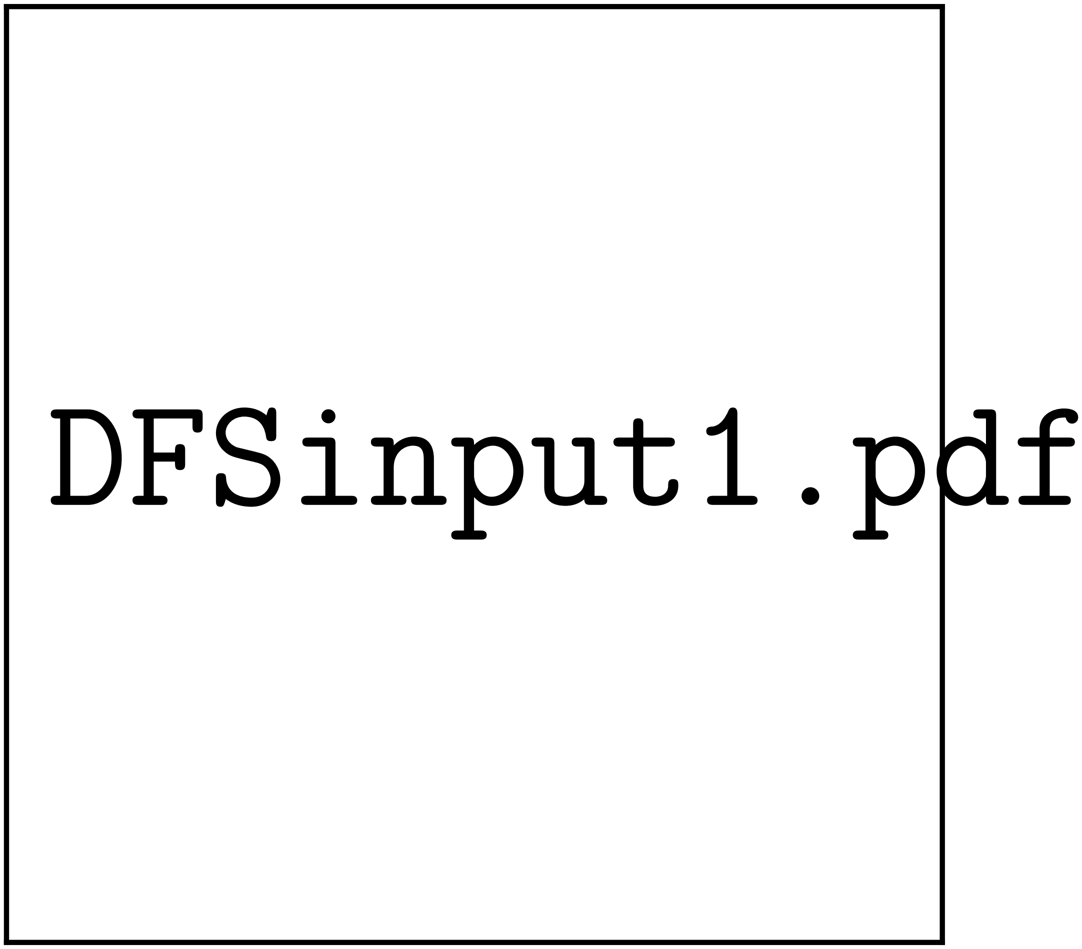

Figure 1: Input for DFS algorithm

Question 4:[15 points]
Consider the graph given in Figure 1 above.

- Run DFS, from A, on the graph and classify each edge as being either: Tree edge, Forward edge, Back edge, or Cross edge. Show and argue: the algorithm execution, pre(v) and post(v) time intervals and the edge-classification. (An edge type may or may not appear in a particular graph.)

- Find a topological order of the nodes or argue that no such order can exist. How does the DFS help detect that?

- Consider two intervals $[pre(u), post(u)]$ and $[pre(v), post(v)]$ for vertices u and v, respectively. Argue precisely in your own words, why the intervals cannot overlap (other than if one is contained in the other).

.