ADS 1 — problem sheet 4

Exercise 1: How to determine if given graph has precisely one topological order?

Solution: From lecture we know that the graph G has a topological order if and only if it is a directed acyclic graph.

Let $\prec$ be a topological order. We say that two vertices $a \prec b$ are *consecutive* if and only if there is no vertex c such that $a \prec c \prec b$. If there are two consecutive vertices a and b not connected by an edge then $\prec$ is not unique, because we can swap them and resulting order is topological again.

Conversely if every pair of consecutive vertices is connected by an edge, then $\prec$ is unique: one can check that there is only one source in G and that is the unique vertex that can be first in a topological order. After removing it we have again only one source, which is the unique vertex that can be second and so on.

Exercise 2: Design $O(n+m)$ algorithm for topological sorting that is based on the idea of repeated removal of sources.

Solution: Given graph $G = (V, E)$ to the following

1. For every vertex v calculate $i(v)$ to be the number of vertices w such that $(w, v) \in E$ (that the in-degree of v)
2. Create queue Q and put there all vertices such that $i(v) = 0$.
3. Create empty queue O.
4. Until Q is non-empty repeat:
   (a) Dequeue vertex v from Q.
   (b) Enqueue vertex v to O.
   (c) For every w such that $(v, w) \in E$ decrease $i(w)$ and if $i(w)$ is 0 then add w to Q

This algorithm can be easily checked to run in $O(n+m)$. If the input graph G is a DAG the queue O, once algorithm terminates, will contain a topological order: this can be verified by observing that Q is first initialised by all sources of G and then decreasing $i(w)$ corresponds to "removing" vertex form G and updating the in-degrees.

Exercise 3: Describe algorithm to find a shortest path in a edge-labelled DAG which makes use of the topological ordering of vertices. *Edge-labelled DAG* an acyclic oriented graph $G = (V, E)$ where every edge $e \in E$ has associated *length* $\ell(e)$. The *length* of a given path is then sum of lengths of all edges in it. On next class we will study an algorithm to answer this question on graphs in general.

Solution: This is similar to the problem of counting number of paths discussed on the class. Given graph $G = (V, E)$ and vertex v_0 first put $h(v_0) = 0$. Then process vertices in the topological order. All vertices before v_0 are not reachable from v_0 and thus we can put $h(v_0) = \infty$ for other vertices we can put

$$h(v) = \min_{w,(w,v)\in E} h(w) + \ell(w, v).$$

At the end of the algorithm $h(v)$ will contain distances form v_0. As usual we can also compute predecessors array which will identify a shortest path.

Exercise 4: Do the same to find the longest path in an edge-labelled DAG (this is a problem we do now know how to solve effectively on general graphs).

Solution: This is the same algorithm replacing min with max

Exercise 5: We say that connected graph $G = (V, E)$ is *semi-connected* if for every pair of vertices $u, v \in E$ there exists an oriented path from u to v or from v to u (possibly both). Design linear-time algorithm to decide if given graph G is semi-connected.

Solution: Observe that if G is DAG then it is semi-connected if and only if it has unique topological order. We know how to solve this problem. To solve the problem for general graphs, first construct component graph using the algorithm from the class and then apply Exercise 1.

Homework 2: Construction of a house consists of multiple tasks where some needs to be finished before others can start. We describe this as a graph: vertices are tasks and each vertex has associated duration. Oriented edges represents dependencies. Write algorithm to determine shortest time in which the house can be constructed (assuming there is unlimited number of workers doing individual tasks) and find critical ones (that is, tasks whose delays would result in longer overall construction times).