3 Topological sort (p. 15)

3.1 Topological searches (p. 16)

1. Let t be a node with out-degree 0 in a dag $G = (V, E)$. Give a $O(|V| + |E|)$ algorithm to determine the number of paths from each vertex $u \in V \setminus \{t\}$ to t.

Solution: The algorithm has the following steps:

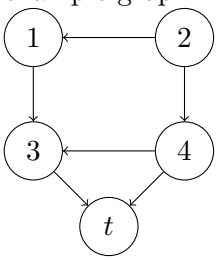
(a) Perform a topological sort on $G = (V, E)$ (in $O(|V| + |E|)$)
This places the nodes such that all edges point in the same direction.

(b) Visit all nodes in the reversed order of the above top. sort. Keep track of a counter c_u (initialized with 0) for each $u \in V$. Upon visiting u, compute

$$c_u = \sum_{v \in \Gamma(u)} c_v + 1[(u, t) \in E]$$

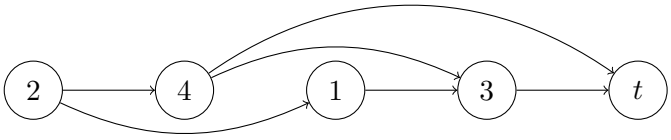
(Alternatively, we can also initialise $c_u = 0$ for every $u \in V \setminus \{t\}$ and $c_t = 1$, and compute $c_u = \sum_{v \in \Gamma(u)} c_v$ upon visiting u.)

(Note, that it doesn't matter where we start the topological sort.) Starting with an example graph:



There are 2 possible topological sorts of this graph: $2, 4, 1, 3, t$ and $2, 1, 4, 3, t$.

However, both of these result in the same number of paths. In this example we use first of the two sorts.



We visit every node from right to left in the topological sort and get the following counters (in this order):

- $c_3 = 0 + 1 = 1$
- $c_1 = 1 + 0 = 1$
- $c_4 = 1 + 1 = 2$
- $c_2 = 2 + 1 = 3$

2. Let s and t be two vertices in a dag G such that the in-degree of s and the out-degree of t is 0. Use the result of the previous exercise to determine, for each edge $(u, v) \in E$, the number of paths from s to t that run through (u, v) and this in $O(|V| + |E|)$ time.

Solution:

(a) Run the algorithm of the previous exercise (4.1.1) on G with t and keep track of the number of paths b_v from v to t.

(b) Reverse the edges and do the same for the number of paths a_u from u to s.

(c) Visit every edge $(u, v) \in E$ and compute the number of paths from s to t through (u, v) as

$$c_{(u,v)} = a_u \cdot b_v$$

as u can be reached from s in a_u ways and v has b_v ways of reaching t.

3. One can also sort a dag G by repeatedly selecting a vertex u with in-degree 0 and removing all its outgoing edges from G. How do you implement this in $O(|V| + |E|)$ time?

Solution: This is also known as **Kahn's algorithm**. We assume an adjacency list implementation.

The algorithm consists of the following steps:

(a) Create a list of nodes and their in-degrees.
This can be done in $O(|V| + |E|)$, e.g. by reversing edges and counting out-degree/adjacency list lengths for each node.

(b) G is a dag, so there must be at least 1 node with in-degree 0 (see exercise 5). Find these nodes (by scanning the list from step 1) and place them in a queue Q.

(c) Create a list $L = []$ and perform the following operations until Q is empty:

i. $u = \text{DEQUEUE}(Q)$
ii. $L = [L, u]$
iii. Delete all outgoing edges of u.
For each such deleted edge (u, v), decrease the in-degree counter of v.
If the in-degree of v is equal to 0, do $\text{ENQUEUE}(Q, v)$.

We can enqueue each node in V at most once and remove each (outgoing) edge at most once.
If we implement the "list" of nodes and their in-degrees as a (hash)map or an array (if nodes are enumerated with numbers in $\mathbb{N}$), the counter can be updated in $O(1)$.
This indicates the algorithm runs in $O(|V| + |E|)$ time.

4. Suppose that we execute the topological sort on a graph G that is not acyclic in order to sort the vertices of G. Show that the number of edges pointing in the wrong direction is not necessarily minimal using a graph with 4 vertices. ☆

5. Argue that every directed acyclic graph contains at least one vertex with zero outgoing edges. ☆

> **Solution:** Solution 1: Perform a topological sort on the graph. Then, by definition, the last vertex in the linked list has zero outgoing edges (and the first zero incoming edges).
> Solution 2: One can also show this without relying on topological sort but through a simple contraposition.

6. Given a weighted acyclic graph $G = (V, E)$. Give an $O(|V| + |E|)$ algorithm to find a path with maximum weight in G. ☆

> **Solution:** Note: This exercise is a out of place in the course as it is about Shortest paths in a DAG (p. 85).

Extra exercise: Suppose we are given L, a topologically sorted list of vertices of a DAG $G = (V, E)$. Let G' be the same graph but with reversed edges. Give an $O(|V|)$ algorithm to find a topological sort of G'.
Extra exercise: In this course you will only encounter graphs with finitely many vertices. We have seen in exercise 5 that in a dag there always exist vertices with out-degree 0. Does this still hold if we also consider graphs with infinitely many vertices?

4 Strongly Connected Components (p. 17)

4.1 Strongly Connected Components (p. 19)

1. What is the maximum number of edges in a directed graph $G = (V, E)$ that has exactly two SCCs? ☆

> **Solution:** Let m and n be the number of vertices in the first (C_1) and second (C_2) SCC respectively (with $m + n = |V|$). By definition, there exists a path between any two vertices inside a SCC. Therefore, the maximum number of edges inside C_1 and C_2 is $m(m-1)$ and $n(n-1)$ respectively: an edge from every vertex to every other vertex inside a SCC. The maximum number of edges between the two SCCs is mn: an edge from every vertex of C_1 to every vertex of C_2. In total, we have that the number of edges is
>
> $$\begin{aligned} m(m-1) + n(n-1) + mn &= m(m-1) + (|V| - m)(|V| - m - 1) + m(|V| - m) \\ &= m^2 - |V|\,m + |V|^2 - |V|\,. \end{aligned}$$

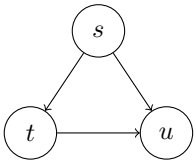
We now need to find an $m \in \{1, \ldots, |V| - 1\}$ for which the above expression is maximized. As the expression gives an upward parabola, it is maximized for $m = 1$ or $m = |V| - 1$, that is for the case where one of the SCCs is a single vertex. We then have $|V|^2 - 2|V| + 1 = (|V| - 1)^2$ edges.

2. Determine the minimum number of edges $|E|$ in a graph $G = (V, E)$ given that it has $k \in \{2, \ldots, |V|\}$ strongly connected components. ☆

Solution: Obviously, we must have zero edges between the different SCCs. So for $k = |V|$ we have $|E| = 0$. Suppose now $k < |V|$. Inside a SCC with m vertices (where $m > 1$) the minimal number of edges is achieved when each vertex has only one incoming and only one outgoing edge, i.e. the SCC is a cycle and has m edges. Therefore the minimum number of edges is achieved when $k - 1$ SCCs consist of a single vertex and one SCC is a cycle. In this case we have $|E| = |V| - k + 1$.

3. Show that replacing an edge (u, v) by (v, u) can increase or decrease the number of SCCs in a graph by more than one. ☆

Solution: Consider the graph



It has 3 SCCs. If we change the direction of (s, u) then the graph will have one SCC. (Obviously, changing the direction of this edge once more will increase the number of SCC by 2.)

4. Assume $G = (V, E)$ has k SCCs with $|V(C_i)|$ vertices in component C_i. What is the smallest possible value for $|E|$? ☆