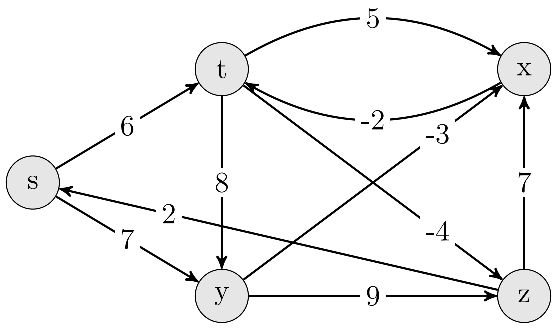
Exercise X, Algorithms 2024-2025

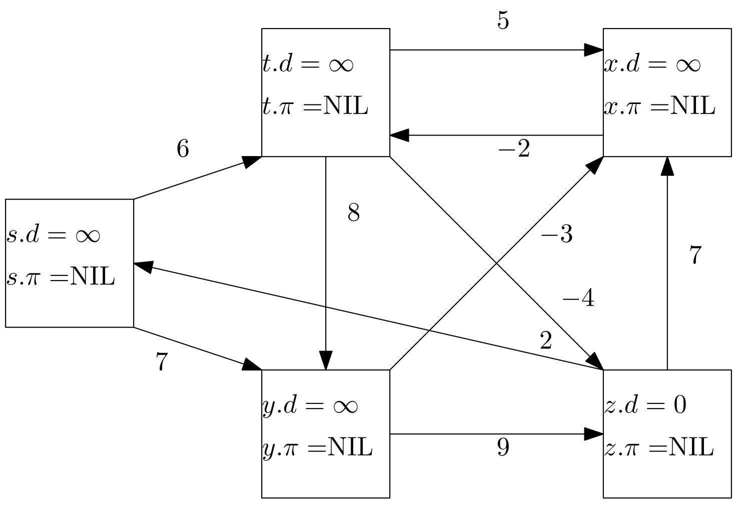
These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. There are many problems on this set, solve as many as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

1 Consider the following edge-weighted directed graph:

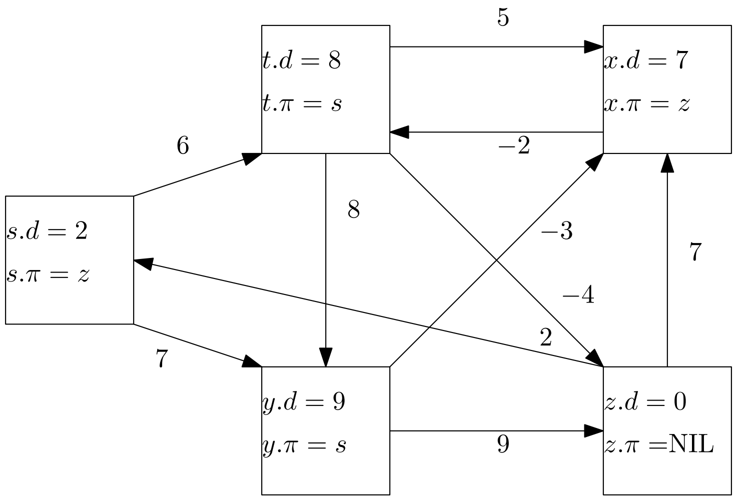


Run the Bellman-Ford algorithm using vertex z as a source. In each pass, relax edges in the order $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$, and show the d and π values after each pass. Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.

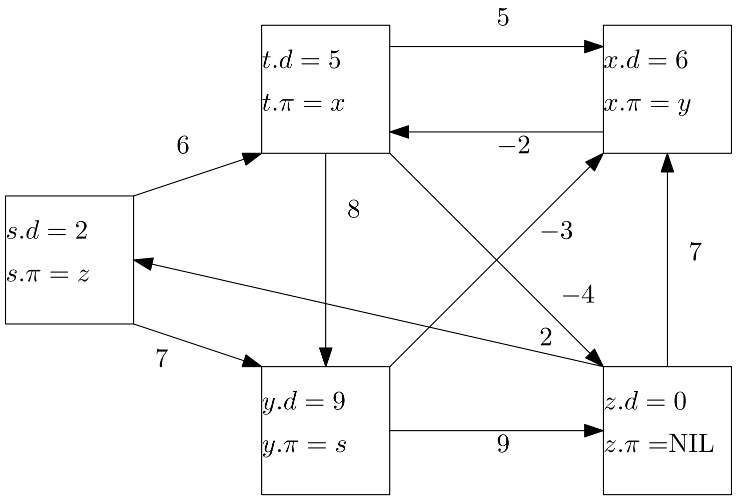
Solution: The state of the algorithm before the first iteration will be:



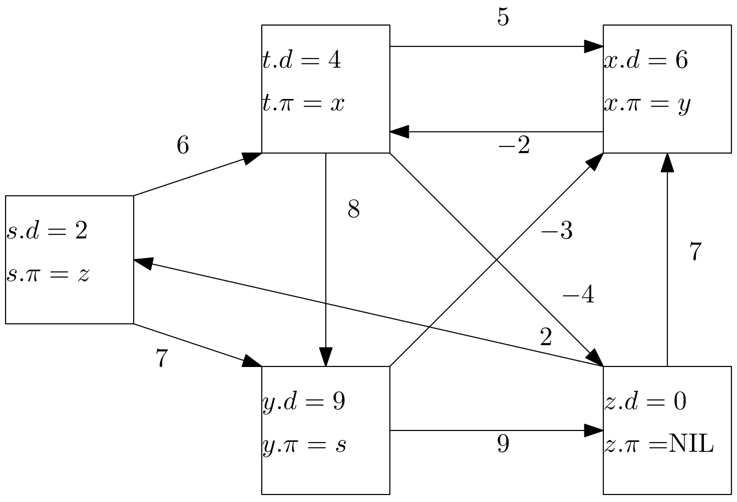
The state of the algorithm after the first iteration will be:



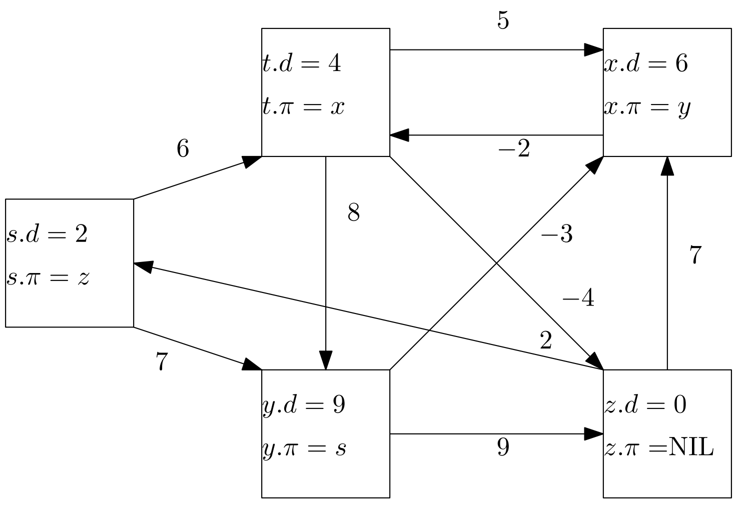
The state of the algorithm after the second iteration will be:



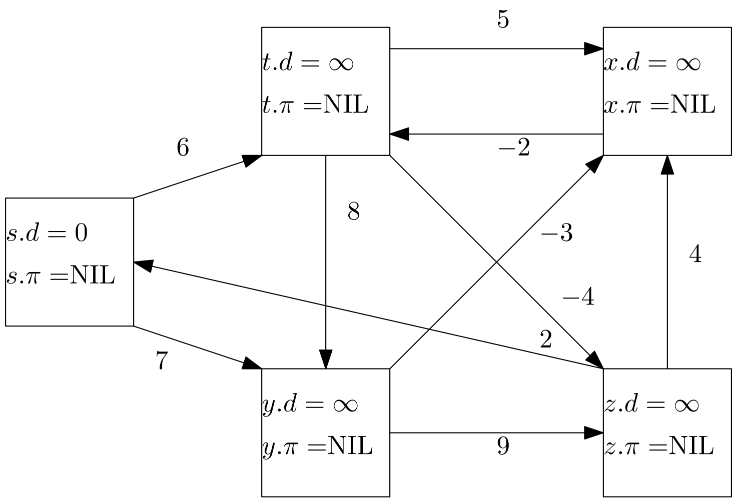
The state of the algorithm after the third iteration will be:



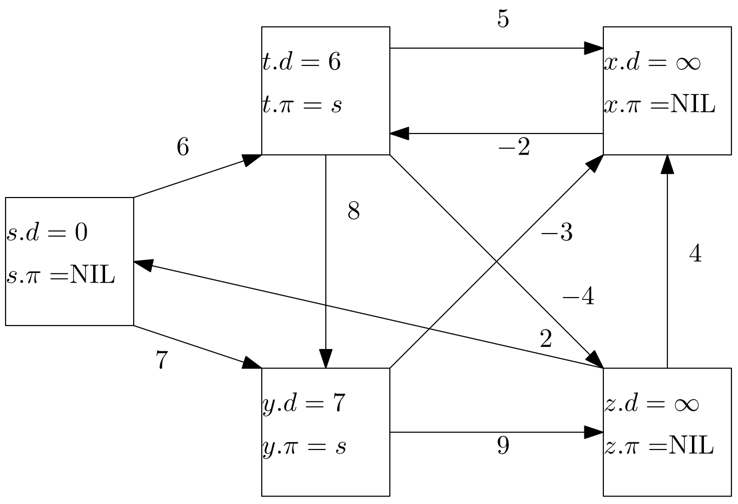
The state of the algorithm after the fourth iteration will be:



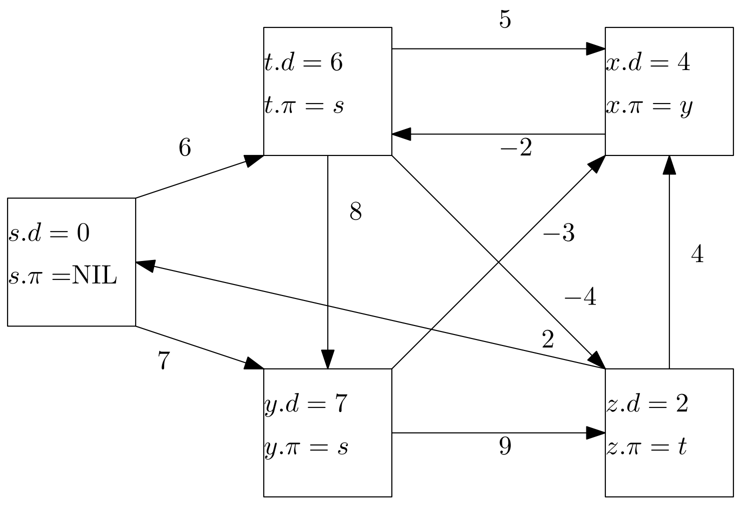
Now, let us change the weight of edge (z, x) to 4. The state of the algorithm before the first iteration will be:



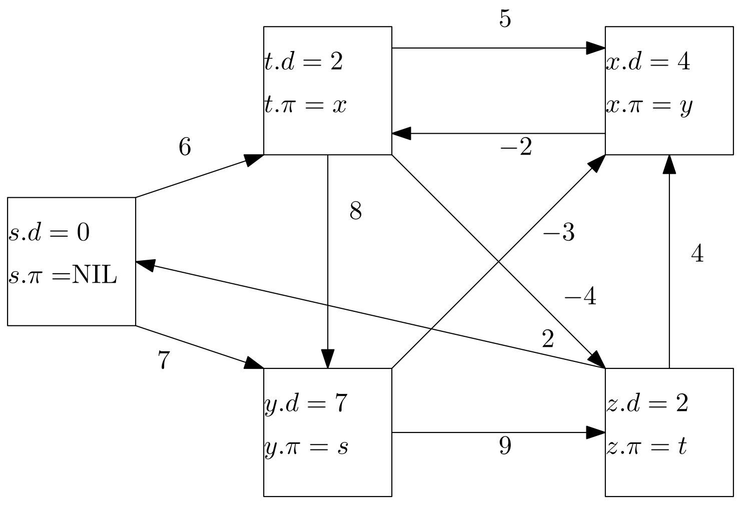
The state of the algorithm after the first iteration will be:



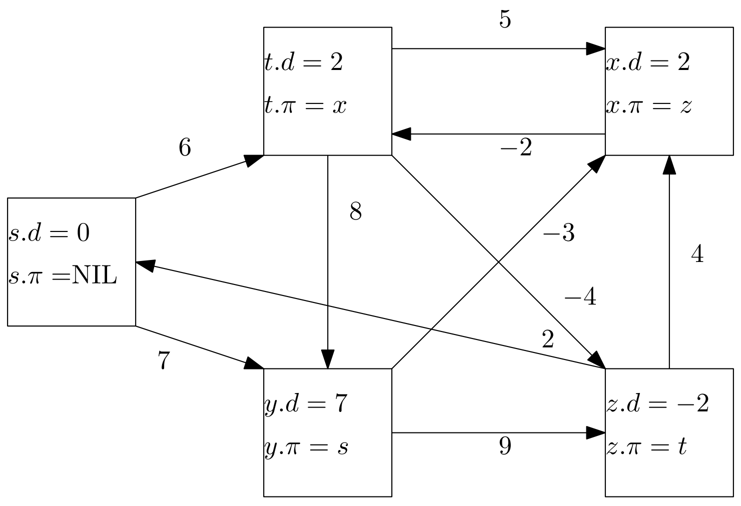
The state of the algorithm after the second iteration will be:



The state of the algorithm after the third iteration will be:



The state of the algorithm after the fourth iteration will be:



Note that for edge (z, t) it holds that $t.d > z.d + w(z, t)$, therefore we have a negative cycle(in this case, it is $z \to t \to x$).

2 (Exercise 24.1-3) Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

Solution: If the size of the longest shortest path from the source is m, then we know that after m iterations of the Bellman-Ford algorithm, every vertex v has its shortest path weight $v.d$ correct. In all subsequent iterations, no $v.d$ value will change. So, in order to run exactly $m + 1$ iterations, after each iteration we should check if a $v.d$ value has changed. If not, we are done and we can stop the algorithm. (We will still perform no more than $|V|$ iterations.)

3 (*, Exercise 24.1-6) Suppose that a weighted directed graph $G = (V, E)$ has a negative-weight cycle. Give an efficient algorithm to list the vertices of one such cycle. Argue that your algorithm is correct.

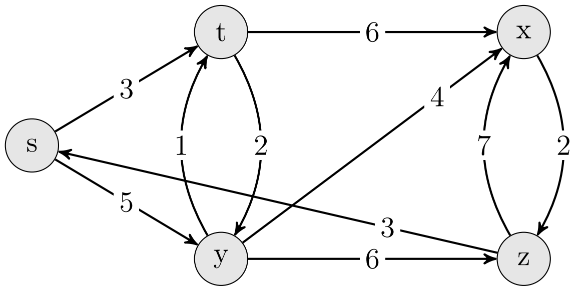
Solution: We have seen in class that the Bellman-Ford algorithm can be used to detect a negative cycle. With a slight modification, we can indeed list all vertices that belong to this negative cycle. (Because we are trying to detect all negative cycles, not just ones reachable from a given source, we can start by adding a new source vertex s and connecting it by a 0-weight edge to each other vertex. Alternatively, we could begin by setting $v.d = 0$ for every v.)

First, there is a negative cycle if and only if after $|V| - 1$ iterations of Bellman-Ford there exists a vertex v that can still be relaxed by an edge (u, v). (This was proved in the lecture.)

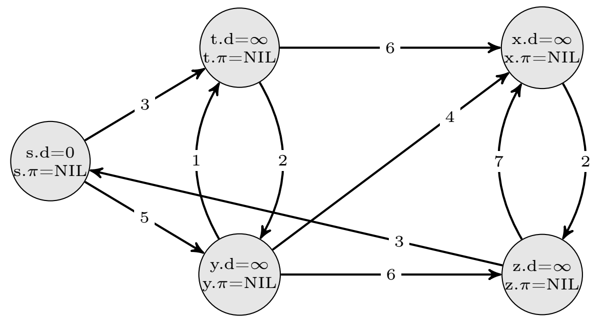
Second, in this case the predecessor array returned by the algorithm will also contain a cycle, and any such cycle will be negative. To see this, assume towards a contradiction that the array contains no cycles. In this case, it represents a directed tree from s to all vertices reachable from s, and we know that this tree has the property that for each vertex w reachable from s, the path from s to w in the tree is a shortest (simple) path from v to w in the graph. This is true after $|V| - 1$ iterations as well as after $|V|$ iterations, implying that the tree cannot change in the $|V|$-th iteration (since any change means making a path shorter). However, it did change – a contradiction.

Therefore, it's enough that we find a cycle in the predecessor array. This can be done in linear time.

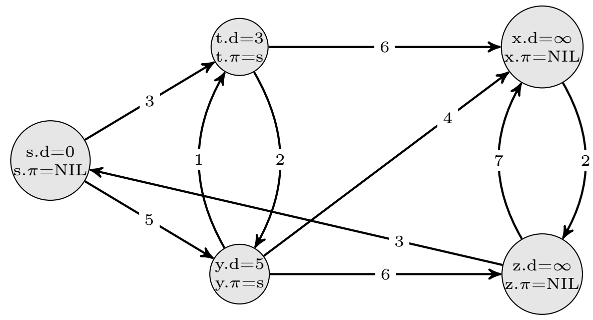
4 (Exercise 24.3-1) Run Dijkstra's algorithm on the following edge-weighted directed graph. Use first vertex s as a source and then vertex z as a source.



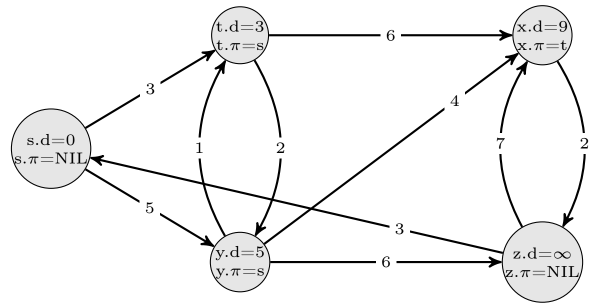
Solution: Let us consider the case where the source is s; the state of the algorithm is initially as follows:



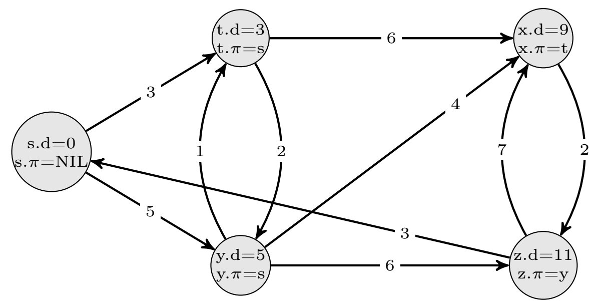
In this case, s will be the first vertex added to the set S used by Dijkstra's algorithm:



Next, t will be inserted into S:

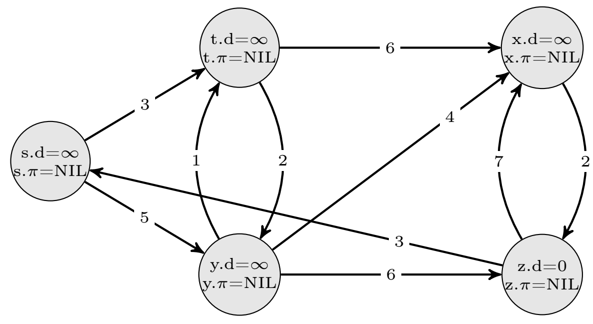


After that, y will be inserted into S:

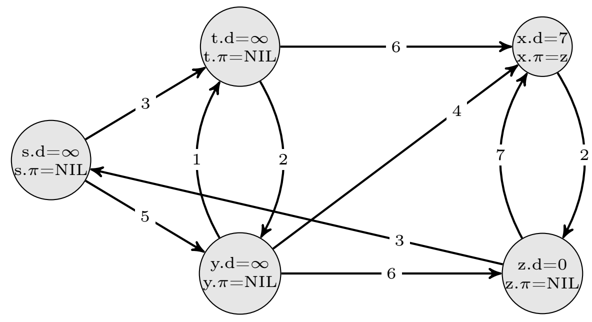


Finally, x and z will be inserted into S (in that order), but no change in the shortest paths structure will occur.

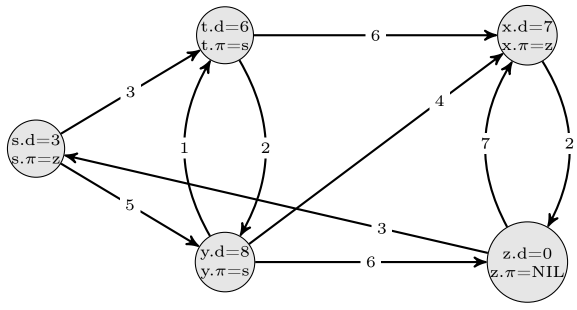
Next, let us consider the case where z is the source:



The first vertex to be put into S will be z:



Next, s will be inserted into S:



After that, t, x and y will be inserted into S, in that order, but there will be no change in the shortest paths structure.

5 (half a *) Suppose you are standing at the top station of Mount-Everest and you wish to ski down. There are several different stations where you can rest (vertices) and different routes between stations (directed edges). All routes go down hill so they are directed from the station of higher altitude to the station of lower altitude. Although there are different routes from the top station to other stations, if one continues to ski down one will sooner or later hit the base camp. As the view is very nice, we wish to make the ski route as long as possible. In other words, design and analyze an efficient algorithm that calculates the *longest* route to ski down starting at the top station of Mount-Everest and ending in the base camp.

Solution: We begin our reasoning by observing the following fact: the graph is acyclic. This follows since any slope is directed from a station of higher altitude to a station of smaller altitude. Therefore the graph does not contain any negative cycles even if we let the weight of each arc be minus its length, i.e., the weight of an arc (u,v) is $-\ell$ if the length of the slope from u to v is ℓ.

Now we solve the shortest path problem (using the Bellman-Ford algorithm) with respect to these weights. As the graph does not contain any negative cycles, the algorithm is guaranteed to find the shortest path from the source to the base camp. That is, it finds the path p that minimizes $\sum_{e\in p} -w(e)$ which is equivalent to maximizing $\sum_{e \in p} w(e)$, which is the length of the path.

The total running time of the algorithm is $O(|V||E|)$.

A small comment is that we heavily relied on that the graph was acyclic to find the longest path. It is strongly believed that there are no efficient algorithms for this problem in general graphs.

Finally, let us remark that a more efficient (and arguably simpler) solution can be designed by dynamic programming. We leave this as an exercise: the approach is basically to compute the longest path from the top to each station in the order of decreasing altitudes of the stations. The solution runs in $O(|V| + |E|)$ time.