Lab **Valeria Fionda, Mouna Kacimi, Werner Nutt, Simon Razniewski**

12. Graphs

1. Longest Paths

For a DAGs, one can *topologically sort* the graph vertices.

(i) Explain what is a topological sort of the vertices of a DAG.

Consider the directed graph G_2 with the set of vertices $\{ A, B, C, D, E, F, H \}$ and the edges

(A, B), (A, C), (A, D), (A, E), (C, F), (D, E), (D, H),
(F, H), (H, B), (H, E)

(ii) Write down an ordering of the vertices that is a topological sort for the graph G_2.

In some applications it is important to find the *longest path* in a DAG. The longest path problem is, given a DAG $G = (V, E)$ and a start vertex s in V, to label every vertex v in V with the length $v.len$ of the longest path from s to v.

(iii) Explain what it means that a problem has the optimal substructure property! Does the longest path problem have the optimal susbstructure property? Explain your answer.

(iv) Describe an algorithm that, given a DAG and a start vertex s, returns for each vertex v the length of the longest path from s to v. You need not write pseudo-code and can describe the steps of the algorithm in words.

Hint: You may want to first topologically sort the vertices of the graph.

(v) Explain how your algorithm finds the length of the longest path in the graph G_2 above from A to B, from A to C, and from A to D.

(vi) How would you generalize your algorithm for longest paths to weighted graphs, that is graphs where every edge has a non-negative edge weight?

2. Maximal Bottleneck

We consider now the following scenario:

- the nodes in a graph are computers that can forward messages (i.e., routers),
- there is an edge from router x to router y if x can forward messages to y, and
- the weight on the edge from x to y indicates the bandwith of the link from x to y, that is, the number of bits that can be forwarded per second.

Such a scenario can be modeled by a weighted directed graph.

We define formally: for a *path* from x to y, the *bottleneck* of the path is the *minimum* of the edge weights on the path. For example, for the path

$$A \xrightarrow{7} B \xrightarrow{5} C \xrightarrow{3} D \xrightarrow{9} E \xrightarrow{8} F$$

the bottleneck is 3. This is because the minimum determines the bandwidth of the entire path: it is not possible to send more that 3 bits per second across this path.

The Maximal Bottleneck Problem is defined as follows:

> Given a start node s, find for every node t that path from s to t for which the bottleneck from s to t is maximal.

In our application, a path from s to t with maximal bottleneck is a path from s to t over which one can route the highest possible number of bits.

(i) Modify Dijkstra's algorithm so that it solves the Maximal Bottleneck Problem. Briefly, explain why your algorithm is correct.

Hint: To get an idea, observe that

- Dijkstra's algorithm finds a path from s to t such that the sum of weights is minimal,
- the new algorithm has to find a path from s to t such that the minimum of weights is maximal.

Consider the weighted directed graph with the set of vertices $\{ A, B, C, D, E, F \}$ and nine weighted edges, given by the following triples:

$$(A, B, 3),\ (A, D, 5),\ (B, C, 6),\ (B, D, 6),\ (B, E, 5),$$
$$(C, F, 2),\ (D, E, 1),\ (E, C, 4),\ (E, F, 3).$$

For example, the triple $(A, B, 3)$ says there is an edge from A to B with weight 3.

(ii) Make a drawing of the graph with sufficient space among the nodes.
Run your new algorithm on this graph with start node A.
Visualize the execution of the algorithm,
like we did it for Dijkstra's algorithm in the lecture.
Make it clear in which order the different steps were executed.
In particular, one should see what is the maximal bottleneck
between A and any other node in the graph.

(iii) Explain why the changed algorithm solves the Maximal Bottleneck Problem.