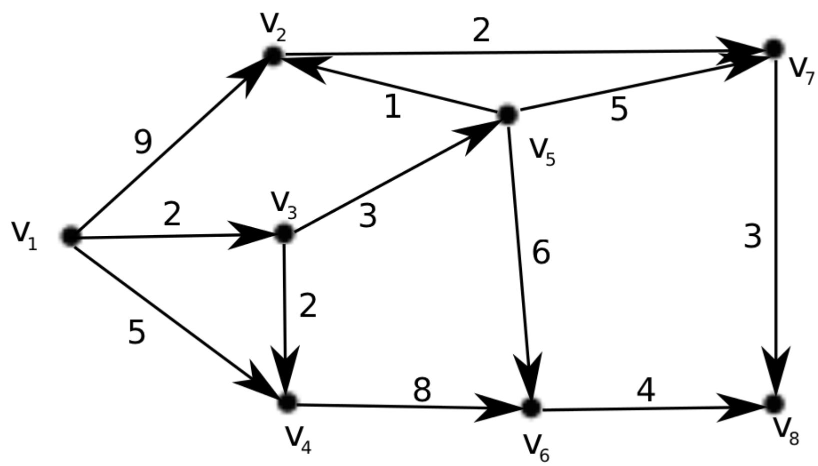
Communications and Transport Systems
Department of Science and Technology **Fall 2021**
Linköping University

Christiane Schmidt

Design and Analysis of Algorithms Part 1 - Mathematical Tools and Network Problems homework 5, 14.01.2022

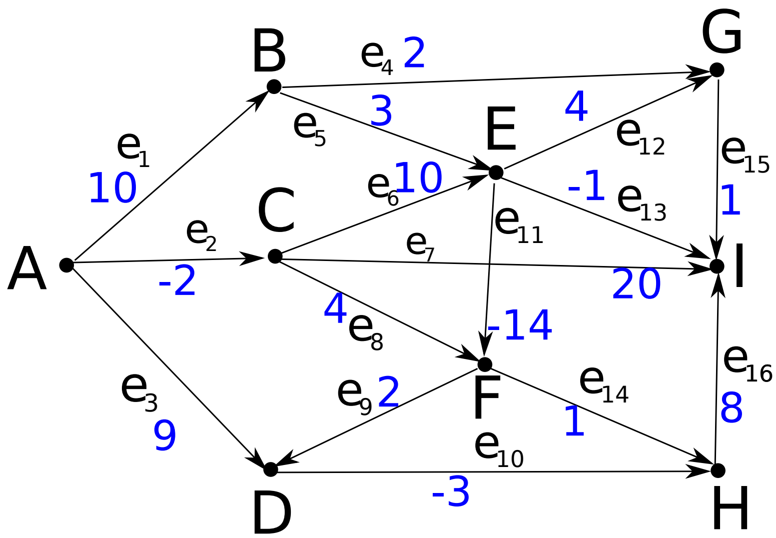
Problem 1 (Shortest r-Arborescence): Given a digraph $D = (V, A)$, a vertex r, and a weight function $\ell : A \to \mathbb{Q}_+$.
We look for a shortest r-arborescnece (that is, an arborescence rooted in r).
The greedy algorithm for this problem is:
Start at r and iteratively extend the r-arborescence of the subset $U \subseteq V$ by the shortest edge leaving U.
Does this algorithm compute a shortest r-arborescence? Motivate your claim.

Problem 2 (Dijkstra):



Use Dijkstra's algorithm to determine a shortest path from v_1 to v_8. (Tie breaking: choose the vertex with lowest index.)

Problem 3 (Moore-Bellman-Ford):



Use the Moore-Bellman-Ford algorithm to determine a shorest path from A to I.

Problem 4 (Shortest Paths in Graphs with Arbitrary Weights):
The algorithms by Dijkstra and Moore-Bellman-Ford for non-negative or conservative edge weights, respectively, use Lemma 5.33. Show that for general graphs the minimum is not defined, that is, show that graphs with arbitrary real edge weights exist, such that for two vertices s and t no shortest path exists.

Problem 5 (Dijkstra and MSTs):
Given an undirected graph G with edge weights $c : E(G) \mapsto \mathbb{R}$ (not pairwise different). Prove or disprove: The shortest paths tree computed with Dijkstra's algorithm is also an MST.

(10 points)

Problem 6 (Shortest Paths Between All Pairs of Vertices):
Input: Digraph G, conservative edge weights $c : E(G) \to \mathbb{R}$.
Output: Shortest paths for all $s, t \in V(G)$, d.h.

l_{st}: Length of a shortest $s-t$–path

p_{st}: Predecessor of t in a shortest $s-t$–path.

(a) Which runtime to we get by solving the problem by repeatedly applying Moore-Bellman-Ford?

(b) Consider the algorithm by Floyd and Warshall, algorithm 1. What is its runtime?

(c) Show that the algorithm by Floyd and Warshall is correct. Hint: Show the following statement:
After the outer loop with $j = 1, \dots, j_0$ the variable l_{ik} contains the length of a shortest i–k–path that only considers the intermediary vertices $1, \dots, j_0$; (p_{ik}, k) is the last edge of such a path.

Algorithm 1: Floyd, Warshall (1962)

Input : Digraph G, conservative edge weights $c : E(G) \to \mathbb{R}$
Output: For each pair of vertices $i, j \in V(G)$: l_{ij}: Length of a shortest $i - j$–path, p_{ij}: Predecessor of j in a shortest path.

Consider $V(G) = \{1, \dots, n\}$

1.
$l_{ij} := c((i,j))$ for all $(i,j) \in E(G)$
$l_{ij} := \infty$ for all $(i,j) \in V(G)^2 \setminus E(G), i \neq j$
$l_{ii} := 0$ for all $i \in V(G)$
$p_{ij} := i$ for all $(i,j) \in E(G)$

2.
```
for j = 1 to n do
    for i = 1 to n do
        if i ≠ j then
            for k = 1 to n do
                if k ≠ j then
                    if l_ik > l_ij + l_jk then
                        l_ik := l_ij + l_jk;
                        p_ik := p_jk
```

Problem 7 (Shortest paths in graphs with arbitrary weights):
The algorithms by Dijkstra and Moore-Bellman-Ford for non-negative and conservative edge weights, repsectively, use Lemma 5.33.
Show that the minimum is not defined for general graphs, that is, show that there exist graphs with arbitrary real edge weights for which there is no shortest path between two vertices s and t.

Problem 8 (Longest paths and the knapsack problem):
We consider n objects. Each object has a *weight* a_j and a *value* c_j, both are positive integers. We ask for a subset of these objects, such that the sum of their weights does

not exceed a bound b (the size of the knapsack) and the sum of their values is maximum (that is, we want to pack as valuable stuff as possible).
Reduce this problem to finding a longest path in an appropriate network.
Hint: Consider an acyclic network with a start vertex s, and end vertex t and $b+1$ vertices for each object.

Problem 9 (Paths and cuts):
Let G be an undirected graph with weights $c : E(G) \to \mathbb{Z}_+$ and two vertices $s, t \in V(G)$, where t is reachable from s. Remember: for a vertex set $X \subseteq V(G)$ we call the edge set $\delta(X) = \{\{x, y\} \in E(G), x \in X, y \in V(G) \setminus X\}$ a cut in G. If $s \in X$ and $t \notin X$, $\delta(X)$ separates the vertices s and t.
Show that the minimum length of an s-t-path is equal to the maximum number of cuts that separate s and t, such that each edge e is contained in at most $c(e)$ such cuts.
(Hint: Why does it suffice to consider a graph with unit weights? Show that the maximum number of such cuts is as well an upper as a lower bound for the minimum length. For the proof of the lower bound give a set of cuts by using a BFS tree.)