Communications and Transport Systems
Department of Science and Technology **Spring 2017**
Linköping University

Valentin Polishchuk
Christiane Schmidt

Design and Analysis of Algorithms Part 1 - Mathematical tools and Network problems homework 2, 30.03.2017

Problem 1 (The Kevin Bacon oracle):
The *Kevn Bacon oracle* is based on the actor graph G: actors are given as vertices. Two actor vertices are connected by an edge if they appeared in a movie together. The vertex of Kevin Bacon has value 0; the *Kevin-Bacon number* (KBN) of another actor is the length of a shortest path in G. (Tom Hanks played with Kevin Bacon in Apollo 13, thus, he has Kevin-Bacon number 1.)
The oracle is available here: `http://oracleofbacon.org/`. The movie data it is based on is taken from the *Internet Movie Database*: `http://www.imdb.com`.
Our questions:

(a) Describe a strategy to definitely find an actor with a KBN as high as possible in G, even if you've never heard of Hollywood. On which graph algorithm is this strategy based?

(b) Find a vertex with KBN at least 4.

Problem 2 (Trees - moved from Homework set 1):

(a) Prove Theorem 1.61 from the lecture.

(b) Prove Theorem 1.62 from the lecture.

(c) Prove Corollary 1.64 from the lecture.

Problem 3 (Directed cycles and directed cuts - moved from Homework set 1):
Show:
In a digraph G, each edge belongs either to a (directed) cycle or to a directed cut. Moreover, the following statements are equivalent:

(a) G ist strongly connected.

(b) G contains no directed cut.

(c) G is connected and each edge of G belongs to a cycle.

(Hint: Take a look at the statements you proved in Problem 2.)

Problem 4 (Eulerian Path):

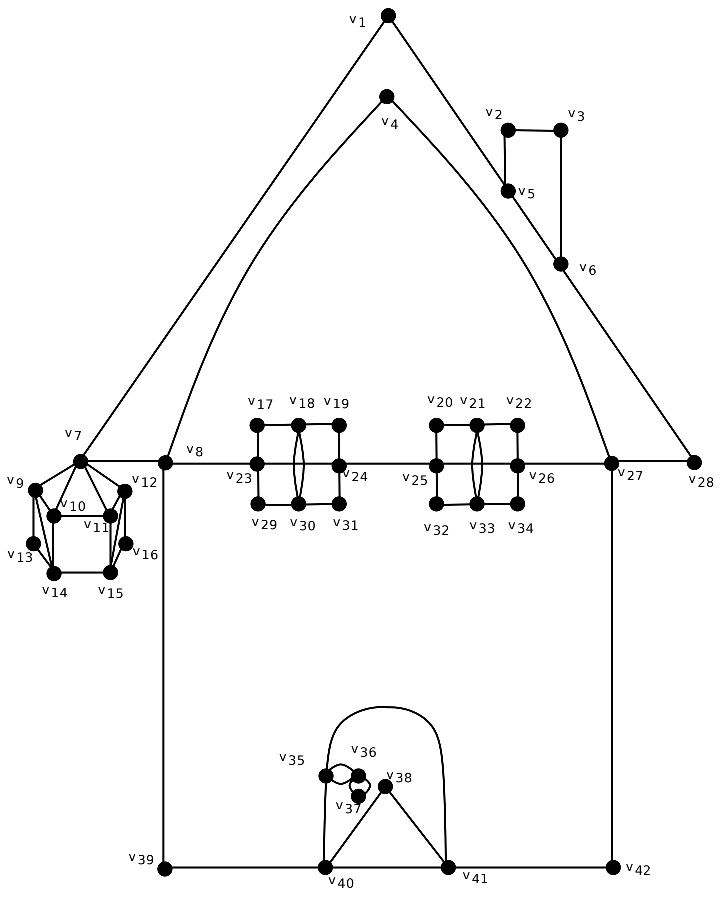


Abbildung 1: Euler on his way home!

Find a Eulerian path in the graph from Figure 1 or show that none exists.

Problem 5 (BFS and DFS):

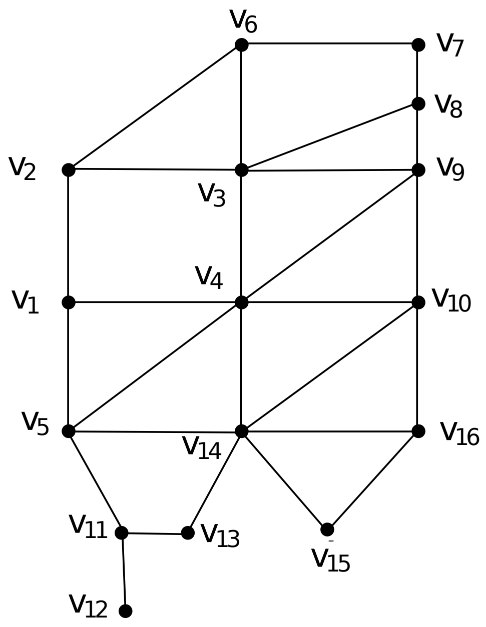


Abbildung 2: The graph G.

a) Apply BFS with start vertex v_1 to graph G from Figure 2.

b) Apply DFS with start vertex v_1 to graph G from Figure 2.

c) Give the adjacency list for G.

(Ad a) and b): If at any time there is more than one vertex to choose from, use the one with the smallest index.)

Problem 6 (BFS and DFS in trees):
Construct an algorithm that determines whether an arbitrary given graph G=(V,E) is a tree based on

(a) DFS

(b) BFS

Problem 7 (Trees and Leaves):

Show that (also during winter) each (undirected) tree has a leaf. (Hint: In an undirected tree a leaf is defined as a vertex of degree 1.)

(10 points)

Problem 8 (BFS):

Let $G = (V, E)$ be a graph and $s \in V$ a vertex; for an arbitrary vertex $x \in V$ let $d(s, x)$ denote the length of a shortest path from s to x. Let $e = \{u, v\} \in E$ be an edge.

a) Prove: $d(s, v) \leq d(s, u) + 1$.

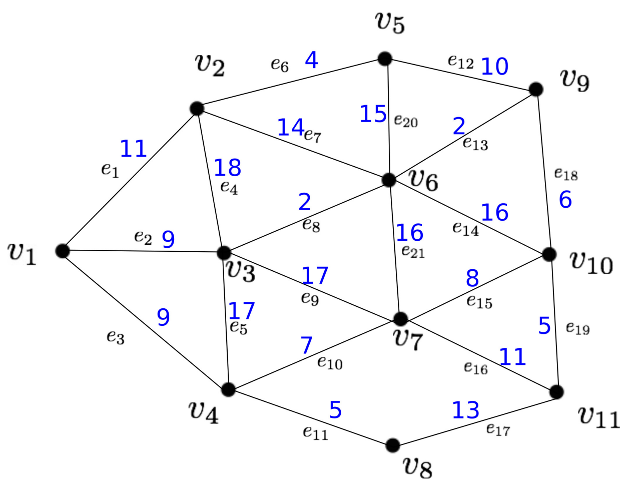
b) Prove or disprove: $d(s, u) \leq d(s, v) + 1$.

c) Does $d(s, v) = d(s, u) + 1$ oder $d(s, u) = d(s, v) + 1$ always hold?

Problem 9 (Forests and Connected Components):
Show: Given a forest with n vertices, m edges and p connected components, then $n = m + p$ holds.

(8 points)

Problem 10 (Prim's algorithm):



Use Prim's algorithm to determine a minimum spanning tree; start with vertex v_1. (Note: we break ties, when several vertices could be chosen in an iteration, by choosing the vertex with the lowest index.)

Problem 11 (Independence Systems and Matroids):
Given an undirected, connected graph G. Let $E = E(G)$,
$\mathcal{I} = \{F \subseteq E : F$ is subset of an Hamiltonian circuit in $G\}$.

(a) Show that the set system $(E, \mathcal{I})$ is an independence system.

(b) Test whether the set system $(E, \mathcal{I})$ is a matroid.

Let E_2 be a finite set, k a positive integer, and $\mathcal{I}_2 = \{F \subseteq E_2 : |F| \leq k\}$.

(c) Test whether the set system $(E_2, \mathcal{I}_2)$ is a matroid.

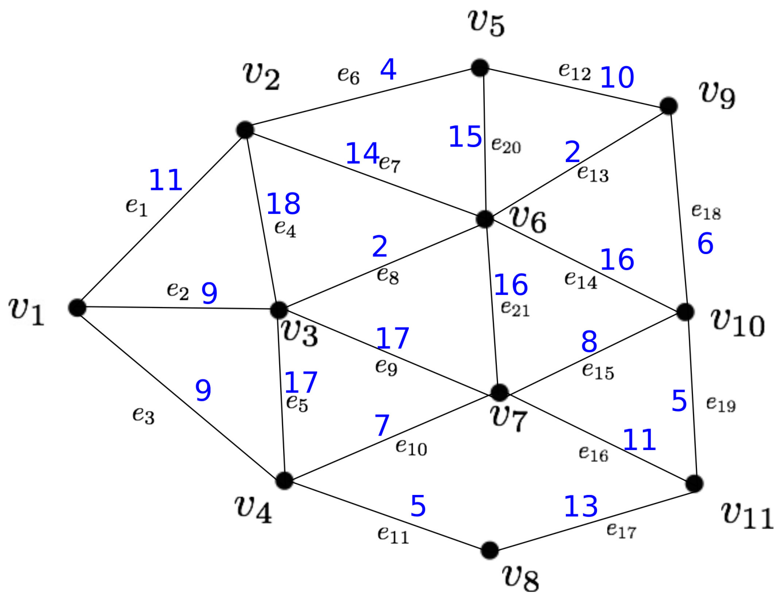
Problem 12 (Trees): Let (V, T_1) and (V, T_2) be two trees on the same set of vertices V. Show: For each edge $e \in T_1$ there exists an edge $f \in T_2$, such that both $(V, (T_1 \setminus \{e\}) \cup \{f\})$ and $(V, (T_2 \setminus \{f\}) \cup \{e\})$ are trees.

Problem 13 (Minimum Spanning Trees):
Given an undirected, connected graph $G = (V, E)$. Prove or disprove the following statement:

If G contains a cycle with a unique lightest edge e (= edge of lowest weight), e is contained in every MST.

Problem 14 (Bottleneck Spanning Trees):
Give a linear time algorithm that for a given graph G and an integer b computes whether the value of a bottleneck spanning tree is at most b.

Problem 15 (Kruskal):



Determine an MST using Kruskal's algorithm. Give the edges in the order in which they are included to the tree, and draw the resulting solution to the problem. Tie breaking: if in any step several edges could be chosen, choose the one with the smallest edge index.

We assume that we write edges as $e = (v_i, v_j)$ with $i < j$.

IMPORTANT: To obtain a runtime of $O(m \log n)$, the data structure presented in the seminar can be used. Give the state of the data structure after each edge insertion.

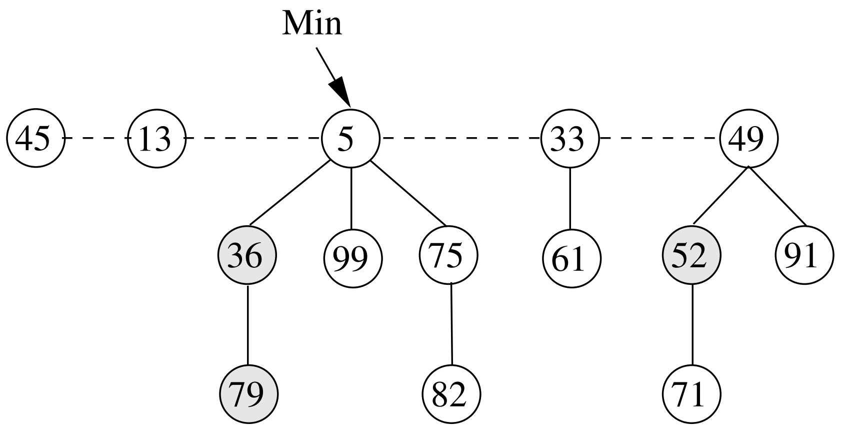
(Note: if there is more than one possibility to add an edge, choose the edge that runs from the vertex with lower index to the vertex with higher index.)

Problem 16 (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 17 (Fibonacci Heaps):

Fibonacci heaps allow for an efficient implementation of Dijkstra's shortest paths algorithm. More detailed information on this data structure can, for example, be found here `https://www.cs.princeton.edu/~wayne/teaching/fibonacci-heap.pdf`, or in Chapter 19 of the book Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein.
In the following Fibonacci heap marked vertices are shown in gray.

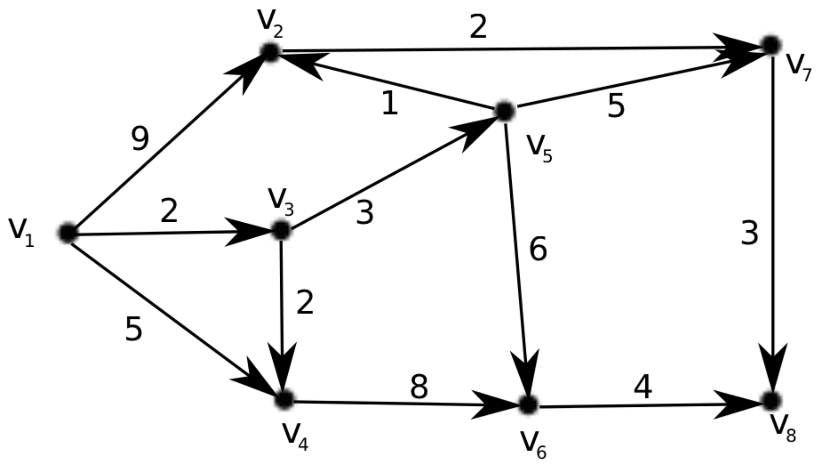


Execute the following operations one after another on the given Fibonacci heap:

Insert(42), Delete_Min, Decrease_Key(91 to 60), Decrease_Key(71 to 61)

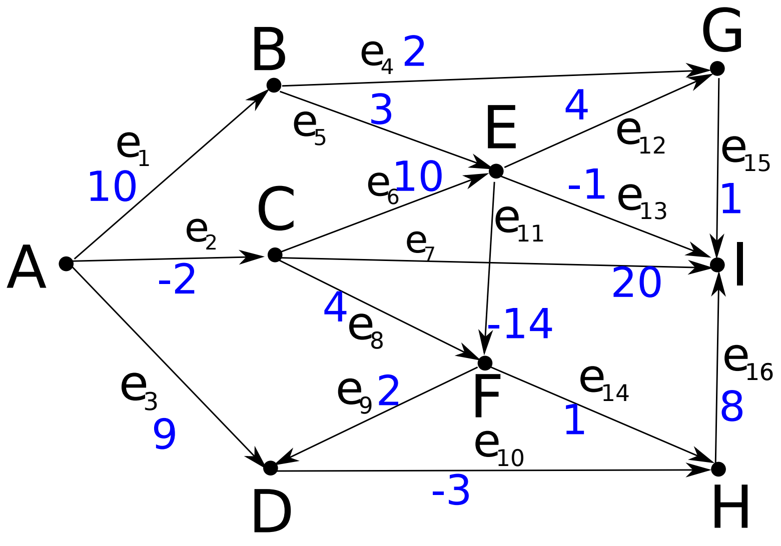
Depict each resulting Fibonacci heap (possibly with intermediate states).

Problem 18 (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 19 (Moore-Bellman-Ford):



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

Problem 20 (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 21 (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 22 (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 $V(G) = \{1, \ldots, n\}$
1.
$l_{ij} := c((i,j))$ for all $(i,j) \in E(G)$
$l_{ij} := \infty$ for all $(i,j) \in V(G)^2 \backslash E(G), i \neq j$
$l_{ii} := 0$ 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 $i \neq j$ then
            for $k = 1$ to $n$ do
                if $k \neq j$ then
                    if $l_{ik} > l_{ij} + l_{jk}$ then
                        $l_{ik} := l_{ij} + l_{jk}$;
                        $p_{ik} := p_{jk}$
```