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 6, 31.01.2022

Problem 1 (Algorithm 8.7, Ford-Fulkerson):

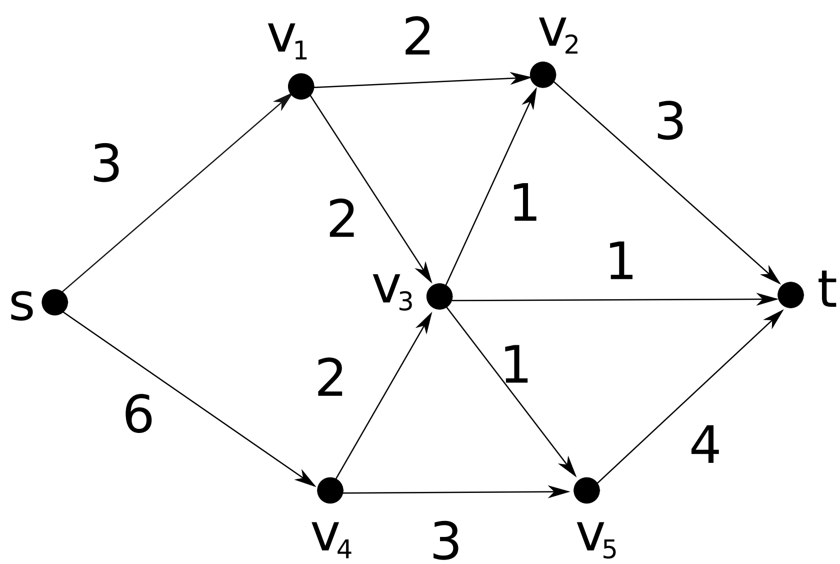

Figure 1: The network (G, u, s, t). The numbers at the edges give the capacities

Use the algorithm by Ford and Fulkerson to determine a maximum $s-t$-flow in the network (G, u, s, t). Give the residual graph in each step.
In addition: give a minimum cut.

Problem 2 (Menger's Theorem (Menger 1927)):
Two paths P and Q are called edge-disjoint if they have no common edge.
Let G be a graph (directed or undirected), let s and t be two vertices and $k \in \mathbb{N}$. Then there are k edge-disjoint s-t-paths if and only if after deleting $k-1$ edges t is still reachable from s.

Problem 3 (Ford-Fulkerson algorithm and irrational capacities):
Show that the algorithm by Ford and Fulkerson might not terminate when it is applied to a network with irrational capacities.

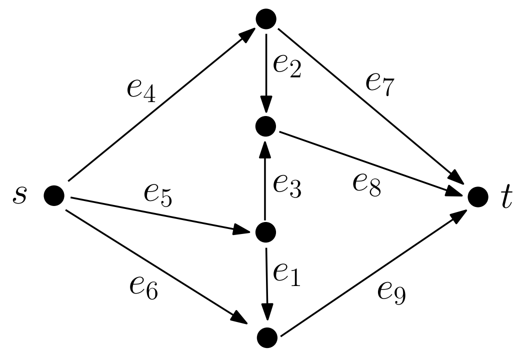

Figure 2: A network with irrational capacities

Consider the network in Figure 2 with capacities $u(e_1) = 1$, $u(e_2) = \sigma$, $u(e_3) = 1$ und $u(e_4) = u(e_5) = \ldots = u(e_9) = 4$, with $\sigma = \frac{\sqrt{5}-1}{2}$. First show $\sigma^n = \sigma^{n+1} + \sigma^{n+2}$.
(Hint: Consider the paths $P_1 = \{e_4, e_2, \overleftarrow{e_3}, e_1, e_9\}$, $P_2 = \{e_5, e_3, \overleftarrow{e_2}, e_7\}$, $P_3 = \{e_6, \overleftarrow{e_1}, e_3, e_8\}$ und $P_4 = \{e_5, e_3, e_8\}$. Show by induction that we can change the *residual capacities* of e_1, e_2 and e_3 from σ^n, σ^{n+1} and 0 to σ^{n+2}, σ^{n+3} and 0, respectively. Induction base: augment along P_4. Induction step: augment, consecutively, along P_1, P_2, P_1 and P_3.)

Problem 4 (Integer Flow): Show Corrolary 8.12 from the seminar: Let $N = (G, u, s, t)$ be a network. If the capacities $u(e)$ are all integers, then there exists a maximum flow in N, such that all $f(e)$ are integeres (in particular, the optimum flow is integer).

Problem 5 (PUSH-RELABEL algorithm):
This exercise will not be part of the examination, but we can discuss at the examination date.
For each proof you can use all theorems, lemmata etc. with a smaller number.

(a) Show Proposition 8.20: During the execution of the Push-Relabel algorithm f is always an s-t-preflow and ψ is always a distance labeling with respect to f. (Hint: Show that the procedures PUSH and PRELABEL preserve these properties.)

(b) Show Lemma 8.21: If f is an s-t-preflow and ψ is a distance labeling with respect to f, then

   (1) s is reachable from any active vertex v in G_f.

   (2) t is not reachable from s in G_f.

   (Hint: For (1) consider the set of vertices that are reachable from an active vertex v. For (2) use contradiciton.)

(c) Show Theorem 8.22: When the algorithm 8.19 terminates, f is a maximum s-t-flow.

(d) Show Lemma 8.24: The number of saturating pushes is at most mn.

Problem 6 (MIN CUT problem): The MIN CUT problem is defined as follows:
INPUT: Network (G, u, s, t).
OUTPUT: An s-t-cut of minimum capacity.

Show how you can compute a MIN CUT in time $O(n^3)$.

Matching will be covered in the last lecture on February 21, 2022!

Problem 7 (Maximum matching in bipartite graphs):

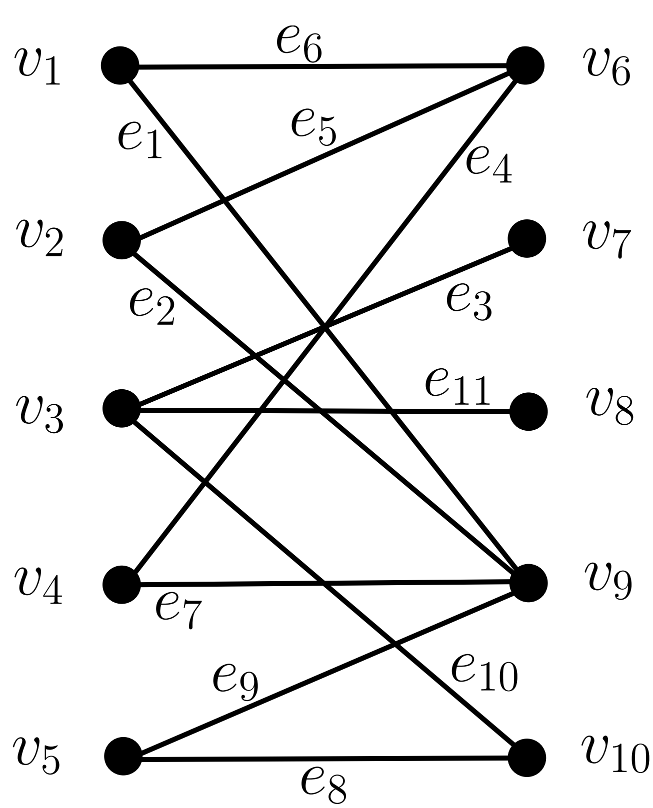

Figure 3: A graph.

Use the flow formulation from the lecture to determine a maximal matching in the graph G from Figure 3. Use your preferred flow algorithm.

Problem 8 (Matching and Vertex Cover):
In bipartite graphs we have $\nu(G) = \tau(G)$ (see seminar notes). In general: $\nu(G) \leq \tau(G)$.

(a) Give a graph with $\nu(G) < \tau(G)$, more precisely $\tau(G) = 2 \cdot \nu(G)$.

(b) Give a graph class with $\nu(G) < \tau(G)$, more precisely $\tau(G) = 2 \cdot \nu(G)$.

Problem 9 ((Inclusion-wise) maximal matchings):

A matching M_0 in a graph G is called *(inclusion-wise) maximal*, if there is no matching M in G with $M_0 \subset M$. Let G be a graph and M_1, M_2 two (inclusion-wise) maximal matchings in G. Show that $|M_1| \leq 2|M_2|$ gilt.
(Hint: Why do the vertices of the matching edges from M_1 and M_2 each constitute a vertex cover? Moreover, we showed that every matching is smaller every vertex cover.)

Problem 10 (Perfect matching in bipartite graphs):
A perfect matching $M \subseteq E$ is a set of pairwise nonadjacent edges, where there is *exactly one* edge incident to each vertex. Show that in a bipartite graph $G = (V, E)$ with $V = V_1 + V_2$ in which each vertex has exactly degree $k \geq 1$, there is a perfect matching. Use the theorem by Hall.

Problem 11 (Blossom Algorithm I.):

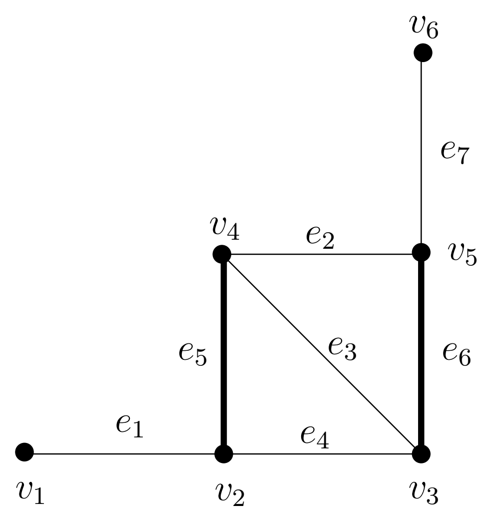

Figure 4: Graph G.

(a) Is the graph G from Figure 4 bipartite? Justify your claim.

(b) Given the graph G from Figure 4 and the matching $M = \{e_5, e_6\}$.

With the help of the blossom algorithm from the lecture decide whether G has a perfect matching or not. Startwith the matching M. After each

- *Augmentation* give the new matching
- *Tree-extension operation* give the new tree
- *Shrinking* give the new tree and the graph G'

Always choose the unmatched vertex with smallest index as starting vertex for your tree. If there is more than one edge to choose from in step 3 of the blossom algorithm, choose the edge with smallest edge index.

Problem 12 (Blossom Algorithm II.):

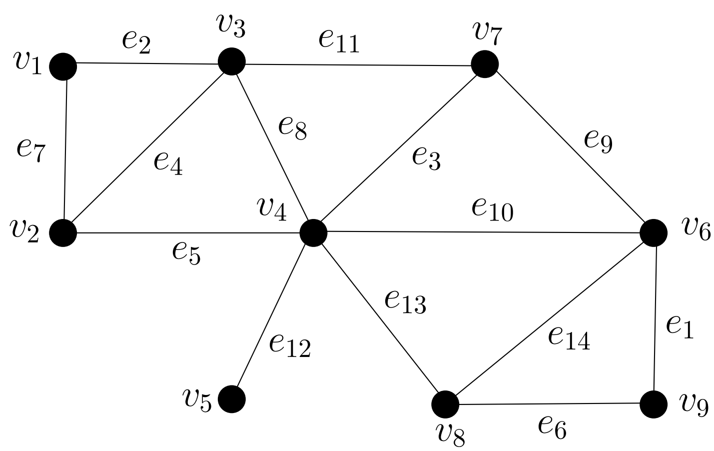

Figure 5: Graph H.

Use the blossom algorithm from the lecture to decide whether the graph H from Figure 5 has a perfect matching or not.
Always choose the unmatched vertex with smallest index as starting vertex for your tree. If there is more than one edge to choose from in step 3 of the blossom algorithm, choose the edge with smallest edge index. Consider the constructed tree when the algorithm stops. Delete the black vertices from G and justify that a perfect matching exists.

Problem 13 (Perfect Matching):

Use the theorem from Tutte to show whether the graph H' from Figure 6 has a perfect matching.

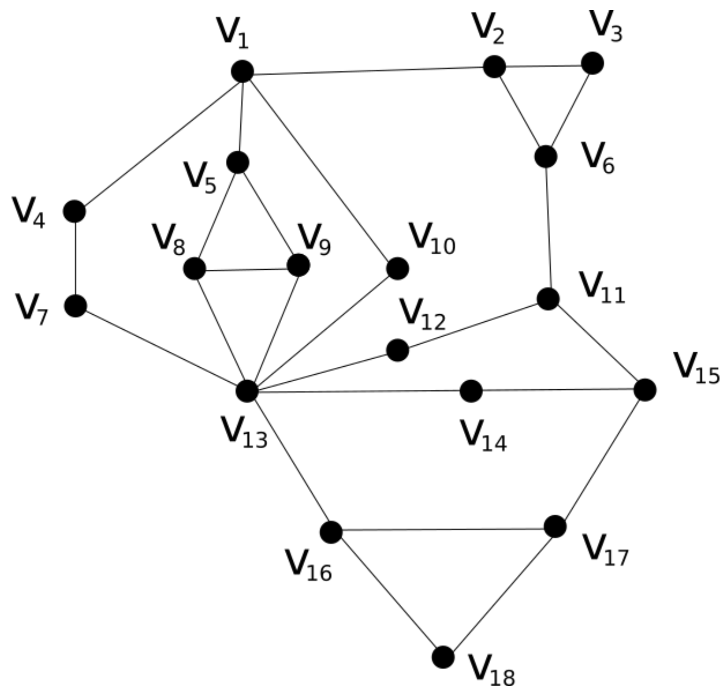

Figure 6: Graph H'.