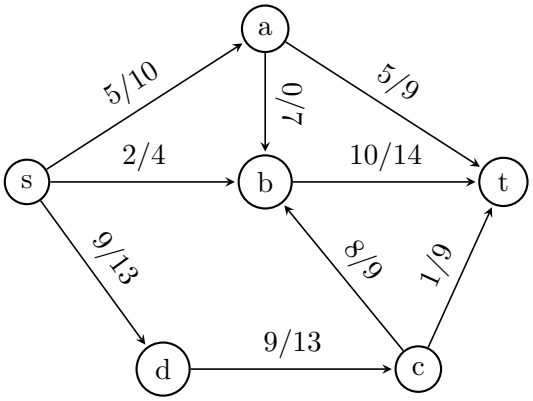
University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Marc Fuchs

Algorithm Theory
Sample Solution Exercise Sheet 8

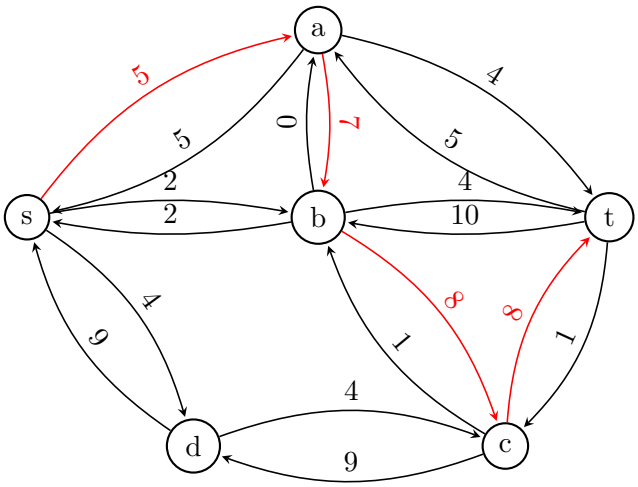
Due: Friday, 12th of December 2025, 10:00 am

Exercise 1: Ford-Fulkerson Algorithm *(4 Points)*

Consider the following flow network, where for each edge, the capacity (second number) and a current flow value (first number) are given. Solve the maximum flow problem on the network by using the Ford-Fulkerson variant that always picks a best possible augmenting path (an augmenting path that improves the current flow from s to t by as much as possible) in every iteration. Give intermediate results, i.e., draw the residual graph with all the residual capacities in every iteration.

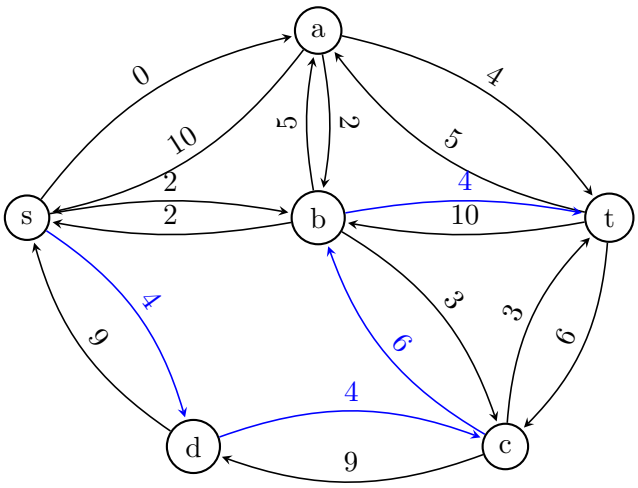


Sample Solution



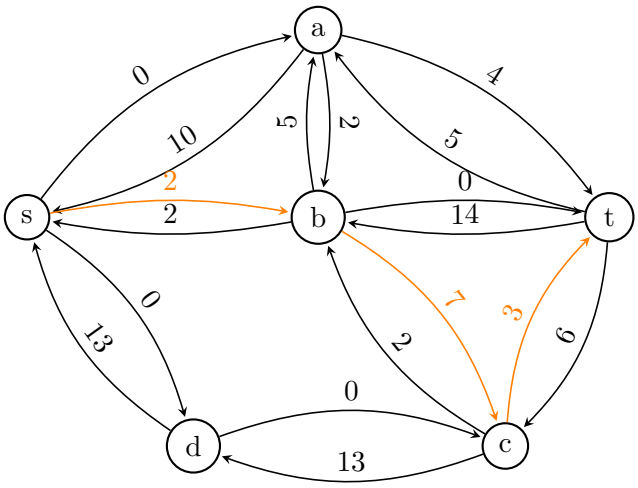
The best possible augmenting path is: $s \to a \to b \to c \to t$. This path increases the flow by 5. Hence, the resulting flow value is 21.

The best possible augmenting path now is: $s \to d \to c \to b \to t$. This path increases the flow by 4.



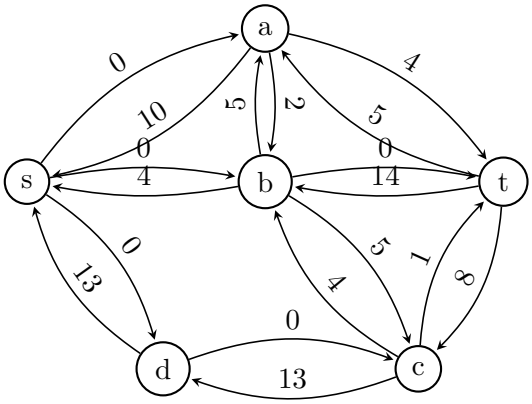
Hence, the resulting flow value is 25.

The best possible augmenting path now is: $s \to b \to c \to t$. This path increases the flow by 2. Hence,



the resulting flow value is 27.

No further improvement possible. Thus, the max. flow is 27.



Exercise 2: Escape Problem *(6 Points)*

An $n \times n$ grid is an undirected graph consisting of n rows and n columns of vertices. We denote the vertex in the ith row and the jth column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices, which are the points (i, j) for which $i = 1, i = n, j = 1$, or $j = n$. Given $\ell \leq n^2$ starting points $(x_1, y_1), (x_2, y_2), \ldots, (x_\ell, y_\ell)$ in the grid, the escape problem is to

determine whether or not there are ℓ vertex-disjoint paths from the starting points to any ℓ different points on the boundary. Give an algorithm that solves the escape problem in polynomial time. Check the figure for an example.

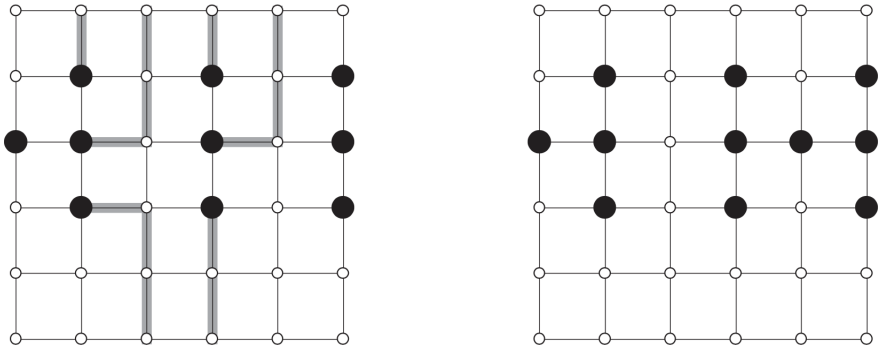


Figure 1: Grids for the escape problem. Starting points are black, and other grid vertices are white. The grid on the left has an escape, shown by shaded paths. The grid on the right has no escape.

Sample Solution

We construct a flow network as follows:

- Replace every edge in the grid by two directed edges (with capacity 1)
- Split each node v in the grid into two nodes, v_{in}, v_{out}, where all ingoing edges of v will now go into v_{in} and all outgoings go out from v_{out}. Add a directed edge from v_{in} to v_{out} (capacity 1)
- Add a new startnode v_{start} that has an outgoing edge to every starting point (i.e. we add ℓ outgoing edges with cap 1).
- Add new target node v_{end} such that all boundary nodes have a directed edge to v_{end} (we ad).

We will now run Ford-Fulkerson (or any Max-Flow) algorithm on that network. If there exists a flow with flow value exactly ℓ, we know that ℓ vertex disjoint path exists as no two path can cross the same grid vertex as by our v_{in}, v_{out} construction.

Exercise 3: Smallest Minimum Cut *(10 Points)*

Let $G = (V, E)$ be a flow network with *integer capacities* $c_e \geq 0$ for all $e \in E$. Give a new flow network $G' = (V, E)$ (that has the same nodes and edges as G) with *integer capacities* $c'_e \geq 0$ such that any minimum cut in G' is a minimum cut in G with the smallest number of edges (of all minimum cuts in G). Proof your statement!
Hint: Consider capacities $c'_e := c_e + 1$. How does the capacity of a cut change by this choice? Does this already solve the task or do you need to adjust it?

Sample Solution

Let $m := |E|$ and $n := |V|$. The intuition is the following: If we would set $c'_e = c_e + 1$, we increase the capacity of each cut (compared to the original value) by the number of edges of the cut. However, the problem is that cuts that were not minimal in the first place, but contain just few edges, may become minimum cuts by that choice of c'_e. To overcome this problem we use that no cut is containing more than m edges.

We define the new capacity for each edge e like follows: $c'_e := m \cdot c_e + 1$. Let in the following $(C, V \setminus C)$ be a minimum cut in G'. Now we want to show that $(C, V \setminus C)$ is also a min cut in G and second, that $(C, V \setminus C)$ has the smallest number of edges among the min cuts of G. We will now show the first of these statements by contradiction i.e., assume there is a cut $(S, V \setminus S)$ that has a smaller capacity than $(C, V \setminus C)$ in G. We start measuring the capacity of this cut in G' and use that the capacity of $(S, V \setminus S)$ is at least 1 unit smaller than the capacity of $(C, V \setminus C)$ in G since we have integer capacities.

$$\begin{aligned}
\sum_{e \in (S, V \setminus S)} c'_e &= \sum_{e \in (S, V \setminus S)} (m \cdot c_e + 1) \\
&= m \sum_{e \in (S, V \setminus S)} c_e + \sum_{e \in (S, V \setminus S)} 1 \\
&\leq m \left(-1 + \sum_{e \in (C, V \setminus C)} c_e \right) + m \\
&= \sum_{e \in (C, V \setminus C)} m \cdot c_e \\
&< \sum_{e \in (C, V \setminus C)} (m \cdot c_e + 1) \\
&= \sum_{e \in (C, V \setminus C)} c'_e
\end{aligned}$$

Thus, the capacity of $(S, V \setminus S)$ would be smaller than the capacity of $(C, V \setminus C)$ in G' and that is a contradiction the definition of $(C, V \setminus C)$. Now we know that $(C, V \setminus C)$ is also a minimum cut in G. It remains to show that we use a minimum number of edges in G. Again, assume for contradiction purpose that there is a cut $(S', V \setminus S')$, that is also a minimum cut in G but uses fewer edges i.e. $\sum_{e \in (S', V \setminus S')} 1 < \sum_{e \in (C, V \setminus C)} 1$ but $\sum_{e \in (S', V \setminus S')} c_e = \sum_{e \in (C, V \setminus C)} c_e$. Contradiction follows by the following lines.

$$\begin{aligned}
\sum_{e \in (S', V \setminus S')} c'_e &= \sum_{e \in (S', V \setminus S')} (m \cdot c_e + 1) \\
&= m \sum_{e \in (S', V \setminus S')} c_e + \sum_{e \in (S', V \setminus S')} 1 \\
&< m \sum_{e \in (C, V \setminus C)} c_e + \sum_{e \in (C, V \setminus C)} 1 \\
&= \sum_{e \in (C, V \setminus C)} (m \cdot c_e + 1) \\
&= \sum_{e \in (C, V \setminus C)} c'_e
\end{aligned}$$

Exercise 4: Session Problem: Seating Arrangement *(0 Points)*

A group of students goes out to eat dinner together. To increase social interaction, they would like to sit at tables such that no two students from the same faculty are at the same table. For that purpose assume there are students from x different faculties while $n_1, n_2, ..., n_x$ describe the affiliated number of students from these x faculties. Also assume that there are y tables available while r_j students can take place on the j-th table.

Let us define the *seating arrangement* as the decision problem returning `true` if one can distribute the students from same faculties to different tables and `false` otherwise. Formulate this *seating arrangement* problem as a *maximum flow* problem and write down the condition that should hold whenever the original decision problem returns `true`. Further, give the runtime it takes to solve the corresponding flow problem in terms of x, y, n_i and/or r_j for all $1 \leq i \leq x$ and $1 \leq j \leq y$.

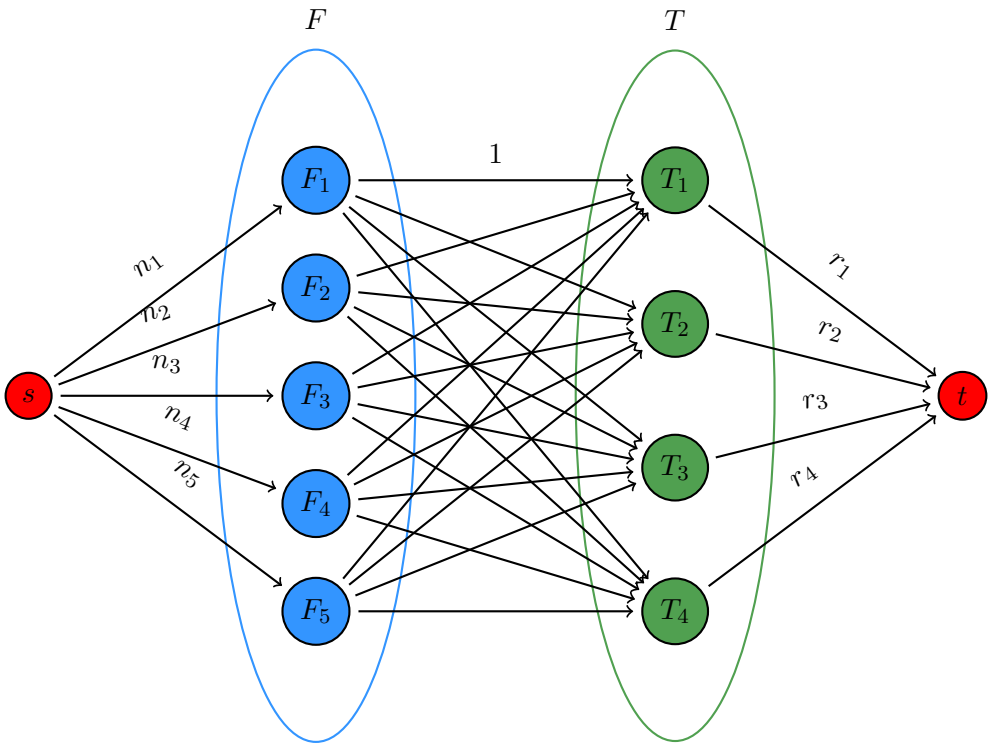
Sample Solution

The idea is to construct a complete bipartite graph G, where the left side consists of the faculties and the right side consists of the possible tables. Each edge in between has capacity 1. Adding a node s on the left, connected to all faculties with corresponding n_i capacities and a node t connected to all tables with corresponding r_i capacities completes the graph. The seating arrangement problem will return `true`, when the maximum flow of G is $\sum_{i=1}^{x} n_i$ and `false` otherwise. The runtime of Ford-Fulkerson is $O(m \cdot C)$ where $m = x + y + x \cdot y = O(x \cdot y)$ and the maximum flow C is at most $\min\{\sum_{i=1}^{x} n_i, \sum_{i=1}^{y} r_i\}$. (Note that one can refine the runtime by the fact that if $\sum_{i=1}^{y} r_i < \sum_{i=1}^{x} n_i$ there is no proper seat arrangement possible.)
Alternatively, a more formal definition of the graph:

$$
\begin{aligned}
&F = \{F_1, F_2, ..., F_x\} \\
&T = \{T_1, T_2, ..., T_y\} \\
&G = (V, E) \text{ with} \\
&V = \{s, t\} \cup F \cup T \\
&E = \{(s, v) | v \in F\} \cup \{(u, v) | u \in F, v \in T\} \cup \{(v, t) | v \in T\}
\end{aligned}
$$

where the capacities are like follows:

$$
\begin{aligned}
c(s, v) &= n_i \text{ if } v = F_i \\
c(u, v) &= 1 \text{ if } u \in F, v \in T \\
c(v, t) &= r_i \text{ if } v = T_i
\end{aligned}
$$



Exercise 5: Session Problem II: Blocked Streets *(0 Points)*

Mr. X has just robbed a bank and is now heading to the harbor. However, the police wants to stop him by closing some streets of the city. As it is expensive to close a street, how can the police decide which streets to close so that there is no route between the bank and the harbor and such that the number of closed streets is minimal?

Assume the city is modeled as an n-node m-edge graph, where the edges represent the streets and the nodes represent the places including the bank and the harbor.

Sample Solution

We weight every edge of the graph with weight 1 and we select the bank as start node s and the harbor as target node t. The min cut between s and t gives us a cut in the graphs consisting of a minimal number of edges (as each edge has the same weight) and when deleting all of them, s and t are disconnected. Hence, computing the min cut using Ford-Fulkerson is enough to determine which streets to block.