# Advanced Algorithms — Exercise Set 1

Name: _____

---

- Submit in class on **September 9, 2025**.
- Feel free to discuss with others, but write up your own solutions.
- This will be graded for completion. Use it to learn!

---

## Flows

**Problem 1.** *Walk through and understand the Ford-Fulkerson demo (FFdemo.pdf) on the course webpage.*

**Problem 2** (Unit capacity networks)**.** *Consider the special case of max-flow in which every edge in the network has a capacity of* 1*. Explain why the running time of the Ford-Fulkerson algorithm is* $O(mn)$ *in this special case.*

**Solution.**

**Problem 3.** *Give an example of a flow network with all integer capacities and a maximum flow on that network which has a non-integer value on at least one edge.*

**Solution.**

**Problem 4** (Multi-source / Multi-sink reduction)**.** *Suppose we generalize the maximum flow problem so that there are multiple source vertices $s_1, \ldots, s_k \in V$ and sink vertices $t_1, \ldots, t_\ell \in V$. Assume no vertex is both a source and a sink, source vertices have no incoming edges, and sink vertices have no outgoing edges. A flow is defined as before: a nonnegative number $f_e$ for each $e \in E$ such that capacity constraints hold on every edge and conservation holds at every vertex that is neither a source nor a sink. The value of a flow is the total outgoing flow at the sources:*

$$\sum_{i=1}^{k} \sum_{e \in \delta^+(s_i)} f_e.$$

*Explain how to solve the maximum flow problem in graphs with multiple sources and sinks using the single-source single-sink version. (Hint: Consider adding additional vertices and/or edges.)*

**Solution.**

**Problem 5** (Undirected Flows)**.** *In class we have focused on the maximum flow problem in directed graphs. In the* undirected *version of the problem, the input is an undirected graph $G = (V, E)$, a source vertex $s \in V$, a sink vertex $t \in V$, and an integer capacity $c_e \geq 0$ for each edge $e \in E$.*

*Flows are defined exactly as before and remain directed. Formally, a flow consists of two non-negative numbers $f_{uv}$ and $f_{vu}$ for each (undirected) edge $(u, v) \in E$, indicating the amount of traffic traversing the edge in each direction. Conservation constraints (*flow in = flow out*) are defined as before. Capacity constraints now state that, for every edge $e = (u, v) \in E$, the total amount of flow $f_{uv} + f_{vu}$ on the edge is at most the edge's capacity $c_e$.*

*The value of a flow is the net amount*

$$\sum_{(s,v)\in E} f_{sv} - \sum_{(v,s)\in E} f_{vs}$$

*going out of the source.*

*Explain how to solve the maximum flow problem in undirected graphs using an algorithm for the maximum flow problem in directed graphs. That is, given an instance of the undirected problem, show how to 1) produce an instance of the directed problem such that 2) given a maximum flow to this directed instance, you can recover a maximum flow of the original undirected instance.*

*[Hint: consider bidirecting each edge.]*

**Solution.**

**Problem 6** (Global Minimum-cuts)**.** *In the (undirected) global minimum cut problem, the input is an undirected graph $G = (V, E)$ with a nonnegative capacity $c_e$ for each edge $e \in E$, and the goal is to identify a **global cut**, i.e., a partition of $V$ into non-empty sets $A$ and $B$ that minimizes the total capacity $\sum_{e \in \delta(A)} c_e$ of the cut edges. (Here, $\delta(A)$ denotes the edges with exactly one endpoint in $A$.)*

*Explain why this problem reduces to solving $n-1$ maximum flow problems in undirected graphs. That is, given an instance of the global minimum cut problem, show how to*

1. *produce $n-1$ instances of the maximum flow problem (in undirected graphs) such that*
2. *given maximum flows to these $n-1$ instances, you can compute an optimal solution to the global minimum cut instance.*

*[Hint: remember that you can use max-flow to get the minimum capacity cut which separates any given pair of vertices. ]*

**Solution.**