# CS 217 – Algorithm Design and Analysis

Shanghai Jiaotong University, Spring 2021

Handed out on Thursday, 2021-04-15
First submission and questions due on Thursday, 2021-04-22
You will receive feedback from the TA.
Final submission due on Thursday, 2021-04-29

# 4 Network Flow

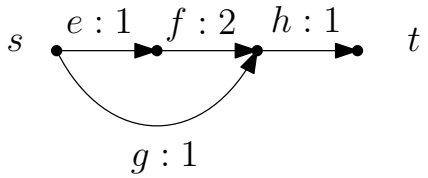
**Exercise 1.** Let $G = (V, E, c)$ be a flow network. Prove that flow is "transitive" in the following sense: If there is a flow from $s$ to $r$ of value $k$, and a flow from $r$ to $t$ of value $k$, then there is a flow from $s$ to $t$ of value $k$.

**Hint.** The solution is extremely short. If you are trying something that needs more than 3 lines to write, you are on the wrong path.

## 4.1 Always, Sometimes, Never Full

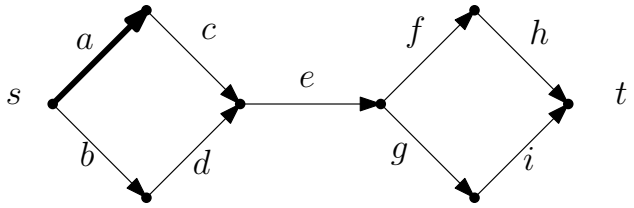
Let $(G, s, t, c)$ be a flow network, $G = (V, E)$. A directed edge $e = (u, v)$ is called *always full* if $f(e) = c(e)$ for every maximum flow; it is called *optionally full* if $f(e) = c(e)$ for some but not all maximum flows; it is called *never full* if $f(e) < c(e)$ for all maximum flows.

Let $(S, V \setminus S)$ be a cut. That is, $s \in S, t \in V \setminus S$. We say the edge $e = (u, v)$ is crossing the cut if $u \in S$ and $v \in V \setminus S$. We say $e$ is *always crossing* if it crosses every minimum cut; *optionally crossing* if it crosses some, but not all minimum cuts; *never crossing* if it crosses no minimum cut. For example, look at this flow network:




Example network: the edges $e, g$ are optionally full and never crossing; $f$ is never full and never crossing; $h$ is always full and always crossing.

**Exercise 2.** Consider this network:

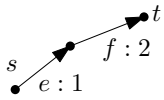
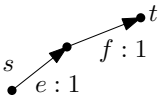



The fat edge $a$ has capcity 2, all other edges have capacity 1.

1. Indicate which edges are (i) always full, (ii) optionally full, (iii) never full.

2. Indicate which edges are (i) always crossing, (ii) optionally crossing, (iii) never crossing.

**Exercise 3.** An edge $e$ can be $(x)$ always full, $(y)$ optionally full, $(z)$ never full; it can be $(x')$ always crossing, $(y')$ optionally crossing, $(z')$ never crossing. So there are nine possible combinations: $(xx')$ always full and always crossing, $(xy')$ always full and optionally crossing, and so on. Or are there? Maybe some possibilities are impossible. Let's draw a table:

| The edge $e$ is: | $x$: always full | $y$: optinally full | $z$: never full |
|---|---|---|---|
| $x'$: always crossing |   | Possible or impossible? | Possible or impossible? |
| $y'$: optionally crossing |   | Possible or impossible? | Possible or impossible? |
| $z'$: never crossing | Possible or impossible? | Possible or impossible? | Possible or impossible? |
| | | | |

The nine possible cases, some of which are maybe impossible.

The two very simple flow networks in the table already show that $(xx')$ and $(yy')$ are possible; that is, it is possible to be always full and always crossing, and it is possible to be always full and optionally crossing. Fill out the table! That is, for each of the remaining seven cases, find out whether it is possible or not. If it is possible, draw a (simple) network showing that it is possible; if impossible, give a proof of this fact. **Remark.** Be rigorous. No handwaving or appealing to intuition here.

## 4.2 Bottleneck Paths

Let $G = (V, E)$ be a directed graph with an edge capacity function $c : E \to \mathbb{R}^+$. For a path $p = u_0 u_1 \dots u_t$ define its *capacity* to be

$$c(p) := \min_{1 \le i \le t} c(\{u_{i-1}, u_i\}) \ . \tag{1}$$

> **Maximum Capacity Path Problem (MCP).** Given a directed graph $G = (V, E)$, an edge capacity function $c : E \to \mathbb{R}^+$, and two vertices $s, t \in V$, compute the path $p^*$ maximizing $c(p)$. We denote by $p^*$ the optimal path and by $c^* := c(p^*)$ its cost.

**Exercise 4.** Suppose the edges $e_1, \dots, e_m$ are sorted by their cost. Show how to solve MCP in time $O(n + m)$.

**Exercise 5.** Give an algorithm for MCP of running time $O(m \log \log m)$. **Hint:** Using the median-of-medians algorithm, you can determine an edge $e$ such that at most $m/2$ edges are cheaper than $e$ and at most $m/2$ edges are more expensive than $e$. Can you determine, in time $O(n + m)$, whether $c^* < c(e)$, $c^* = c(e)$, or $c^* > c(e)$? Iterate to shrink the set of possible values for $c^*$ to $m/4$, $m/8$, and so on.

**Exercise 6.** Give an algorithm for MCP that runs in time $O(m \log \log \log m)$? How about $O(m \log \log \log \log m)$? How far can you get?