Mathematical Foundations of Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

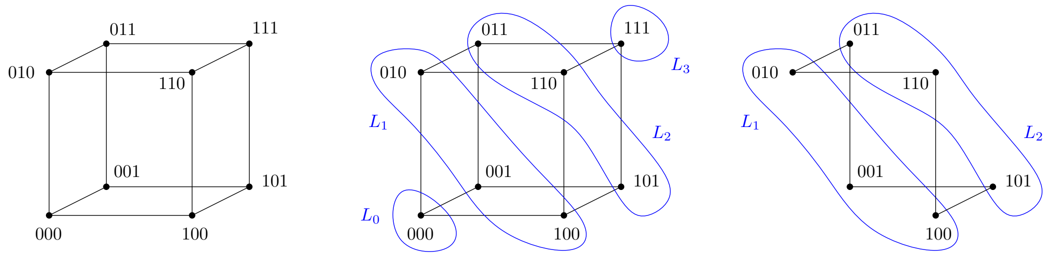
Spring 2019

11 Matchings and Network Flow

- Homework assignment published on Thursday, 2019-05-23.
- Submit questions and first solutions by Wednesday, 2019-05-29, 12:00.
- Submit final solution by Wednesday, 2019-06-05.

11.1 Matchings

Consider the Hamming cube $\{0,1\}^n$. We can view it as a graph H_n, where the vertex set is $\{0,1\}^n$ and two vertices x, y are connected by an edge if x and y differ in exactly one coordinate. Define the k^{th} layer to be $L_k := \{x \in \{0,1\}^n \mid |x|_1 = k\}$, where $|x|_1$ denotes the number of 1s in x. Note that the subgraph induced by layer k and layer $k+1$ is a bipartite graph $H_n[L_k \cup L_{k+1}]$. See the picture below for an illustration ($n = 3, k = 1$):



Exercise 11.1. Let $0 \le k < n/2$. Show that the bipartite graph $H_n[L_k \cup L_{k+1}]$ has a matching of size $|L_k| = \binom{n}{k}$.

Exercise 11.2. Let $G = (V, E)$ be a bipartite graph with left side L and right side R. Suppose G is d-regular (every vertex has degree d), so in particular $|L| = |R|$. Show that G has a perfect matching (that is, a matching M of size $|L|$).

Exercise 11.3. Let G a d-regular bipartite graph. Show that the edges $E(G)$ can be partitioned into d perfect matchings. That is, there are matchings $M_1, \dots, M_d \subseteq E(G)$ such that (1) $M_i \cap M_j = \emptyset$ for $1 \le i < j \le d$ and (2) $M_1 \cup M_2 \cup \dots \cup M_d = E(G)$.

11.2 Networks with Vertex Capacities

Suppose we have a directed graph $G = (V, E)$ but instead of *edge capacities* we have *vertex capacities* $c : V \to \mathbb{R}$. Now a flow f should observe the *vertex capacity constraints*, i.e., the outflow from a vertex u should not exceed $c(u)$:

$$\forall u \in V : \sum_{v \in V, f(u,v) > 0} f(u, v) \le c(u) \ .$$

Exercise 11.4. Consider networks with vertex capacities.

1. Show how to model networks with vertex capacities by networks with edge capacities. More precisely, show how to transform $G = (V, E, c)$ with $c : V \to \mathbb{R}^+$ into a network $G' = (V', E', c')$ with $c' : E' \to \mathbb{R}^+$ such that every s-t-flow f in G that respects the vertex capacities corresponds to an s-t-flow f' (of same value) in G' that respects edge capacities, and vice versa.

2. Draw a picture illustrating your solution.

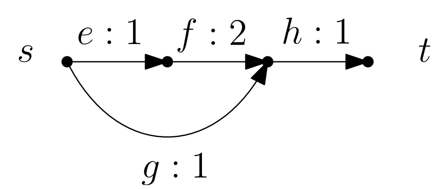
3. Show that there is a polynomial time algorithm solving the following problem: Given a directed graph $G = (V, E)$ and two vertices $s, t \in V$. Are there k paths $p_1, \dots, p_k$, each from s to t, such that the paths are *internally vertex disjoint*? Here, internally vertex disjoint means that for $i \neq j$ the paths p_i, p_j share no vertices besides s and t.

Exercise 11.5. Let H_n be the n-dimensional Hamming cube. For $i < n/2$ consider L_i and L_{n-i}. Note that $|L_i| = \binom{n}{i} = \binom{n}{n-i} = L_{n-i}$, so the L_i and L_{n-i} have the same size. Show that there are $\binom{n}{i}$ paths $p_1, p_2, \dots, p_{\binom{n}{i}}$ in H_n such that (i) each path p starts in L_i and ends in L_{n-i}; (ii) two different paths p, p' do not share any vertices.

11.3 Always, Sometimes, or Never Full

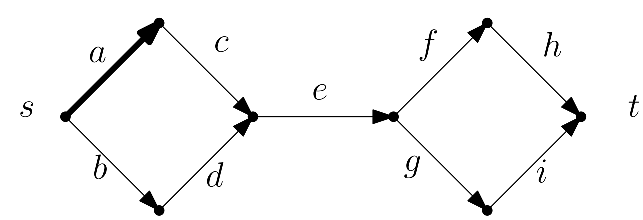
Let (G, s, t, c) be a flow network, $G = (V, E)$. A directed edge $e = (u, v)$ is called always full if $f(e) = c(e)$ for every maximum flow; it is called sometimes full if $f(e) = c(e)$ for some but not all maximum flows; it is called never full if $f(e) < c(e)$ for all maximum flows.

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; sometimes 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 sometimes full and never crossing; f is never full and never crossing; h is always full and always crossing.

Exercise 11.6. Consider this network:

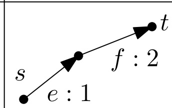
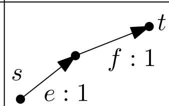


The fat edge a has capcity 2, all other edges have capacity 1.

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

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

Exercise 11.7. An edge e can be (x) always full, (y) sometimes full, (z) never full; it can be (x') always crossing, (y') sometimes crossing, (z') never crossing. So there are nine possible combinations: (xx') always full and always crossing, (xy') always full and sometimes 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: sometimes full	z: never full
x': always crossing		Possible or impossible?	Possible or impossible?
y': sometimes 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.

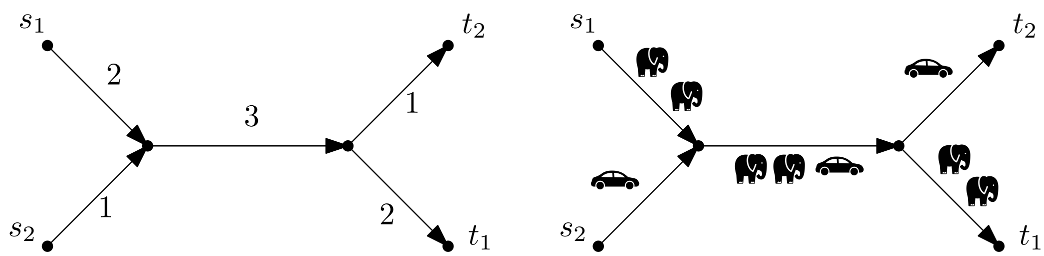
The two very simple flow networks in the table already show that (xx') and (yy') are possible; that is, it is possible to be always full and always crossing, and it is possible to be always full and sometimes crossing. Fill out the table! That is, for each of the remaining seven cases, find out whether 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.

11.4 Multi-Commodity Flow

In class, we discussed the Multi-Commodity Flow problem. Formally, a multi-commodity network is given by a directed graph $G = (V, E)$, a capacity function $c : E \to \mathbb{R}^+$, and sources $\vec{s} = (s_1, \ldots, s_k)$ and sinks $\vec{t} = (t_1, \ldots, t_k)$ in V. A *multi-commodity flow* in $(G, c, \vec{s}, \vec{t})$ is a tuple $(f_1, \ldots, f_k)$ where each f_i is an s_i-t_i-flow in (G, c, s_i, t_i), that is, f_i is individually a flow, satisfying flow conservation constraints at all vertices except s_i and t_i; the *value* of f_i, $\mathrm{val}(f_i)$, is the outflow at s_i, as usual. Furthermore, $\sum_{i=1}^{k} f_i(e) \le c(e)$ for all edges $e \in E$. Think of each f_i as being a way to route units of good i from s_i to t_i.

The *Maximum Multi-Commodity Flow Problem* (Max-MCF) asks for a multi-commodity flow in the network maximizing the total value, i.e., $\mathrm{val}(f_1) + \cdots + \mathrm{val}(f_k)$.

In the *Feasibility Multi-Commodity Flow Problem* (F-MCF), we are additionally given *demands* $d_1, \ldots, d_k$, and we want to decide whether there is a multi-commodity flow $(f_1, \ldots, f_k)$ with $\text{val}(f_i) = d_i$. If there is such a multi-commodity flow, we say the instance of F-MCF is *feasible*, otherwise we say it is *infeasible*.



A multi-commodity flow routing two elephants from s_1 to t_1 and one car from s_2 to t_2. Note that the two flows share the middle edge; also, all flow values and all capacities are integers.

For each of Max-MCF and F-MCF we can define the *integer* version: Max-IMCF and F-IMCF. These are the same problems as above, but we additionally require that all capacities, demands, and flow values be integers.

Exercise 11.8. Find a multi-commodity flow network with integer capacities such that Max-MCF is larger than Max-IMCF. That is, to achieve the maximum possible flow, it is necessary to use non-integral flows.

Exercise 11.9. Find a multi-commodity flow network with integer capacities and integer demands such that the F-MCF problem is feasible but the F-IMCF problem is infeasible. That is, it is possible to satisfy all the demands, but not all flows must be integer.