# Mathematical Foundations
of
Computer Science

CS 499, Shanghai Jiaotong University, Dominik Scheder

11   Matchings and Network Flow

- Homework assignment published on Monday 2018-05-14

- Submit questions and first solutions by Sunday, 2018-05-20, 12:00

- Submit final solution by Sunday, 2018-05-27.

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 kth 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 ($n = 3, k = 1$):
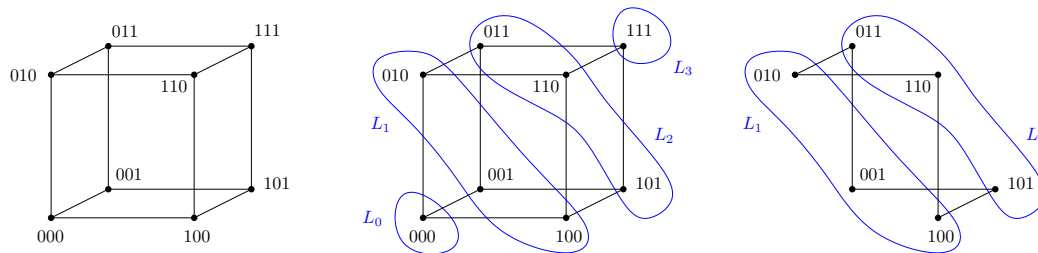
Exercise 11.1. Let $0 \leq k < n/2$. Show that the bipartite graph $H_n[L_k \cup L_{k+1}]$ has a matching of size $|L_k| = \binom{n}{k}$.

Proof. Since $|\Gamma(L_k)| = \binom{n}{k+1} \geq |L_k|$, $\delta(L_k) \leq 0$. According to Hall's Therorem, we can find matchings satisfying $M(A) = |L_k| - \delta(A)$. Then H_n has a matching of size $\binom{n}{k}$. $\qquad\square$

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 $|L|$).

Proof. Since G is d-regular, we have $|\Gamma(L_k)| = |R| = |L|$. Also according to Hall's Therorem, we can find a matching with size $|L| - \delta(L) = |L|$, which is a perfect matching. $\qquad\square$

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 $M_1, \ldots, M_d \subseteq E(G)$ such that (1) $M_i \cap M_j = \emptyset$ for $1 \leq i < j \leq d$ and (2) $M_1 \cup M_2 \cup \cdots \cup M_d = E(G)$.

Proof. We can prove it by induction.
Base case: When $d = 1$, we can easily find a perfect matching satisfying two conditions.
Induction step: Assume any k-regular bipartite graph G can be partitoned into k perfect matchings. Those matchings are denoted as $M_1, \ldots, M_k \subseteq E(G)$. Then we increase the degree of each vertex by 1 by adding $|L|$ edges between L and R. We get a new $k+1$-regular bipartite graph G'
The newly added edges consist a perfect maching denoted as M_{k+1}. Obviously, $M_i \cap M_j = \emptyset$ for $1 \leq i < j \leq k+1$ and $M_1 \cup M_2 \cup \cdots \cup M_{k+1} = E(G')$
Since we can construct and $k+1$-regular bipartite graph from k-regular bipartite graph, we can always show that $E(G)$ can be partitionedinto d perfect matchings content with the conditions. $\qquad\square$

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 $c(u)$:

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

**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.

   Solution. For any vertex v in G, we replace it with two vertexs $v_i n$ and $v_o ut$ and edge $(v_i n, v_o ut)$ with capacity equal to $c(v)$.
   Then for any edge (u, v) in G, we replace it with $(u, v_i n)$ and for any edge (v, u) in G, we replace it with $(v_o ut, u)$. Those edges have unlimited capacity. ∎

2. Draw a picture illustrating your solution.

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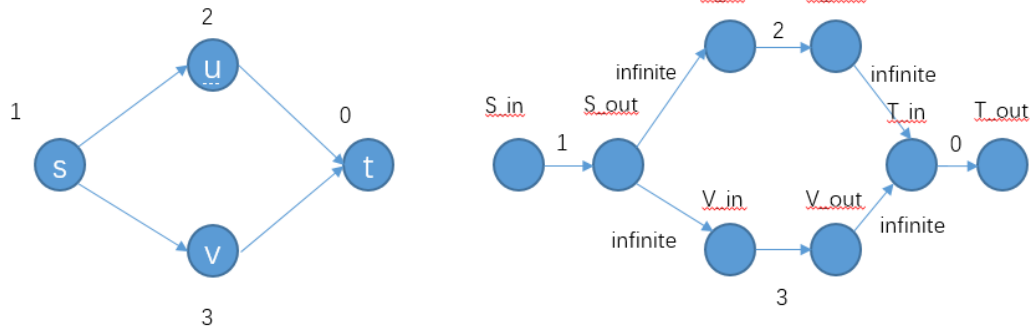
3. Show that there is a polynomial time algorithm solving the following problem: Given a directed graph $G = (V, E)$ and two vertices $s, t \in V$. Are there k paths $p_1, \ldots, p_k$, each from s to t, such that the paths are *internally vertex disjoint*? Here, internally vertex disjoint means that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

   Proof. Let each vertex capacity be one. Then there are k internally vertex disjoint paths if and only if there is a flow with size k.
   Since we can transform a model with vertex capacity to a network

with edge capacity, we can use Ford-Fulkerson algorithm to find the maximum flow in polynomial time. Then we can solve this problem in polynomial time. □

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 $p_1, p_2, \ldots, p_{\binom{n}{i}}$ 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.

Proof.   1. Let each vertex capacity be one.

2. Add source vertex and target vertex with unlimited capacity, connect source vertex to all vertexs in L_i and all vertexs in L_{n-i} to target vertex.

3. Transform the vertex capacity network to edge capacity network, then we can use Ford-Fulkerson Algorithm to find the flow of network is $\binom{n}{i}$, which means there are $\binom{n}{i}$ content with the two conditions.

□

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:
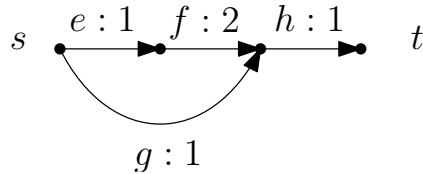
Example network: the edges e, g are sometimes full and never crossing; f is never full and never crossing; h is always full and always crossing.
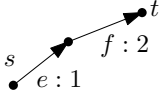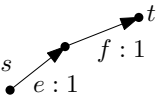
Exercise 11.6. Consider this network:
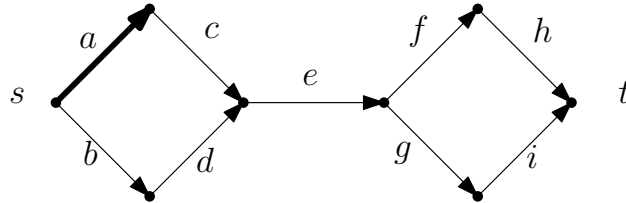
The fat edge a has capcity 2, all other edges have capacity 1.

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.

solution

1. Always full:e; Sometimes full:b, c, d, f, g, h, i; Never full:a.

2. Always crossing:e; Sometiems crossing:None; Never crossing:a, b, c, d, f, g, h, i.

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.