CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Lecture 15: Matchings

Harvard SEAS - Fall 2021 *Oct. 26, 2021*

1 Announcements

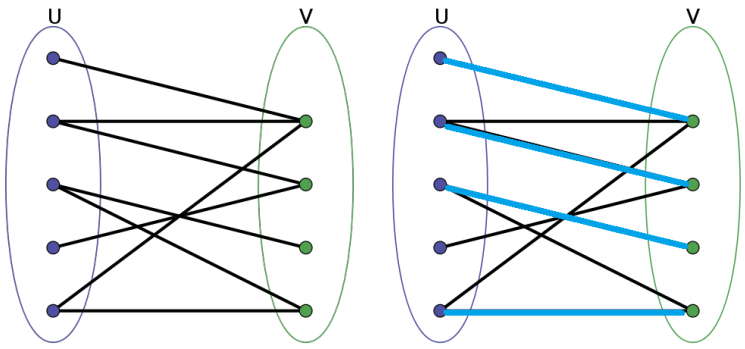
- Please fill out the mid-semester feedback form! Up on Ed

2 Definitions

Motivating Problem: Kidney Exchange. Collection of patients (who need a kidney) and donors (willing to donate a kidney). Each donor can only donate one kidney (they need their other one to survive!) and only to certain patients (due to blood type and HLA type compatibilities). This is a large-scale real-world problem, in which algorithms like what we will cover play a significant role. There over 100,000 patients currently on the kidney waiting list in the US, with a little over 10,000 donations happening per year, and patients spending an average of about 3.5 years on the waiting list.

How many patients can we give kidneys to?

Q: How to formulate graph-theoretically?
Create a graph with $v_1, \ldots, v_t$ representation potential donors, and $w_1, \ldots, w_l$ representing potential recipients. Then, for v_i, w_j we place an edge connecting the two if they are compatible.



Definition 2.1. For a graph $G = (V, E)$, a *matching* in G is a subset $M \subseteq E$ such that every vertex $v \in V$ is incident to at most one edge in M. Equivalently, no two edges in M intersect.

Input	: A graph $G = (V, E)$
Output	: A matching $M \subseteq E$ in G of maximum size

Computational Problem Maximum Matching

A **matching** is a set $M \subseteq E$ where no two edge in M share an endpoint.
Additional considerations in real-life kidney exchange (beyond scope of this course):

- Priority: We may want to give higher priority to some patients, like those who have been waiting longest, or have more advanced kidney disease, or are organ donors themselves.
- Donor preference: most kidney donors sign up because they have a loved one who needs a kidney, but aren't compatible with each other. So they are only willing to donate their kidney if their loved one receives a kidney from someone else.
- Chains: due to the donor preference above, and laws disallowing contracts around organ donations, a donor may only want to do their donation if it is nearly simultaneous with their loved one receiving a kidney. This leads to simultaneous or near-simultaneous surgeries, with the longest chain to date (Dec 2020) involving 35 donations (70 surgeries) over multiple locations.
- Location : It is easiest and most cost effective if the exchange happens in the same hospital, though it has become common to ship the kidneys on ice from the donor's hospital to the patient's hospital!

3 Matching vs. Independent Sets

Maximum Matching can be viewed as a special case of the Independent Set problem we studied last time, i.e. there is an efficient reduction from Maximum Matching to Independent Set:

Given a matching instance $G = (V, E)$, want to reduce to an independent set instance $G' = (V', E')$. In max matching, we want to find a set of edges that don't conflict, whereas independent set tries to find a set of vertices that don't conflict. Thus, let

$$V' = E \quad \text{and} \quad E' = \{\{e, e'\} : e, e' \text{ share an endpoint}\}.$$

(The graph G' is known as the *line graph* of G.)

Unfortunately, the fastest known algorithm for Independent Set runs in time approximately $O(1.2^n)$. However, as we saw last time for IntervalScheduling-Optimization, special cases of IndependentSet can be solved more quickly. Matching is another example!

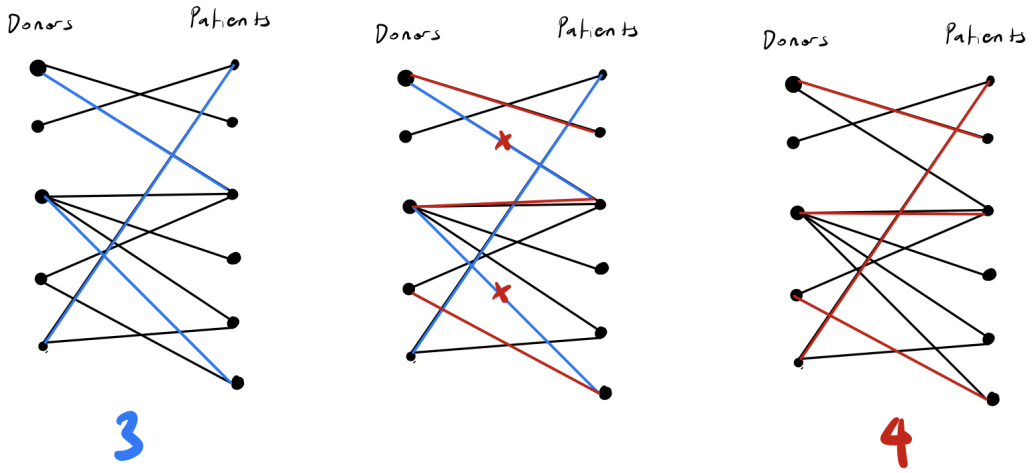
4 Maximum Matching Algorithm

Like in a greedy strategy, we will try to grow our matching M on step at a time, building a sequence $\emptyset = M_0, M_1, M_2, \ldots$, with $|M_k| = k$. However, to get M_k from M_{k-1} we will do more sophisticated operations than just adding an edge.

Definition 4.1. Let $G = (V, E)$ be a graph, and M be a matching in G. Then:

1. An *alternating path* P in G with respect to M is a sequence of vertices $v_0, v_1, \ldots, v_\ell$ such that for every $i = 1, \ldots, \ell - 1$, $\{v_{i-1}, v_i\} \in M \Leftrightarrow \{v_i, v_{i+1}\} \in E - M$.
2. An *augmenting path* P in G with respect to M is an alternating path in which v_0 and v_ℓ are unmatched by M, and in which all of the vertices on the path are distinct.

Example of an augmenting path:



Lemma 4.2. *Given a graph $G = (V, E)$, a matching M, and an augmenting path P with respect to M, we can construct a matching M' with $|M'| = |M| + 1$ in time $O(n)$.*

Proof.
Let $P = (v_0, \ldots, v_\ell)$ be an augmenting path. We have that $\{v_0, v_1\} \in E \setminus M$ since v_0 is unmatched, and $\{v_{\ell-1}, v_\ell\} \in E \setminus M$ since v_ℓ is unmatched. Thus, let

$$M' = M \cup \{\{v_0, v_1\}, \{v_2, v_3\} \ldots, \{v_{\ell-1}, v_\ell\}\} \setminus \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{\ell-2}, v_{\ell-1}\}\}.$$

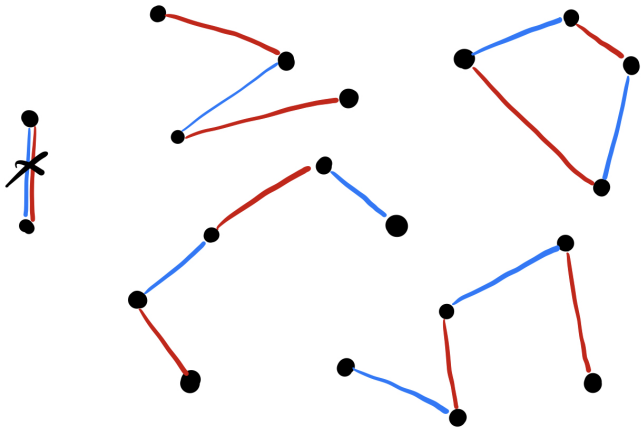
We can show that $|M'| = |M| + 1$ via a counting argument. To show M' is a matching, note that v_0 only appears in the first edge when we insert it into our matching (since the path has no duplicate vertices). Since v_1 was only connected to one prior edge $\{v_1, v_2\}$ (since M was a matching), removing it must ensure $\{v_0, v_1\}$ can be added to M. Then an equivalent argument holds for the other vertices in the path.

□

So augmenting paths are one way to grow our matching. How do we know they always exist?

Lemma 4.3. *Let $G = (V, E)$ be a graph, and $M \subseteq E$ be a matching. If M is not a maximum-size matching, then G has an augmenting path with respect to M.*

Proof.



Suppose we have a matching M and a larger matching M'. Then consider $G_\Delta = (V, M \Delta M')$ where $M \Delta M'$ contains the edges in *exactly one* of M and M'. Note that G_Δ has degree at most 2. This is since an edge has to come from M or M', and both of those have degree at most 1. Furthermore, every path or cycle in this graph alternates between edges in M and M'. Finally, at least one path must start and end with an edge in M'. This is because M' is bigger. Every cycle in G_Δ contains an equal number of edges from M and M', so there must be some path with more M' edges than M edges, and this is only possible if both endpoints are in M'. But this path is an augmenting path by definition, so we are done. □

How can we find an augmenting path? We will be able to do so in *bipartite* graphs, like the donor–patient graphs in kidney exchange.

Definition 4.4. A graph $G = (V, E)$ is *bipartite* if it is 2-colorable. That is, there is a partition of vertices $V = V_0 \cup V_1$ (with $V_0 \cap V_1 = \emptyset$) such that all edges in E have one endpoint in V_0 and one endpoint in V_1.

Lemma 4.5. *Let $G = (V_0 \cup V_1, E)$ be bipartite and let M be a matching in G that is not of maximum size. Let U be the vertices that are not matched by M, and $U_0 = V_0 \cap U$ and $U_1 = V_1 \cap U$. Then:*

1. *G has an alternating path that starts in U_0 and ends in U_1.*
2. *Every shortest alternating path from U_0 to U_1 is an augmenting path.*

Note this is the only place where we use that the graph is bipartite!

Proof.

1. By Lemma 4.3, we know that G has an augmenting path P, with vertices $v_0, v_1, \ldots, v_\ell$. Since v_0 and v_ℓ are unmatched, the edges $\{v_0, v_1\}$ and $\{v_{\ell-1}, v_\ell\}$ are not in M. Since the path is alternating between edges from $E - M$ and from M, the path must then be of odd length ℓ. Since paths in a bipartite graph alternate between V_0 and V_1, we have either $v_0 \in U_0$ and $v_1 \in U_1$ or $v_0 \in U_1$ and $v_1 \in U_0$. So either P is the path we want, or the reverse of P is the path we want.

2. Let P be a shortest alternating path $v_0, v_1, \ldots, v_\ell$ with $v_0 \in U_0$ and $v_\ell \in U_1$. Suppose P has a repeated vertex, i.e. $v_i = v_j$ for some $i < j$. Since the graph is bipartite, $j - i$ must be even. Thus if we remove the vertices $v_{i+1}, \ldots, v_j$ from the P, the path P will still be alternating (since the portion we remove will either begin with M and end with $E - M$ or vice-versa).

□

With these lemmas, we obtain a matching algorithm: repeatedly search for a shortest alternating path from U_0 to U_1, and use it to grow our matching.

```
MaxMatchingAltPaths(G)
  Input  : A bipartite graph G = (V, E)
  Output : A maximum-size matching M ⊆ E
Remove isolated vertices from G;
Let V_0, V_1 be the bipartition (i.e. 2-coloring) of V;
M = ∅;
repeat
    Let U be the vertices unmatched by M, U_0 = V_0 ∩ U, U_1 = V_1 ∩ U;
    Try to find a shortest alternating path P from U_0 to U_1;
    if P ≠ ⊥ then augment M using P via Lemma 4.2;
until P = ⊥;
return M
```

The correctness of Algorithm 10 follows from Lemma 4.5, so we only need to analyze the run time:

Theorem 4.6. *Maximum Matching can be solved in time $O(mn)$ on bipartite graphs with m edges and n vertices.*