CS120: Intro. to Algorithms and their Limitations Hesterberg & Vadhan

Lecture 13: Matchings

Harvard SEAS - Fall 2022 *Oct. 18, 2022*

1 Announcements

- Embedded EthiCS Module on Thursday. You are expected to attend; there will be material on ps6 building on the module.
- One extra late day for everyone! (Rationale: some students missed PP1 due date, many had Stat110 midterm on PS4 due date.)
- Responses to midterm survey feedback
  - Typical (mean, median, mode) time spent on class is approx 15hrs/week including everything (lecture, section, etc.). More reasonable than first weeks, but we'll keep looking for ways to reduce without compromising learning objectives.
  - Typical time spent preparing for midterm approx 10hrs, which is reasonable. Midterm was short so one confusion could have big impact; it is less than 10% of final grade.
  - Most students found midterm to be reasonable in length, difficulty, and coverage, but some questions could have been worded more clearly or offered more opportunities for partial credit.
  - Rationale for grading scheme: give everyone opportunities for revisions (time-consuming to grade revisions on a point system), focus on learning objectives rather than points. An R- is a good grade!
  - All topics covered so far were found to be interesting/relevant, and most course elements found useful for learning (except participation portfolios and in-class exercises).
  - Problem expectations on hws can be made clearer. Note taken!
  - Goal of Participation and Learning Portfolios: Get you to be actively reflecting on what types of engagement are most effective for your and your classmates' learning. We will continue to tweak based on what we see in your submissions and feedback.
  - Some want more sophisticated programming problems: Programming skills are not a learning objective of cs120 per se (and many theory classes have zero programming), but we will look for ways to give you opportunities (e.g. extra credit) to stretch your programming skills.
  - Sections and Office Hours receive top ratings for contributing to learning (take advantage if you aren't already!), and Classmates and TFs for supporting your learning.
  - Lectures a bit fast. The course content has not increased (we have cut things), but we did intentionally increase the pace in the first half so that the second half (starting now!) will be less rushed. Will continue looking for ways of inserting checkpoints.
- Participation Portfolio Highlights II due this Sat. 10/22.

- Shafi Goldwasser Distinguished Lecture Thurs

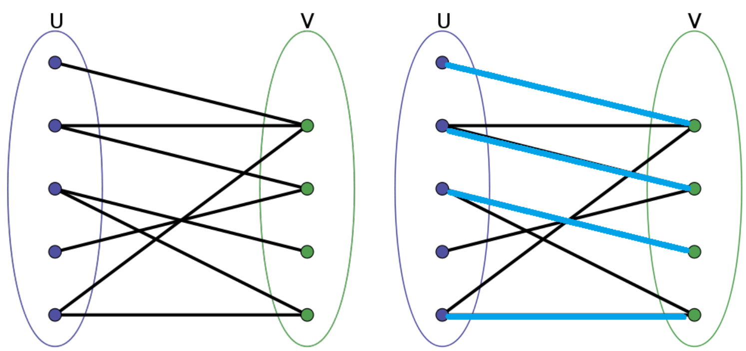
Recommended Reading: Cormen–Leiserson-Rivest–Stein, Sec. 25.1.

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 nearly 100,000 patients currently on the kidney waiting list in the US, with a little over 25,000 donations happening per year, and patients spending an average of about 3.6 years on the waiting list.

How many patients can we give kidneys to?

Q: How to formulate graph-theoretically?
Create a graph with $u_1, \ldots, u_t$ representation potential donors, and $v_1, \ldots, v_l$ representing potential recipients. Then, for u_i, v_j we place an (undirected) 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.[1] Equivalently, no two edges in M share an endpoint.

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

Computational Problem Maximum Matching

Additional considerations in real-life kidney exchange (to be discussed more in Embedded EthiCS Module on Thurs!):

- 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.

[1] Saying a vertex v is *incident* to an edge e is another way of saying v is an endpoint of e. It is more symmetric, in that we would also say that e is incident to v.

- 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 walk* W in G with respect to M is a walk $(v_0, v_1, \ldots, v_\ell)$ in G 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 walk in which v_0 and v_ℓ are unmatched by M, and in which all of the vertices in the walk are distinct.

Let's see why augmenting paths are useful.

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

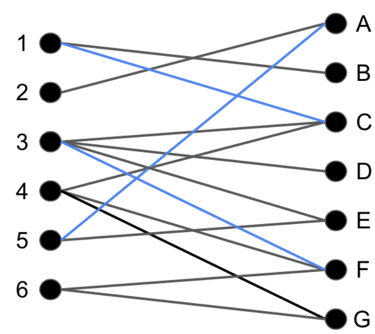
$$M' = (M - \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{\ell-2}, v_{\ell-1}\}\}) \cup \{\{v_0, v_1\}, \{v_2, v_3\} \ldots, \{v_{\ell-1}, v_\ell\}\}.$$

In words, we "flip" each of the edges in this augmenting path: any edge of the augmenting path originally in the matching M is now not in the matching M', and any edge of the augmenting path not in M is in M'. 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.

□

Example:

For example, consider the following graph with $M = \{\{1, C\}, \{3, F\}, \{5, A\}\}$.



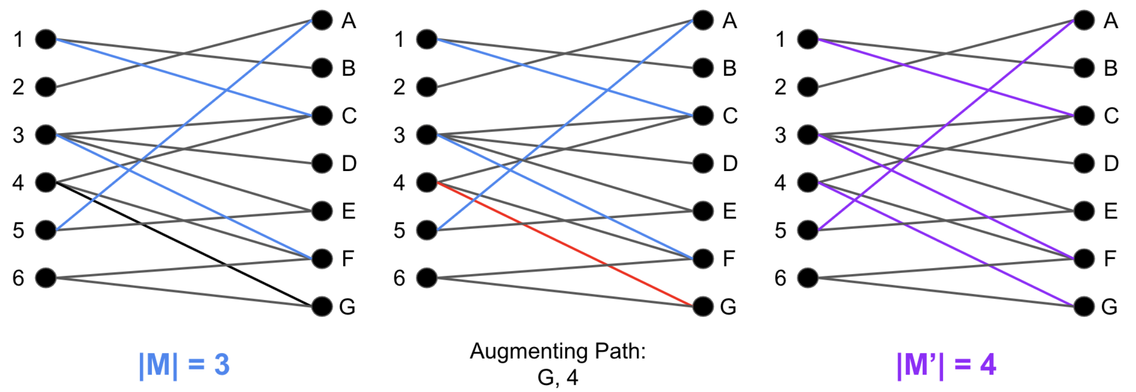
|M| = 3

We check whether the following sequences of vertices constitute an alternating walk or an augmenting path.

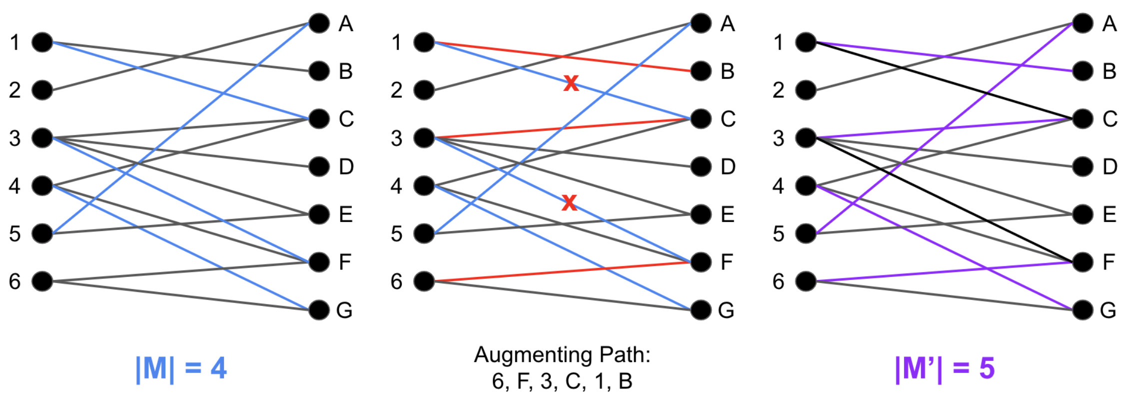
why choose this particular ordering of paths?ß

Path	Alternating Walk	Augmenting Path
$2, A, 5$	Yes	No
$G, 4$	Yes	Yes
$3, F, 4, C, 3$	No	N/A
$1, C, 6$	No	N/A
$6, F, 3, C, 1, B$	Yes	Yes

Now, we can try running our maximum matching algorithm on the graph. We first consider the augmenting path $(G, 4)$ and grow our matching accordingly.



Next, we add the augmenting path $(6, F, 3, C, 1, B)$. Note that $\{3, F\}$ and $\{1, C\}$ are not in M', since they were part of the original matching M.



This suggests a natural algorithm for maximum matching: repeatedly try to find an augmenting path and use it to grow our matching. We will be able to make this idea work in *bipartite* graphs, like the donor–patient graphs in kidney exchange.

Definition 4.3. 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.

`MaxMatchingAugPaths(`G`)`
Input : A bipartite graph $G = (V, E)$
Output : A maximum-size matching $M \subseteq E$
Remove isolated vertices from G;
Let V_0, V_1 be the bipartition (i.e. 2-coloring) of V;
$M = \emptyset$;
repeat
  Let U be the vertices unmatched by M, $U_0 = V_0 \cap U$, $U_1 = V_1 \cap U$;
  Try to find an augmenting path P that starts in U_0 and ends in U_1;
  if $P \neq \perp$ **then** augment M using P via Lemma 4.2;
until $P = \perp$;
return M

How do we know that augmenting paths always exist and how can we find them efficiently?

Theorem 4.4 (Berge's Theorem)**.** *Let $G = (V, E)$ be a graph, and $M \subseteq E$ be a matching. If (and only if) M is not a maximum-size matching, then G has an augmenting path with respect to M.*

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 walk with respect to M that starts in U_0 and ends in U_1.* ???

2. *Every shortest alternating walk from U_0 to U_1 is an augmenting path.*

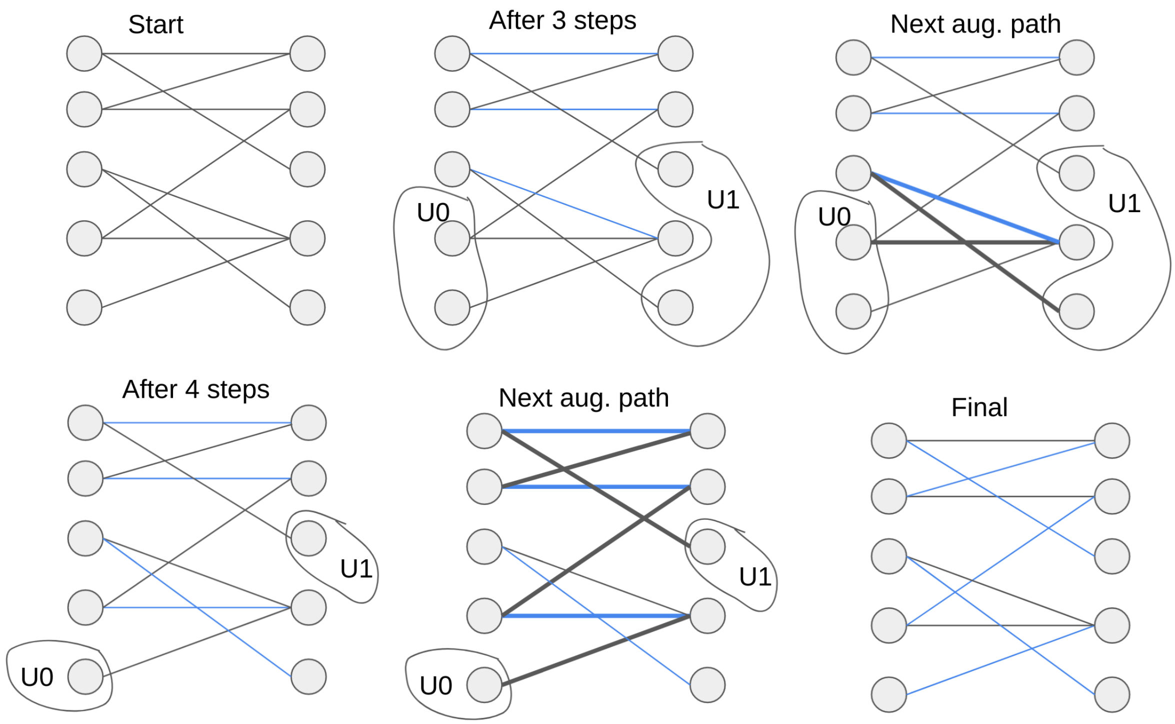
Before proving these lemmas, let's see how they suffice for us to analyze the correctness and runtime of Algorithm 10.

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

Proof.
Lemma 4.5 implies the correctness of Algorithm 10. If M is not a maximum matching, then there is an augmenting path from U_0 to U_1, which we will find and use to increase the size of M by 1. Since a matching can have at most $n/2$ edges, Algorithm 10 will always halt within $n/2$ iterations of the loop, and when it does, it will output a maximum-size matching (since no augmenting path from U_0 to U_1 exists).

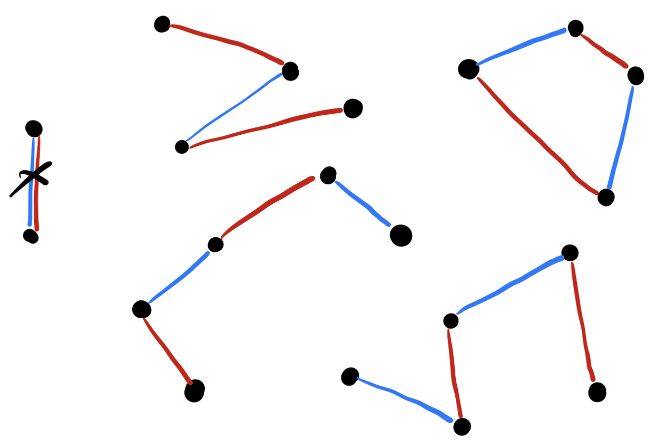
For runtime, Lemma 4.5 implies that we can find augmenting paths by finding shortest alternating walks. An alternating walk is a special case of a 2-rotating walk (PS4), and thus a shortest alternating walk can be found in time $O(m)$. (After removing isolated vertices, we have $n \leq 2m$, so $O(n + m) = O(m)$.) Augmenting the matching M using the augmenting path P takes time $O(n) \leq O(m)$ by Lemma 4.2. Since we already argued that the loop is executed at most $n/2$ times, our total run time is $(n/2) \cdot O(m) = O(mn)$. $\square$

Example:



Let's now prove our correctness lemmas.

Proof of Theorem 4.4.



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.

The “only if” part of Theorem4.4 follows from Lemma 4.2. ☐

Proof of Lemma 4.5.

1. By Theorem 4.4, 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 or the reverse of P is an alternating walk starting in U_0 and ending in U_1.

2. Let W be a shortest alternating walk $v_0, v_1, \ldots, v_\ell$ with $v_0 \in U_0$ and $v_\ell \in U_1$. Assume for sake of contradiction that W 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 W, the path W will still be alternating (since the portion we remove will either begin with M and end with $E - M$ or vice-versa), but will be of shorter length. This contradicts our hypothesis that W was a shortest alternating walk. Thus, our assumption that W has a repeated vertex must have been incorrect.

☐

Note that this lemma is the only place where we used the assumption that the graph is bipartite.