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?

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

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

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:

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

2. An *augmenting path* P in G with respect to M is

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.

□

Example:

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.

```
1  MaxMatchingAugPaths(G)
   Input  : A bipartite graph G = (V, E)
   Output : A maximum-size matching M ⊆ E
2  Remove isolated vertices from G;
3  Let V_0, V_1 be the bipartition (i.e. 2-coloring) of V;
4  M = ∅;
5  repeat
6  |  Let U be the vertices unmatched by M, U_0 = V_0 ∩ U, U_1 = V_1 ∩ U;
7  |  Try to find an augmenting path P that starts in U_0 and ends in U_1;
8  |  if P ≠ ⊥ then augment M using P via Lemma 4.2;
9  until P = ⊥;
10 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.

□

Example:

Let's now prove our correctness lemmas.

Proof of Theorem 4.4.

□

Proof of Lemma 4.5.

□

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