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

Lecture 13: Matchings

Harvard SEAS - Fall 2023 *2023-10-19*

1 Announcements

- Three extra late days for everyone.
- Final exam can replace midterm for purposes of getting to a satisfactory grade.
- Embedded EthiCS Module on Thursday. You are expected to attend; there will be material on ps6 building on the module.
- Please fill out midterm survey (comes with pset 4 survey).

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

2 Loose ends from lecture 12

Theorem 2.1. *If the input intervals are sorted by*
then we have that `GreedyIntervalScheduling(`x`)` *will find an optimal solution to IntervalScheduling-Optimization, and can be implemented in time* $O(n \log n)$*.*

Proof.

□

3 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 3.1. 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!):

[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.

4 Matching vs. Independent Sets

This section will not be covered in lecture, and is optional (but recommended) material.

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!

5 Maximum Matching Algorithm

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

Definition 5.1. 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 5.2. *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.

$\square$

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 5.3. 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 ⊆ 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 an augmenting path P that starts in U_0 and ends in U_1;
    if P ≠ ⊥ then augment M using P via Lemma 5.2;
until P = ⊥;
return M
```

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

Theorem 5.4 (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 5.5. *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 5.6. *Maximum Matching can be solved in time $O(mn)$ on bipartite graphs with m edges and n vertices.*

Proof.

$\square$