CMSC451 (Summer 2023) Lecture 1 Notes

Instructor: Nathaniel Grammel

1 Introduction and Course Overview

Course Website: `https://www.cs.umd.edu/class/summer2023/cmsc451/`

Grades will be based off homeworks and exams. There will typically be 1–2 homeworks per week. The tentative grading breakdown is 40% for homeworks and 60% total across both exams. Note that specifics such as topics covered, grading breakdown and number of homeworks is subject to change. The most up-to-date information about the course can be found on the course website.

We will use Piazza extensively. You are expected to be signed up for the course Piazza in order to receive updates and to discuss topics from the course. Please email me if you don't have access to the Piazza, and I will add you.

NB: These notes are not meant to be a replacement for the lecture recording. They are simply meant to provide an easy-to-read typed version of what was written on the whiteboard. Please watch the lecture recording to see the full discussion from class.

2 Stable Matching

We are given sets $\mathcal{C}$ and $\mathcal{S}$, representing, e.g., companies and students. We assume that $|\mathcal{C}| = |\mathcal{S}|$, i.e. that there are the same number of companies as students. We want to match students to internships at companies, so that every student has an internship and every company has a student intern.

Definition 1 (Matching)**.** Given sets $\mathcal{C}$ and $\mathcal{S}$, a *matching* is a set of pairs $\{(c, s) \mid c \in \mathcal{C}, s \in \mathcal{S}\}$ such that each element of $\mathcal{C}$ appears in at most one pair and likewise each element of $\mathcal{S}$ appears in at most one pair.

Definition 2 (Perfect Matching)**.** Given sets $\mathcal{C}$ and $\mathcal{S}$, a *perfect matching* is a matching where each element of $\mathcal{C}$ appears *exactly* once.

In the internship matching problem we described above, we desire a *perfect matching* (i.e., each company gets exactly one intern, and each student gets exactly one internship). Notice that this is only possible if $|\mathcal{C}| = |\mathcal{S}|$ (otherwise, some student or company won't be able to be matched).

2.1 Stable Matching Problem Statement

In the stable matching problem, we are given:

- A set $\mathcal{C}$ of n companies
- A set $\mathcal{S}$ of n students
- For each company $c \in \mathcal{C}$, a *ranking* of all of the students in $\mathcal{S}$
- For each student $s \in \mathcal{S}$, a *ranking* of all of the companies in $\mathcal{C}$

The ranking is just an ordering of the elements of the set: for a company's ranking of $\mathcal{S}$, they order the elements with the most preferred student first, then their second choice, etc., and similarly for the students' rankings of $\mathcal{C}$.

Note: While the sets are each of size n, the size of the input is more like $2n + 2n^2$, since each of the $2n$ elements also gives a ranking of the n elements of the other set.

We can't necessarily give every student their first choice for company (e.g., if all students pick the same company as their top choice). Similarly, we may not be able to guarantee every company will get their favorite candidate among the students. Instead, we can look for a *stable* matching.

Definition 3 (Stable Matching)**.** A matching is *unstable* if there exists a pair (c, s) not in the matching such that c prefers s to its match and s prefers c to its match. A matching is *stable* if it is not unstable (i.e., there does *not* exist such a pair).

Algorithm 1 Gale-Shapley (1962)

```
while there exists a company c with no student do
    let s be the highest-ranking (i.e., most preferred) student to whom c has not yet made an offer
    if s is free (i.e., not engaged) then
        (c, s) become engaged
    else
        let c' be the company to which s is engaged
        if s prefers c over c' then
            (c, s) become engaged
            c' becomes free (unengaged)
        end if
    end if
end while
```

It is not obvious that the Gale-Shapley algorithm is correct. It's not even obvious that it terminates (i.e., how do we know that we don't just keep re-assigning companies forever?) However, we can prove that it does in fact terminate, and that, when it does, the result is a stable matching. We will prove the following:

Theorem 1. *The Gale-Shapley Algorithm terminates after at most n^2 iterations of the while loop, and produces a stable matching.*