CMPS 102 — Quarter Spring 2017 – Homework 1

VLADOI MARIAN

April 21, 2017

I have read and agree to the collaboration policy.Vladoi Marian
I want to choose homework heavy option.
Name of students I worked with: Victor Shahbazian and Mitchell Etzel

Solution to Problem 1: Resident Matcing

Google Matching Algorithm: We have n students applying for an intership at one of Google Team. We have m Google Teams , each with a fixed capacity, say p capacity. Each team had a ranking of the students in order of preference, and each student had a ranking of the teams in order of preference. Assume that there were more students who want an internship at Google than there were slots available in the teams * capacity of the teams (m*p).We keep a min heap , representing the team , so we willbe able to add a new student in a team , or to remove a student from a team. The students do not have to play games. They rank the teams in order of preference even if they know that have no change to get to that team. Ranking decision should be private. The algorithm is both Students and Teams Friendly, more friendly to students because the core of the algorithm plays around the student' s preference list. Matching a student and a team is called Tentative. A student might be tentatively matched with a team, and if he will be unmatched during the algorithm , an attempt is made to rematch the aplicant until his preference list will be exhausted. He might remain unmatched. At the beginig of the algorithm all students are free and all teams are empty. We keep all the free students in a Queue named Q, and all the matched students that are tentatively matched in Q complement $\complement Q$.

The pseudocode for the Google Matching Algorithm:

```
while (Q with students is not empty)
And (The Student's rank ordered list from Q is not exausted) do
    Choose one student s from the Q
    for (Check the rank ordered List of students s ) do
        Choose first Team t , from the ranked ordered List of students s
        if (The Google Team t has also ranked the Student s) then

            if (The Google Team t has an unfilled possition) then
                Assign the Student s and Google Team t as a Tentative Match
                Student s is now in $\complement Q$
            else if (Another Student tentatively matched to that Team can be unmatched) then
                Un-assign the Student x ranked lower by the Google Team
                Push the un- assign Students x from $\complement Q$ back to Q
                Assign the Google Team and Student s as tentative match.
                Student s is now in $\complement Q$
            end if
        end if
    end for
end while Return the set $\complement Q$ of matched teams and students.
```

The main ideas:

(1.1) Google teams are empty at the beginning of the algorithm. The teams have p capacity. The teams will accept all the tentative matching until it fill all the empty spots. It is guaranteed that the teams will fill the position because there are more students than teams positions. Google Teams, once it reached the capacity , remains with full capacity from the point at which it receives p+1 capacity proposal; and the sequence of students that it accept from that moment gets better and better (in terms of its preference list).

(1.2) The sequence of teams to whom s apply gets worse and worse (in terms of his preference list).

(1.3) The running time of the Google Matching Algorithm $O((n+k)*m)$ Assuming n students that apply for intership and m Google Teams. The outer $while$ loop will iterate at least for every student, and the inner for loop will iterate at most m times (this happens in the situation when all students were considered and they ranked all teams). The running time can be calculated O(n*m). Because we are talking about tentative matching, each time a student is unmatched, the outer $while$ loop has to repeat for the student unmatched. Consider there are k tentative matching canceled (to remove or to add a student is log p) the running time will be $O((n+k)*m)$ which can be considered quadratic time performance $O(n^2)$. For the Space complexity we have n*m matrix, m*n matrix , and each team has a priority queue data structure. Considering that the capaciy of all teams can not be bigger that number of students, $(pm <= n, m = \frac{n}{p})$, the total space complexity is $f(n) = nm + n = n(m+1) = n(\frac{n}{p}+1) = \frac{n^2}{p} + n = O(n^2)$.

(1.4) If s is free at the end of this algorithm,then it means that all Google Teams ranked highter other applicants than him.

(1.5) The set $\mathbb{C}Q$ returned at termination is a perfect matching. This means that the teams have chosen the best aplicants considering their's ordered ranking List. Let us suppose that the algorithm terminates with a free students s that is ranked higher in at least one of Google Team. At termination of outer while loop, it must be the case that s had already proposed to each team from his ordered ranking list. So if the Google team also ranked him higher, then will be accepted, which is a contradiction.

(1.6) Consider an execution of the Google Matching algorithm that returns a set of pairs $\mathbb{C}Q$. The set $\mathbb{C}Q$ is a stable matching.We have already shown in (1.5) that $\mathbb{C}Q$ is a perfect matching. We will proove this by contradiction. Assume that is an instability in $\mathbb{C}Q$.

This type of instability that can occur is that there are teams t_0 and t_1 and students s_0 and s_1 so that:

- s_0 is matched with t_0, and s_1 is matched with t_1, and
- t_0 favors s_1 over s_0, and s_0 favors t_0 over t_1.

In the execution of the algorithm that produced $\mathbb{C}Q$, s_0's last application was, by definition, to t_0. Then we ask : did s_0 apply to t_1 at some earlier point in this execution? If he didnt, then t_0 must occur higher on s_0's preference list than t_1, contradicting our assumption that s_0 prefers t_1 to t_0 . If he did, then he was rejected by t_1 in favor of some other student s_2, whom t_1 prefers to s_0 . s_1 is one of the student accepted by t_1 so either $s_2 == s_1$ or t_1 prefers its final student s_1 to s_2 either way this contradicts our assumption that t_1 prefers s_0 to s_1 . It follows that $\mathbb{C}Q$, is a stable matching.