Assignment 1

Algorithms Complexity (CIS 522-01)

Javier Arechalde

January 31, 2018

Problem 1. Matching Residents to Hospitals

We have m hospitals, each one of them has certain number of available positions to hire residents.

In a given year n medical students were graduating, each one of them interested in joining one of the hospitals.

Each one of the hospitals had a ranking of the students in order of preference, and each student had a ranking of hospitals in order of preference.

In this problem we will assume that $n > m$.

We are looking to assign each student to at most one hospital, in a way that all available positions in hospitals are filled. As $n > m$ some students may have none hospitals assigned at the end of the algorithm run.

For this problem there will be two types of unstability:

- First type

  There are students s and s', and hospital h.

  - s is assigned to h. - s' is assigned to no hospital. - h prefers s' to s.

- Second type

  There are students s and s' and hospitals h and h'.

  - s is assigned to h - s' is assigned to h' - h prefers s' to s - s' prefers h to h'

So the difference between this problem and stable matching problem is that hospitals want more than one student usually, and there is more students that positions available in the hospitals.

In this case I think it makes more sense to start from the students side, checking if there is any positions available in the first hospital they want to get a position in, and if there is one get it, otherwise, check if the hospital prefers this student to the ones that they have assigned already. If the hospital prefer this student to one of the students that they have already assigned, kick the least preferred student, and get this new student in. This algorithm should run while the number of students "free" is different from the number of positions available, and all the students havent been checked yet, because even though, all the positions are filled, we may be running in the first type of unstability, as h prefers s' to s and s'.

Now we will proceed to describe our algorithm structure.

1 Algorithm implementation

H is the set of hospitals. S is the set of students.

m is the total number of hospitals. n is the total number of students.

h_{pref} is the hospital student preference list for each $s \in S$. s_{pref} is the student hospital preference list for each $h \in H$.

$n_{avail}(h_i)$ is the total number of possitions available in each hospital. $n_{assign}(h_i)$ is the total number of students assigned to a hospital.

N_{pos} is the total number of positions available $\sum n_{avail}(h_i)$. S_{assign} is the total number of students that have a hospital assigned.

S_{check} is the total number of students that we tried to assign a hospital to.

Initially all students $s \in S$ and $h \in H$ are free

While $S_{assign} < N_{pos}$ and $S_{check} < n$ and student s hasnt proposed to all $h_i \in s_{pref}$
Try to assign to that student to a hospital in his preference list $h_i(s)$
If $n_{assign}(h_i) < n_{avail}(h_i)$
Student s is assigned to hospital h_i
Else If $n_{assign}(h_i) \geq n_{avail}(h_i)$
Check $h_{pref}(h_i)$
If student s is higher in the list than any student $s_i \in S$
Student s' is now free, and will try to be reassigned in the next iteration
Student s is now assigned to hospital h_i
Else If student s is not higher in h_{pref} than any $s_i \in h_{pref}(h_i)$
Student s remains free
EndIf EndIf EndWhile

Return the set of hospitals and assigned students to each hospital.

Problem 2. Implementation of Propose-and-Reject Algorithms