Problems for Recitation 7

1 A Protocol for College Admission

Next, we are going to talk about a generalization of the stable marriage problem. Recall that we have some horses and we'd like to pair them with stables so that there is no incentive for two horses to swap stables. Oh wait, that's a different problem.

The problem we're going to talk about is a generalization of the one done in lecture. In the new problem, there are N students $s_1, s_2, \ldots, s_N$ and M universities $u_1, u_2, \ldots, u_M$. University u_i has n_i slots for students, and we're guaranteed that $\sum_{i=1}^{M} n_i = N$. Each student ranks all universities (no ties) and each university ranks all students (no ties).

Design an algorithm to assign students to universities with the following properties

1. Every student is assigned to one university.
2. $\forall i$, u_i gets assigned n_i students.
3. There does not exist s_i, s_j, u_k, u_ℓ where s_i is assigned to u_k, s_j is assigned to u_ℓ, s_j prefers u_k to u_ℓ, and u_k prefers s_j to s_i.
4. It is student-optimal. This means that of all possible assignments satisfying the first three properties, the students get their top choice of university amongst these assignments.

The algorithm will be a slight modification of the mating algorithm given in lecture. For your convenience, we have provided a copy of the mating algorithm on the next page.

Our university-student matching algorithm generalizes the mating algorithm:

Each Day:

- Morning:
  - Each ~~girl~~ university stands on her balcony
  - Each ~~boy~~ student stands under the balcony of his favorite ~~girl~~ university whom he has not yet crossed off his list and serenades. If there are no ~~girls~~ universities left on his list, he stays home and does 6.042 homework.
- Afternoon:
  - ~~Girls~~ universities, u_j (where $j \in \{1, \ldots, M\}$) who have at least ~~one~~ n_j suitor say to their favorite n_j from among the suitors that day: "Maybe, come back tomorrow."
  - To the others, they say "No, I will never marry you!"
- Evening:
  - Any ~~boy~~ student who hears "No" crosses that ~~girl~~ university u_j off his list.

Termination Condition: If there is a day when every ~~girl~~ university has at most ~~one~~ n_j suitor, we stop and each ~~girl~~ university ~~marries her~~ accepts current ~~suitor~~ student(s) (if any).

1. Before we can say anything about our algorithm, we need to show that it terminates. Show that the algorithm terminates after $NM + 1$ days. (SEE NEXT PAGE)

2. Next, we will show that the four properties stated earlier are true of our algorithm. To start, let's show the following: if during some day a university u_j has at least n_j applicants, then when the algorithm terminates it accepts exactly n_j students.

3. Next, show that every student is assigned to one university.

4. Next, show that for all i, u_i gets assigned n_i students.

5. Before continuing, we need to establish the following property. Suppose that on some day a university u_j has at least n_j applicants. Define the *rank* of an applicant s_i with respect to a university u_j as s_i's location on u_j's preference list. So, for example, u_j's favorite student has rank 1. Show that the rank of u_j's least favorite applicant that it says "Maybe, …" to cannot decrease (e.g., going from 1000 to 1005 is decreasing) on any future day. Note that u_j's least favorite applicant might change from one day to the next.

6. Next, show there does not exist s_i, s_j, u_k, and u_ℓ where s_i is assigned to u_k, s_j is assigned to u_ℓ, s_j prefers u_k to u_ℓ, and u_k prefers s_j to s_i. Note that this is analogous to a "rogue couple" considered in lecture.

7. Finally, we show in a very precise sense that this algorithm is *student-optimal*. As in lecture, define the *realm of possibility* of a student to be the set of all universities u, for which there exists some assignment satisfying the first three properties above, in which the student is assigned to u. Of all universities in the realm of possibility of a student we say that the student's favorite is *optimal* for that student.

   Show that each student is assigned to its optimal university.

Touches on the right idea, but sloppy / imprecise argument..

1. We show that the algorithm terminates in $NM+1$ days by observing that each of N students has M preferences and that each day the algorithm has yet to produce a stable matching, there exists a student such that this student (or students) crosses a university off their list(s). Hence, the algorithm terminates when no such student exists, which happens once all NM seats are filled. At worst, this takes NM days so the algorithm must terminate on day $NM+1$.

(2) We will show that if a university u_j has at least n_j students, then university u_j will have exactly n_j students when the algorithm terminates.

<u>Proof</u>. (by contradiction.) Suppose that some university u_j ended up with fewer than n_j students at time of algo termination. Call the number of accepted students n^* ($< n_j$). If at any point during run time u_j had at least n_j students vying for acceptance, then $\geq n_j - n^*$ students left in a subsequent day for a higher preference university. This, however, is a contradiction since $n_j - n^*$ students would be asked to come everyday hence and would comply as u_j is their top choice school. Also, u_j couldn't end up with $> n_j$ students at algo termination since the termination condition is, for all M schools, stop if school u_i has $\leq n_i$ suitors. $\square$

③ We will show that every student is assigned to one university.

Proof (by contradiction.): Suppose $\exists s^*$ (a student not accepted to any university). This would mean $\exists u_j^*$ (a university who never had at least n_j students apply) since if u_j^* had $\geq n_j$ students apply, it would've ended with exactly n_j accepted students (SEE QUESTION ②.). But u_j^* would be the top choice for s^* at some point and since u_j^* always had $< n_j$ applicants, u_j^* would've accepted s^* — a contradiction. $\square$

④ We will show $\forall i.\ u_i$ gets assigned n_i students.

Proof: From ③ we know all $N = \sum_{i=1}^{n} n_i$ students get assigned to a university. By the pigeon-hole principle, the only way this is possible is if $\forall i.\ u_i$ gets assigned n_i students. $\square$

(5) We will show that for u_j with at least n_j applicants on day t_0, that the rank of u_j's least favorite applicant that gets invited back never decreases.

<u>Proof</u>: For each day after u_j gets at least n_j applicants, two cases arise:

$\boxed{A}$ the n_j^{th} applicant is still in the running at u_j or

$\boxed{B}$ not.

If $\boxed{B}$, the rank of u_j's least favorite applicant it invites back has improved (not decreased). If $\boxed{A}$, the rank of u_j's least favorite applicant it invites back is unchanged (not decreased).

$\square$

(6) We will show that $\nexists\, s_i, s_j, u_k, u_\ell$ where s_i is assigned to u_k and s_j is assigned to u_ℓ even though s_j prefers u_k (to u_ℓ) and u_k prefers s_j (to s_k), i.e., no rogue couples can exist.

<u>Proof</u> (by contradiction): Suppose for sake of contradiction that such a "rogue" couple existed. This would mean at some point s_j crossed u_k of ~~the~~ list $\Rightarrow$ u_k had $> n_k$ applicants on that day. By (5), s_i would never be accepted on a subsequent day and if s_i showed up on the same day s_j was rejected, s_i would also be rejected since s_j is preferred over s_i by u_k. Thus a contradiction is reached as desired.

$\square$

(7) We will show that our algorithm is student-optimal.

<u>Proof</u>: We begin by establishing the following Lemma, which is a preserved invariant:

* <u>LEMMA</u>: $\forall u, s.$ u is crossed off s's list $\Rightarrow$ u is not in realm of possibility / feasible.

For a given student in a stable matching, it is matched to its top choice university among its realm of possibility since any university it crossed off was not in this realm of possibility (by LEMMA).

* This lemma as preserved invariant is true b/c if we consider the case u^* (not in s^*'s realm of possibility) is matched to s^*, $\exists$ a rogue pairing thereby contradicting (6).