Stephen Woodbury   4/20/17   CMPS 102

I have read and agree to the collaboration policy. Stephen Woodbury.
Grading: Homework Heavy
Collaborators: none

Assignment 1_1 : Resident Matching

Algorithm PseudoCode:

```
1Initialize each team to be free
2while(some team has positions available & hasn't offered to every student)
3     Choose such a team, t
4     Let s = 1st student that t hasn't offered to on t's list
5     if(s doesn't have a position)
6           Let t offer a position to s
7     else if(s prefers t to current team t')
8           Let t offer a position to s
9           Let position held by s in t' be freed up
10    else
11          s rejects t
```

Assumptions:

1) Every team's preference list contains all students
2) d = number of positions available on team with largest number of positions available.

Description: The algorithm first finds a team that has positions up for hire. Then it selects that team and starts to look through its student preferences in decreasing order. The first student to not have been offered a position by said team is then analyzed. If that student doesn't have a team, then the selected team and student are matched up. If the student is on a team it prefers less to the offering team, then that student and offering team are paired up while the team the student was originally on gains a free position and loses the student. If neither of these apply to the student, then the student rejects the team and the offering team finds the next student on its preference list that it hasn't offered to analyze. Repeat until all positions of said team are filled. Then move onto another team that has positions up for hire and repeat the process. Continue the algorithm until all teams have all positions filled up or until every team has proposed to every student (which would never happen in this problem).

Proof of Correctness: Termination

Observation 1: Teams offer their positions to students in decreasing order of preference

Observation 2: Once a Student is on a team, they are never left without a team

Claim: *Algorithm terminates after **at most** nm iterations of while loop*

Pf: *Each time through the while loop, a team offers a position to a new student.*
There are only nm possible offers that could be made.

Side Note: Never will the while loop iterate nm times. Since there are more students applying than positions available, it is unavoidable that some students are left without a position. Since there are students left without a position, by observation 2, it must be that those students were never offered a position.

Proof of Correctness: Stability

Claim: *No Unstable Pairs.*
Pf (By contradiction): *Suppose Team t and Student s are an unstable pair: that is, each prefers each other to their designated assignment.*
Case 1: *t is matched with student s' and s is left with no position at the conclusion of the algorithm. t prefers s over s'*
-t must prefer s' to s as teams offer positions to students in decreasing order of preference, ie, t-s is stable
Case 2: *t is matched with student s' and s is matched with team t'. t prefers s over s' and s prefers t over t'.*
-For t to not be matched up with s, s must have rejected t when t first offered to s or later on when a team offers to s while s is matched with t.
-if s rejects t when t first makes the offer, it must be that s is with a team that it prefers over t. if s rejects t later on when another team offers to s while matched up to t, it must be that that team is more preferred than t to s. Either way, t-s is stable.
In either case, s-t is stable, which is a contradiction.

Space Complexity:

Components needed to implement the algorithm:
An array of size m for every student. For team preferences: O(nm)
An array of size n for every team. For student preferences: O(nm)
An array of size n for every student and the team they're on: O(n)
An array of arrays that store each team's students: O(nm)
An array of how many times each team has offered to a student: O(m)
Queue of Teams that have positions up for hire: O(m)
Array of size m, gives each team's number of positions initially open: O(m)
Array of size m, gives each team's number of positions left open: O(m)

Total: 3*O(nm) + 4*O(m) + O(n) : ***Space Complexity : O(nm)***

Time Complexity:

Algorithm Analysis
Line 1 : No time cost
Line 2 : O(nm) Look to Proof of Correctness for Termination
Line 3 : O(1) Selecting team operation
Line 4 : O(1) Selecting said student (uses offer count array to find)
Line 5 : O(1) Checking if s has a position
Line 6 : O(d) Setting up team and student assignment array values
Line 7 : O(1) Checking Students Team pref. Array
Line 8 : O(d) Same as Line 6
Line 9 : O(d) Finding student in team's array of positions
Line 10: no time cost
Line 11: O(1) rejecting
Line 6,9 have O(n) because to assign a team a student, have to find a spot in the team's position array, at worst, n positions to look through.
Total: O(nm) * [5*O(1) + 3*O(d)] : ***Time Complexity = O(m*n*d)***