Cody Cameron
Peter Dirks
CS 3050
Group Project: Report
Due May 4, 2015

Problem

Find a method of allocating each job applicant to at most one department in such a way that all the vacancies in all the departments shall be filled. Note that since there are more job applicants than the number of vacancies, there would some job applicants who do not get a job. Job assignments must follow the *well-formed* model.

Well-formed is defined as when the following *doesn't* happen:

1.There are two job applicants A_1 and A_2 and a department D such that
- A_1 is allocated to D, and
- A_2 is not allocated to any department, and
- D prefers A_2 to A_1

2.There are two job applicants A_1 and A_2 and two departments D_1 and D_2 such that
- A_1 is allocated to D_1, and
- A_2 is allocated to D_2 , and
- D_1 prefers A_2 to A_1 and
- A_2 prefers D_1 to D_2.

Solution Design

Our solution is a modification of the stable-marriage algorithm[1]. We could not implement the exact algorithm as the stable-marriage problem calls for a match between exactly two parties. To account for the possibility of a department having multiple job openings, we modeled our algorithm after the hospitals/residents problem[2] (also known as the college admissions problem).

Our algorithm generally followed the following pseudocode:

```
for each applicant a
     while a is not assigned
          if a's top preference has an open spot
               assign a to the job
          else
               if a's top preference has a different top pick
                    let dept's lowest pick be l
                    swap l with a
                    for all depts d
                         check all the d's assignments
```

[1] http://en.wikipedia.org/wiki/Stable_marriage_problem
[2] http://www.nrmp.org/wp-content/uploads/2014/05/Run-A-Match.pdf

```
if d prefers l over any filled pick
      swap l with the lowest pick
else
      break while
```

Solution Analysis

The worst-case scenario of this algorithm would occur if every assignment (after the initial propagation of assignments) were to require a search through a department's current picks and eventually a swap of the lowest hire with a given applicant. Since every applicant could be second-guessed by another applicant, we find our worst-case complexity to be O(n^2).

Running the Program

1) In eclipse, import the project "3050-group-fun" as an existing project in the workspace
2) run the program
3) The user will then be prompted to enter a file as input. The program is directed to read from the "test" folder. For example, to run test4.txt, one would enter "`test4.txt`" at the prompt.
4) At the end of execution, the program should print the read applicants, the preferences of the applicants, and the new hires for each department.

Contributions

Cody Cameron: Created match algorithm skeleton, tweaked logic of algorithm to give proper output.

Peter Dirks: Built initial file I/O, contributed refactoring of match algorithm implementation.

A breakdown of user contributions and commit history may be found at:
https://github.com/PDirks/cs3050-fun-group-project