Christopher Moroney

Due 29 January, 2020

1) I thought the best data structures to use in this problem was a Map and linkedLists. I thought that the linked list could help refer from one man to another, and helped pass back and forth whenever previous matched men needed to be rematched, and even the same with women too. I also thought a map was good as a returned structure because it is convenient and easy to just add the combinations of a man and woman as a pair together into a map. The total amount of generated pairs can all be displayed and referenced by the map as well. I believe that overall, considering the implementation using two linkedlists to refer back and forth between the men and women, I believe that the time complexity of this implementation is O(n^2). This is because in the best case scenario, each of the men (lets say n men) have to be matched up with another woman, and we have to analyze and scan through each man, as well as each woman to compare if the man has been taken. There are also n women, so altogether I believe the final time complexity is O(n\*n) or O(n^2).

Add all men to a linked list (with stack of preference per man)

Add all women to another linked list (with stack of preference per woman)

HashMap GaleShipley(linkedList men, linkedList women)

{

HashMap returnMap = new HashMap <man, woman>

Int mostFavored = 100; // initialize

mensPreferences = men[0] preferences into queue

While (mensPreferences.next() not visited) {

For (int i = 0; i < n; i++){ // n is number of men/ women

If mensPreference.value(i) < mostFavored

{

mostFavored = mensPreference.value(i)

}

//If woman found at most favored is not taken

If (women[mostFavored] == notTaken)

{

returnMap.add(men[i], women[mostFavored];

women[mostFavored == taken

break

}

else if (women[mostFavored] != notTaken)

{

If man[current] < man[previous] // if new guy is favored by woman

{

returnMap.add(men[i], women[mostFavored]

Previous man == notTaken;

}

Else

{

Women == notTaken

}

}

Return returnMap;

}

2)

I will be responding to question part b. The Gale-Shipley method is possible with switching the preferences of a woman to earn a higher preference. The situation is possible if two of the men’s and two of the women's preferences are the same, which in this case is m and m``, as well as w and w`. Also notice how the least favorite for every gender is the same. W` is least favorite for all men, and m`` is least favorite for all women. To start, we see that m will automatically pair with w``. M` now checks w, and will be paired together. M`` will now check w``, who prefers m over m``. M`` now checks w, but w prefers m` over m``. Thus, m`` will pair with w`. The current pairs we have are (m, w``), (m`, w), and (m``, w`). If we now switch w``’s preferences to m` → m`` → m, we will see where the change can come in. Starting over, we see m will first pair with w``. M` will now pair with w. M`` will check w``, and see that w`` “prefers’ m`` over m. Thus, m`` is now paired to w``. M will now check w, and see that w prefers m over m`. Thus, m is now paired to w. M` now checks w``, and will see that w`` “prefers” m` over m``. Thus, m` is now paired to w``. M`` will now check w, but w prefers m over m``. M`` will check w`, who is the last untaken, so m`` is paired now to w`. Thus, the pairs now are (m, w), (m`, w``), and (m``, w`). The switch caused w`` to originally have her middle choice, and by falsely switching her preferences, she ended up getting her top preference in m`. Crazy! Almost seems like deception in real life when a particular person will “fool” other people in order to get to another.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| m | m` | m`` | w | w` | w`` | w``(switch) |
| w`` | w | w`` | m | m | m` | m` |
| w | w`` | w | m` | m` | m | m`` |
| w` | w` | w` and w` | m`` | m``and m`` | m`` | m |

First Time (no false switch)

Second Time (false switch)