Jayden Younger

Ming Li

CS 2123 Data Structures

08 September 2024

CS 2123 Programming Project 1

In my Gale-Shapley algorithm implementation, I aim to match couples based on their preference for specific game genres. For example, if a woman has a preference for RPG games, and there are two potential male partners, one who likes Character Action and the other who likes RPG, the woman will prioritize the man who shares her preference for RPG games.

My adjustments

Preferences with Genre Matching

Each participant has a list of potential partners and preferred genres. The algorithm checks similarities between participants when it comes to their genre. If a woman prefers a Roguelike and is considering 2 men, the algorithm should prioritize the man who also prefers a Roguelike over the other one.

Genre Weighting

The genre compatibility score is added alongside the preference ranking system. The score will increase if the genres are the same between the man and the woman.

Algorithm Workflow

When a man proposes to a woman, instead of only comparing the man's position in the woman's preference list, the algorithm will consider genre compatibility between the two.

Code Example

Initializing the matched pairs (S) using a hash map that contains the 2 named pairs, the preference list (genrePref) that contains the name and genre, and the last var we initialize is the rank.

```
HashMap<String, String> S = new HashMap<>();
HashMap<String, HashMap<String, Integer>> rank = new HashMap<>();
HashMap<String, String> genrePref = new HashMap<>();
```

Build a rank dictionary for each woman in the list.

```
HashMap<String, HashMap<String, Integer>> rank = new HashMap<>();

        for(String m : pref.get(w)) {
            HashMap<String, Integer> mrank = new HashMap<>();
            int i = 1;
            for(String m: pref.get(w)) {
                if(genrePref.get(w).equals(genrePref.get(m))){
                    mrank.put(m, i-1);
                } else {
                    mrank.put(m, i);
                }
                 i++;
            }
            rank.put(w, mrank);
        }
```

Creates a pointer to the next woman.

```
HashMap<String, Integer> prefptr = new HashMap<>();
        for(String m: men) {
            prefptr.put(m, 0);
        }
```

```
creates a list of unmatched men.
ArrayList<String> freemen = new ArrayList<>();
        for (String m: men) {
                freemen.add(m);
        }


Compare the women with the potential man and pair them up based on their preferences
while(!freemen.isEmpty()){
        String m = freemen.remove(0);
        int currentPtr = prefptr.get(m);
        String w = pref.get(m)[currentPtr];
        prefptr.put(m, currentPtr + 1);

        if(!S.containsKey(w)){
                S.put(w,m);

        } else {
                String mprime = S.get(w);
                if (rank.get(w).get(m) < rank.get(w).get(mprime)){
                        S.put(w, m);
                       freemen.add(mprime);
                } else {
                        freemen.add(m);
                }
        }
Returns matched pairs
Return S;
}
```