The Cityzens: Bishesh, Novel, Phuc

Kent Jones

CS-473-1

15 December 2020

Final Project Deliverable #2

**Group Assemble Algorithm**

**Description:**

Group assembler algorithm is a sister version of the stable marriage algorithm. The algorithm helps in solving problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element. The goal of the project is developing our own variation of this algorithm which can help students of CS and math department to form groups and pair according to their preference.

**Challenges:**

While tackling this problem, our first framework we wanted to build was the structure where we did not have two different group of selection but rather have all the member present in the set to be able to pick each other. But this plot had a flaw of redundancy and create a loop of infinite cases and made the algorithm not effective. So, we decided to go with the variation of the stable marriage algorithm which can accommodate for students of the university.

**How to run the problem:**

1. We need to create two different and unique set of groups with even and equal number of elements
2. One group will be the selector and the other will be the suitor
3. Both groups will have a ranking system for their preference of the other group
4. Use these ranking systems, we run the algorithm to pick and pair the elements of the set
5. The algorithm will run until the program has generated stable groups for all the elements
6. The output with the set of stable pairs are displayed

**Assumptions:**

1. The user will create the groups with the criteria if the program
2. The user understands the definition of stable groups
3. The user knows the language code of python
4. The user can comprehend the output pair of groups

**How does the code run:**

1. Initialize all n ∈ G1 and m ∈ G2 to free , where G1 is group 1 and G2 is group 2
2. while ∃ free n who still has a m to pair to
3. m = first element on n’s list to whom m has not yet paired with
4. if m is free (n, m) become a pair
5. else some pair (n', m) already exist
6. if m prefers m to n
7. n' becomes free to get a new partner from m
8. else paired

**Work Citation:**

1. <https://algorithms.tutorialhorizon.com/stable-marriage-problem-gale-shapley-algorithm-java/>
2. <https://www.cse.unsw.edu.au/~tw/prvwaamas09.pdf?fbclid=IwAR3VBRI21tyPhTIg4zpXsayUZ6QJBHcg8aeO3vxFILyOW0SIm458420vtHw>