cs301 - a0

Ece Karasu

March 2022

Problem 1 - Stable Marriage Problem

(i) definition:

SMP problem is a matching problem that aims to create N stable marriages among N women and N men according to their preferences.

In other words, it aims to create N marriages among N women and N men with the fact that there is no one who wants to change their current partner.

Input → N women, N men, and their preferences
Output → N stable marriages

(ii) example:

Assume that there are 2 women (w1, w2) and 2 men (m1, m2).
The preferences of the women are: "w1": ['m1', 'm2'], "w2": ['m2', 'm1']
The preferences of the men are: "m1": ['w1', 'w2'], "m2": ['w2', 'w1']

The matchings (w1, m2), (w2, m1) are not stable because w1 and m1, or w2 and m2 would prefer to have each other as partners.

However, the matchings (w1, m1), (w2, m2) are stable because there is no one who would like to have another partner.

Problem 2 - Gale-Shapley Algorithm

(i) pseudocode:

```
Initialize each variable as single
While there is a single man m
    Set w = the first woman in the list of m who is not yet proposed by m
    If w is single
        Update m and w to married as a couple
    Else
        If w prefers m to her current partner curr_m
            Update curr_m to single
            Update m and w to married as a couple
```

(ii) complexity:

The first while loop can be iterated up to N, the number of men, and for each iteration there can be N women at most. So, the time complexity of the Gale-Shapley algorithm will be O(N*N) $\rightarrow$ O(N^2)

Problem 3 - Implementation

```
men = ['m1', 'm2']
women = ['w1', 'w2']
men_preferences = {"m1": ['w1', 'w2'], "m2": ['w2', 'w1']}
women_preferences = {"w1": ['m1', 'm2'], "w2": ['m2', 'm1']}

single_men_list = []
single_women_list = []
proposings = {}
couples = {}

for each in men:
  single_men_list.append(each)
  proposings[each] = []

for each in women:
  single_women_list.append(each)
  couples[each] = []

while len(single_men_list) > 0:
  m = single_men_list[0]
  for w in men_preferences[m]:
    if w not in proposings[m]:
      proposings[m].append(w)
      if w in single_women_list:
        couples[w] = m
        single_men_list.remove(m)
        single_women_list.remove(w)
        break
      else:
        married_m = couples[w]
        if int(women_preferences[w].index(m)) <
        int(women_preferences[w].index(married_m)):
          single_men_list.append(married_m)
          couples[w] = m
          single_men_list.remove(m)
          break

couples
```

Output becomes:

```
{'w1': 'm1', 'w2': 'm2'}
```