CS 301 - Assignment 0

Rebah Özkoç
29207

March 12, 2022

1 First Question

1.1 Part (i) - Definition

Stable Marriage Problem is defined as given two sets of the same length named as set A and set B, in which the set members specify their preference lists of set members from the other set, the set members must be matched by a "stable" way. The matching is stable if there are no two set members (Ai, Bi) which would both rather have each other than their current matching. If we define this problem as a computational problem, we get this.

Input:
Two sets of items S1 = a1, a2, . . . , an S2 = b1, b2, . . . bn
For each item in S1 and S2:
A preference list that consists of the elements of the other set *such as preference list for a1 = b(i)1, b(k)2, . . . b(l)n*

Output:
A set of length n that consists of pairings (A, B) in which there does not exist any pair (A, B) which would both rather have each other than their current matching.

1.2 Part (ii) - Example

We can give a hypothetical undergraduate university placement system as an example. In this system, there are N many students and N many quotas for the universities in the country. In this system students join a central exam and get a score. After the exam every student lists the universities by their preferences. Also, all the universities list the students by their scores. The university placements can be made by these two preferences lists and the matchings become stable.

2 Second Question

2.1 Part (i) - Pseudo Code

```
N <-- Number of A's

// Rows from 0 to N-1 represent preference lists
//of A's and rows from N to 2*N { 1 represent preference
//lists of B's.
prefer<--[[Preference list of A's and B's.]]
bPartner <-- [] // empty list
// Stores partner of B's. This is our output array that stores passing information.
// The value of bPartner[i] indicates the partner assigned to B N+i.
do for i = 0 to N
{
    bPartner[i] <-- "free"
}
// An array to store availability of A's.
// If aFree[i] is false, then ith A is free, otherwise matched.
```

```
aFree <-- [] // empty list
do for i = 0 to N
{
    aFree[i] <-- False
}
freeCount <-- N
do while freeCount > 0
{
    a <-- 0 // Pick the first free A
    do while a < N
    {
        if aFree[a] is equal to False then
        {
            exit the inner while loop
        }
        a <-- a + 1
    }
    // One by one go to all B's according to
    // a's preferences. Here a is the picked free A
    i <-- 0
    do while i < N and aFree[a] is equal to False
    {
        b <-- prefer[a][i]

    // The B of preference is free,
    // b and a become partners (Note that the partnership maybe changed later).
    if bPartner[b - N] is equal to -1 then
    {
        bPartner[b - N] <-- a
        aFree[a] <-- True
        freeCount <-- i - 1
    }
    else
    {
        // If b is not free Find the current matching of b
        a1 <-- bPartner[b - N]

        if b Prefers A Over A1 then
        {
            //break the matching between b and a1 and match a with b.
            bPartner[b - N]<--a
            aFree[a]<--True
            aFree[a1]<--False
        }
    i <-- i + 1
    }
}
do for i = 0 to N
{
    print i, N, " ", bPartner{[}i{]}
}
```

3 Question 3

Stable Marriage Problem Colab Notebook.

References

Stable Marriage Problem, GeeksForGeeks,