Student Name: Mert Ekici
Student ID: 26772

Algorithms (CS301)
Assignment 0

Sabancı Üniversitesi

Problem 1 (Stable Marriage Problem)

(i) Define SMP as a computational problem: What is the input? What is the output?

Stable Marriage Problem is a problem of matching, the aim is to find a stable match between two equally sized sets given a preference order for each element. A matching is stable when there does not exist any pair (X,Y) which both prefer each other to their current match.

If we approach this problem as a computational one, a function that solves matching problem;

- takes two equally sized sets of elements and the preference order of each element as inputs ($2 \times N \times N$ sized 2D matrix would be enough),
- finds a stable match for each element based on their preference orders,
- returns the matching results as an output (set of matched pairs).

(ii) Give an example for SMP.

Each Ivy League University (8 universities total) is expected to accept one researcher among a group of world's most sucessful 8 scientists for a research competition. Scientists have their own preference ranking of the universities since they offer different benefits and their locations vary. On the other side, universities have also finalized preference list among these 8 candidates. Finding a stable match for this university-scientist case is an example of Stable Marriage Problem.

Problem 2 (Gale-Shapley Algorithm)

(i) Present the Gale-Shapley algorithm with a pseudocode.

The idea of the Gale-Shapley algorithm is to iterate through all free elements of one set (let's call them cats) while there is any free cats available. Every free cat goes to all its preferences (will be referred as dogs) according to the order. For every dog it goes to, it checks if the dog is free, if yes, they both become matched. If the dog is not free, then the dog chooses either says "no" to it or leaves its current match according to its preference list. So a match done once can be broken if a dog gets better option. Time complexity of Gale-Shapley Algorithm is $O(n^2)$.

Algorithm 1 Gale-Shapley Algorithm

Initially all $c \in C$ and $d \in D$ are free, $S = \emptyset$
while $\exists c \in C$ who is free and hasn't tried to match with every $d \in D$ **do**
  Let d be the highest ranking element in c's preference list, to whom c has not yet tried to match
  Now c tries to match with d
  if d is free **then**
    (c, d) become matched (add (c, d) to S)
  else
    d is engaged to c'
    if d prefers c' to c **then**
      c remains free
    else
      d prefers c to c'
      c' becomes free (remove (c', d) from S
      (c, d) are paired (add (c, d) to S
    end if
  end if
end while
Return the set S of matched pairs

(ii) Analyze the asymptotic time complexity of this algorithm.

The Gale-Shapley algorithm takes $O(n^2)$ time where n is the number of elements in each set.

Algorithm iterates over one of the n sized sets and and checks each element in the selected item's preference list based on their ranking. The main loop iterates n times while each preference list check takes $O(n)$ time. Hence, the algorithm runs in $n \times O(n)$ time which is equal to $O(n^2)$.

For clarity, let us consider # of cats = c and number of dogs = d. Note that $c = d = n$.

In the worst case, all cats will be matched with their least preferred choice, so we make d iterations for each cat. Therefore, in the worst case scenario, we make a total of $c \times d$ iterations and as $c = d = n$, time complexity becomes $O(n^2)$.

Problem 3 (Implementation)

Implement the Gale-Shapley algorithm in Python, based on your pseudocode from Problem 2, and illustrate it with your SMP example from Problem 1.

Now we will solve the Ivy League and 8-Scientists Problem with the *stableMatching* function. Then, the result will be checked based on the stable matching definition stated in the Problem 1;

*A matching is **stable** when there does not exist any pair (X, Y) which both prefer each other to their current match.*

```
def stableMatching(n, catPreferences, dogPreferences):
    # Initially, all cats and dogs are free
    freeCats = list(range(n))

    catPartner = [None] * n
    dogPartner = [None] * n
    # Each cat made 0 proposals, which means that
    # his next proposal will be to the dog number 0 in his list
    nextCatChoice = [0] * n

    # While there exists at least one free cats:
    while freeCats:
        # Pick an arbitrary free cat
        c = freeCats[0]

        cPreferences = catPreferences[c]
        d = cPreferences[nextCatChoice[c]]
        dPreferences = dogPreferences[d]

        currentCat = dogPartner[d]

        # Now "cat" tries to match with "dog".
        # Decide whether "dog" accepts, and update the following fields
        # 1. dogPartner
        # 2. catPartner
        # 3. freeCats
        # 4. nextCatChoice
        if currentCat == None:
          #No cat case
          #"dog" accepts any proposal
          dogPartner[d] = c
          catPartner[c] = d
          #"cat" nextchoice is the next dog
          #in the catPreferences list
          nextCatChoice[c] = nextCatChoice[c] + 1
          #Delete "cat" from the
          #free list
          freeCats.pop(0)
        else:
          #dog exists
          #Check the preferences of the
          #current dog and that of the proposed cat's
          currentIndex = dPreferences.index(currentCat)
          hisIndex = dPreferences.index(c)
          #Accept the proposal if
          #"cat" has higher preference in the dogPreference list
          if currentIndex > hisIndex:
             #New stable match is found for "dog"
             dogPartner[d] = c
             catPartner[c] = d
             nextCatChoice[c] = nextCatChoice[c] + 1
             freeCats.pop(0)
             freeCats.insert(0,currentCat)
          else:
             nextCatChoice[c] = nextCatChoice[c] + 1


    return catPartner
```

```
import numpy as np

ivyNames = ["Brown University" ,"Columbia University","Cornell University",
"Dartmouth College","Harvard University","Princeton University","the University of
    Pennsylvania","Yale University"]

sciNames = ["Aziz" ,"Cahit","Albert","Isaac","Robert","Marie","Leia","Xin"]

ivyPrefs = []
sciPrefs = []
for i in range(8):
    print(i," ",ivyNames[i],": ",ivyPrefs[i],)
print("\n")
for i in range(8):
    print(i," ",sciNames[i],": ",sciPrefs[i],)
print("\n")
a = stableMatching(8, ivyPrefs, sciPrefs)
b = []
for i in range(8):
    b.append(str(i)+"-"+str(a[i]))

print(b)
```

Output:

```
0   Brown University :  [1, 5, 6, 4, 2, 7, 0, 3]
1   Columbia University :  [0, 2, 3, 5, 1, 4, 6, 7]
2   Cornell University :  [5, 3, 1, 0, 2, 6, 4, 7]
3   Dartmouth College :  [3, 5, 0, 6, 2, 4, 7, 1]
4   Harvard University :  [4, 1, 3, 7, 0, 5, 6, 2]
5   Princeton University :  [6, 0, 1, 3, 7, 2, 5, 4]
6   the University of Pennsylvania :  [2, 4, 6, 7, 1, 5, 3, 0]
7   Yale University :  [1, 4, 3, 5, 0, 7, 6, 2]


0   Aziz :  [7, 4, 2, 0, 5, 6, 3, 1]
1   Cahit :  [6, 2, 7, 0, 3, 4, 5, 1]
2   Albert :  [4, 6, 7, 1, 5, 3, 0, 2]
3   Isaac :  [7, 4, 2, 0, 6, 5, 3, 1]
4   Robert :  [1, 6, 5, 4, 3, 2, 7, 0]
5   Marie :  [7, 4, 5, 3, 0, 2, 6, 1]
6   Leia :  [0, 5, 1, 2, 7, 3, 4, 6]
7   Xin :  [6, 7, 1, 5, 3, 4, 0, 2]


['0-6', '1-4', '2-1', '3-5', '4-7', '5-0', '6-2', '7-3']
```

When we check all possible pairs, we can see that there does not exist any pair which both prefer each other to their current match.