ICP Final Project

Introduction:
The stable marriage problem involves pairing an equal number of men and women based on the preference lists of each person such that no two persons of the opposite sex would rather have each other than their current engagements
Marriages are considered ‘unstable’ only if both the man and the woman prefer each other over whom they are currently engaged to; hence, in all other instances, the marriages are considered ‘stable’ (so if a woman prefers a man over her current engagement but that man is happy with his current engagement, that will not qualify as an unstable arrangement)
The preference lists of each person consist of people that the person would want to be engaged to, in decreasing order of preference, and marriages are strictly heterosexual and monogamous for the purposes of this problem
The end goal of the problem is finding out if there exists an arrangement of stable marriages, given any number of men and women with their respective preferences
The Gale-Shapley algorithm has been used to solve the stable marriage problem,
It was devised by David Gale and Lloyd Shapley, in 1962, who proved that it is always possible to find a solution to this problem such that all marriages ultimately end up being stable, given an equal number of men and women.
To implement the algorithm for n men and n women, men have been numbered from 0 to n – 1 and women have been numbered from n to 2n – 1, which makes for a total of 2n people
Assuming n = 4 for the purposes of demonstration, {0, 1, 2, 3} is the set of all men who are to be engaged to {4, 5, 6, 7} which is the set of all women
The preference list is implemented as a 2-dimensional array in which the first 4 rows correspond to the preference lists of the men whereas the last 4 rows correspond to the preference lists of the women
0, 1, 2 and 3 propose to 4, 5, 6 and 7 on the basis of their preference lists and they are either tentatively engaged or downright rejected, depending on the women’s preference lists and engagement status
Tentative engagements are not finalized and may be subject to change if the woman is later proposed to by a man she prefers to her current tentative engagement
The functions used, apart from the main method, are:
checkIfWomanPrefersMan1toMan2(preferenceList, woman, man1, man2): returns true if woman prefers man2 to man1 (her current engagement), else returns false
stableMarriages(preferenceList): calculates the stable marriage arrangement for all men and women on the basis of their preference lists

Code:
```
# Stable marriage problem using Gale-Shapley algorithm
# If woman prefers man1 (whom she is currently engaged with) to man2,
return false, else return true
def checkIfWomanPrefersMan2toMan1(preferenceList, woman, man1, man2):
    # Check if woman prefers man2 to man1, whom she is currently
engaged with
    # inv: 0 ⩽ i ⩽ n
    for i in range(n):
        # If man2 precedes man1 in woman's list of preferences, that
implies woman is unhappy with man1, so she must be paired with man2
instead, whom she prefers
        if (preferenceList[woman][i] == man2):
            return True
        # If, on the other hand, man1 precedes man2 in woman's list
of preferences, that implies woman is happy with man1, so nothing
needs to be changed
        if (preferenceList[woman][i] == man1):
            return False
    # assert: i == n
# Print stable marriage arrangement for n men and n women, which
makes a total of 2n people
# 0 to n - 1 are the men
# n to 2n - 1 are the women
def stableMarriages(preferenceList):
    # The value of partnerOfWomen[i] indicates the partner assigned
to woman n+i.
    # -1 as an element of the array indicates that the (n+i)-th woman
is unengaged
    # Initialize the array such that all women are unengaged at first
    partnerOfWomen = [-1 for i in range(n)]
    # Stores the engagement status of men.
    # If isManEngaged[i] is false, then the i-th man is unengaged,
otherwise engaged.
    isManEngaged = [False for i in range(n)]
    # All men are initially unengaged, so number of unengaged men is
n
    numberOfUnengagedMen = n
    # Loop runs as long as there are unengaged men
    # assert: numberOfUnengagedMen == n
    # inv: 0 <= numberOfUnengagedMen <= n
    while (numberOfUnengagedMen > 0):
        man = 0 # Initialize to 0 to select the first unengaged man
and pair him off, continue the cycle below
        # assert: man == 0
        # inv: 0 <= man <= n
        while (man < n):
            if (isManEngaged[man] == False):
                break
            man += 1
        # assert: man == n
        # Pair with each woman on the basis of the selected unengaged
man's preference list to check compatibility
        i = 0
        # inv: 0 ⩽ i < n
        # assert i == 0
        while i < n and isManEngaged[man] == False:
            woman = preferenceList[man][i] # Store the value of the
preferred woman for man
            # If preferred woman is unengaged, man and woman become
tentative partners, subject to change
            if (partnerOfWomen[woman - n] == -1): # If partner of
woman is unengaged
                partnerOfWomen[woman - n] = man # Make man and woman
tentative partners, based on woman's preferences but subject to
change
                isManEngaged[man] = True # Update engagement status
of man
                numberOfUnengagedMen -= 1 # Number of unengaged men
decreases by 1
            else:
                # If woman is engaged, find the present engagement of
woman
                presentEngagement = partnerOfWomen[woman - n] # Store
the value of the man whom woman is currently engaged with
                # If woman prefers man over her present engagement
presentEngagement, woman gets engaged to man instead
                if (checkIfWomanPrefersMan2toMan1(preferenceList,
woman, man, presentEngagement) == False):
                    partnerOfWomen[woman - n] = man # Partner of
woman is man now
                    isManEngaged[man] = True # Update engagement
status of man
                    isManEngaged[presentEngagement] = False #
Original engagement of woman is broken, thus that man is no longer
engaged for now
            i += 1
    # assert numberOfUnengagedMen == 0
    # Display the stable arrangement of men/women
    # inv: n ⩽ i < 2*n
    # assert i == n
    for i in range(n, 2*n):
        print(i, "is paired with", partnerOfWomen[i - n])
    # assert i == 2*n
# Main method that initializes the number of men/women, their
preference lists and displays a stable arrangement on the basis of
preferenceList
if __name__=='__main__':
    n = 4 # Equals the number of men and women each
    preferenceList = [ [7, 6, 4, 5], [4, 6, 5, 7], [6, 4, 7, 5], [6,
7, 5, 4], [1, 2, 0, 3], [3, 1, 2, 0], [3, 0, 2, 1], [2, 1, 0, 3] ]
    print("Stable marriage problem:\nMen are numbered from 0 to n - 1
and women are numbered from n to 2n - 1, where n is the number of men
and women each\nThe number of men/women selected for demonstrating
the program is 4\nTherefore, {0, 1, 2, 3} is the set of all men who
are to be engaged to {4, 5, 6, 7} which is the set of all women\nThe
preference list of each man and woman, from most preferred to least
preferred, are as follows:")
    # inv: 0 ⩽ i ⩽ 2*n
    # assert: i == 0
    for i in range(2*n):
        print(i, "-", preferenceList[i])
    # assert: i == 2*n
    print("Stable marriage arrangements:")
    stableMarriages(preferenceList)
```

Output:

```
Stable marriage problem:
Men are numbered from 0 to n - 1 and women are numbered from n to 2n
- 1, where n is the number of men and women each
The number of men/women selected for demonstrating the program is 4
Therefore, {0, 1, 2, 3} is the set of all men who are to be engaged
to {4, 5, 6, 7} which is the set of all women
The preference list of each man and woman, from most preferred to
least preferred, are as follows:
0 - [7, 6, 4, 5]
1 - [4, 6, 5, 7]
2 - [6, 4, 7, 5]
3 - [6, 7, 5, 4]
4 - [1, 2, 0, 3]
5 - [3, 1, 2, 0]
6 - [3, 0, 2, 1]
7 - [2, 1, 0, 3]
Stable marriage arrangements:
4 is paired with 1
5 is paired with 0
```

```
6 is paired with 3
7 is paired with 2
```

Proofs of correctness and efficiency:

Proving there are no unstable pairs, the algorithm guarantees stability of marriages:
Assume 1 and 5 are an unstable pair such that they prefer each other to their present engagements, which have been decided by the algorithm
There can be 2 cases:
Case I: 1 was never proposed to by 5, which implies 5 prefers his present engagement to 1 since the men are proposing to the women in decreasing order of their preferences and so, 1 and 5 are stable since instability only arises if both of them prefer each other to their present engagements
Case II: 1 was proposed to by 5, in which case 5 got denied either immediately or later on since accepting would have made the pair stable and so, 1 prefers her current engagement to 5, thus 1 and 5 are stable since instability only arises if both of them prefer each other to their present engagements
For both the cases, the pair is stable, thus there are no unstable pairs at the end of algorithm execution.

Proving that all the men and women are paired, there is no unengaged man or woman at the end of algorithm execution:
Assume that a man, for instance 7, remains unengaged at the end of algorithm execution
This means that a woman, for instance 3, also remains unengaged since there are an equal number of men and women and marriages can only be monogamous
This would mean 3 was never proposed to since finding a stable engagement means that engagement would never break off (although a woman could get engaged and switch partners based on stability)
However, 7 must have proposed to everyone because he remains unengaged at the end of algorithm execution, but this is a contradiction
Hence, all men and women are paired, there is no unengaged man or woman at the end of algorithm execution

The algorithm has time complexity of $O(n^2)$ and space complexity of $O(n)$.

References:

https://en.wikipedia.org/wiki/Stable_marriage_problem
https://en.wikipedia.org/wiki/Gale–Shapley_algorithm
https://youtube.com/watch?v=Qcv1IqHWAzg
https://youtube.com/watch?v=A7xRZQAQU8s
https://www.geeksforgeeks.org/stable-marriage-problem/
https://rosettacode.org/wiki/Stable_marriage_problem
https://homes.cs.washington.edu//~anuprao/pubs/CSE421Wi2020/01stable-matching.pdf