**Implementation of iterative improvement strategy**

**for stable marriage problem**

**Aim:**

To implement the stable marriage problem using an iterative improvement strategy.

**Problem Description:**

The stable marriage problem is an iterative improvement problem where men and women are engaged based on their preference order. If a man and woman are already engaged, the woman can be engaged based on her preference, and the process repeats.

**Algorithm:**

* Initially, all men and women are free, and the matching is empty. The following steps are performed iteratively:
  + Randomly select a man and let him propose to the first woman on his preference list who hasn't rejected him so far.
  + If the woman is free, she accepts his proposal, and they become a pair in the matching. If she is not free, she compares her current match with the proposing man. If she prefers the proposing man to her current match, she accepts the proposal and replaces her current match. This way, a new pair is formed. If she doesn't prefer the proposing man, she rejects the proposal, and the proposing man remains unmatched.
* These two steps are repeated until no man remains unmatched. Since no two men can be matched with the same woman, all the women have their matches when the algorithm terminates.

**Code:**

N = 4

def womenPrefersM1OverM(prefer, w, m, m1):

    for i in range(N):

        if (prefer[w][i] == m1):

            return True

        if (prefer[w][i] == m):

            return False

def stableMarriage(prefer):

    wPartner = [-1 for i in range(N)]

    mFree = [False for i in range(N)]

    freeCount = N

    # While there are free men

    while (freeCount > 0):

        # Pick the first free man (we could pick any)

        m = 0

        while (m < N):

            if (mFree[m] == False):

                break

            m += 1

        i = 0

        while i < N and mFree[m] == False:

            w = prefer[m][i]

            if (wPartner[w - N] == -1):

                wPartner[w - N] = m

                mFree[m] = True

                freeCount -= 1

            else:

                m1 = wPartner[w - N]

                if (womenPrefersM1OverM(prefer, w, m, m1) == False):

                    wPartner[w - N] = m

                    mFree[m] = True

                    mFree[m1] = False

            i += 1

    print("Woman\tMan")

    for i in range(N):

        print(i + N, "\t", wPartner[i])

# Driver Code

choice = [[7, 5, 6, 4],

          [5, 4, 6, 7],

          [4, 5, 6, 7],

          [4, 5, 6, 7],

          [0, 1, 2, 3],

          [0, 1, 2, 3],

          [0, 1, 2, 3],

          [0, 1, 2, 3]]

stableMarriage(choice)

**Output:**

**A picture containing text, font, screenshot

Description automatically generated**

**TIME COMPLEXITY:**

The Stable Marriage Problem is typically solved using the Gale-Shapley

algorithm, which has a time complexity of O(n^2), where n is the number of men

or women in the problem.

**TIME COMPLEXITY ANALYSIS:**

* The initialization step takes O(n) time to create and initialize the arrays.
* The main loop executes at most n iterations since each man can make a

proposal to every woman.

* Proposing to a woman and updating the engagement takes O(1) time.
* Therefore, the overall time complexity of the Gale-Shapley algorithm is

O(n^2).

A**LGORITHM ANLAYSIS:**

* **Input:** The algorithm takes as input the preferences of men and women.
* **Initialization:** It initializes the engaged array and men proposals array ,both of size n, to track the current engagements and proposals made by each man, respectively.
* **Main Loop:** The algorithm continues until all men are engaged. Inside the loop, each man proposes to the next preferred woman who has not rejected him yet.
* **Proposal:** If the woman is not engaged, she accepts the proposal, and the engagement is updated. Otherwise, if the woman is already engaged, she compares the current man with her current partner based on her
* **preferences.** If the current man is preferred, the engagement is update by replacing the current partner.
* **Output:** Finally, the algorithm prints the engagements.

**Result:**

Thus, the stable marriage problem has been solved successfully