**Stable Marriage Algorithm**

In this project, we have to slightly modify the algorithm in order to achieve our goal. Since each course may have need of more than one candidate to be assigned, such changes will be made to the original Gale-Shapley Algorithm:

* During each round of proposing, a currently unmatched candidate proposes to his/her top-choice course which he/she has not proposed to yet.
* After candidates finish proposing to courses, each course takes the new proposers, put them into the same "set" with other candidates that are already matched with this course, to form a "temporary" waitlist.
* If the waitlist's length exceeds the course capacity, the waitlist will be sorted by course's preference to waitlist's members, and only the top k ones will be kept, with k being the capacity of that course.
* The algorithm terminates when there are no unmatched candidates or all candidates have proposed to all courses

**\*Gale-Shapley Algorithm\***

INPUT: preference list for men and women

INITIALIZE matching set S to an empty set

WHILE (some woman w in W is still unmatched and hasn't proposed to every man in M)

m <- first man on w's preference list to whom w has not yet proposed

IF (m is unmatched)

ADD pair (m, w) to S

ELSE IF (m prefers w to existing pair w')

REPLACE (m, w') with (m, w)

FREE w'

ELSE

w REJECT m

RETURN: matching S

**Hungarian Algorithm**

The Hungarian Algorithm is a combinatorial optimization algorithm that solves the assignment problem in polynomial time.It was developed and published in 1955 by Harold Kuhn, who gave "Hungarian Algorithm" its name according to the previous works of two Hungarian mathematicians.

**\*Hungarian Algorithm\***

INPUT: n\*n cost matrix A

FOR EACH (row R\_A in A)

SUBTRACT min(R\_A) from R\_A

FOR EACH (column C\_A in A)

SUBTRACT min(C\_A) from C\_A

LABEL appropriate entries so that all zero entries are covered and minimum number of labels are used

IF (# labels = n)

RETURN: labels as assignment

ELSE:

SUBTRACT min(A) from unlabeled R\_A

ADD min(A) to unlabeled C\_A

REPEAT from LABEL

**Maximum Matching Algorithm**

Consider an undirected graph . A matching M is said to be maximal if M is not properly contained in any other matching. Formally, for any matching of . Intuitively, this is equivalent to saying that a matching is maximal if we cannot add any edge to the existing set. And a matching is said to be Maximum if for any other matching , . Generally, maximum matching applied to unweighted graph more but for this project, we would like to modify the algorithm with weights in order to meet our propose. With researching, we decided to implement the method introduced by *Zvi Galil,* Department of Computer Science, Columbia University, in 1986. In his study "Efficient Algorithms for Maximum Matching in Graphs", he developed this method based on Berge's Theorem, "the matching M has maximum cardinality if and only if there is no augmenting path with respect to M."

**\*Maximum Matching\***

INPUT: Graph G

M <- random selected matching

WHILE (there is a blossom and there is an augmenting path in M)

GROW the forest, labeling the vertices even/odd

IF (there is a blossom in the graph)

SHRINK the blossom to obtain a new graph G'

CONTINUE foresting

ELSE

FIND such even - even edges to obtain a maximally disjoint set of augmenting

paths(P1,...,Pk)

M <- switching edges along P's from in-to-out of M and vice-versa

EXPAND all blossoms to obtain the maximum matching in the original graph G.