1. Set all men and women to free.
2. While there is a free man m and has not proposed to every woman
   - Let w be the m's highest-ranking woman m has not proposed to yet.
   - If w is free then (m, w) become engaged
   - Else w is currently engaged to m'.
     - If w prefers m' to m then m remains free
     - else w prefers m to m', (m, w) become engaged and m' becomes free (remove (m', w) from the set of engaged pairs).
3. Return the set of engaged pairs.

Exercise

Try this algorithm on your instance.

Correctness: Questions:

1. Does this algorithm produce a perfect matching?
2. If so, is it stable?

Time complexity: Questions:

1. Does it terminate for every instance?
2. If so, how long does it take as a function of $n = |M| = |W|$?

Specific: Questions:

1. Does it always produce the same matching?
2. If so, is the matching produced special?

Idea: Find a quantity that measure the progress of the algorithm.

Theorem

The algorithm terminates after at most n^2 iterations.

Proof.

1) At each step of the algorithm some man propose to a woman he has never proposed to before.
2) There are at most n^2 pairs of man and woman and hence the algorithm stops after at most n^2 iterations. $\square$

Questions:

1. What is the running time of the algorithm? Is it $O(n^2)$ (is the number of operations bounded by a constant times n^2)?
2. This depends on:
   How do you find a free man?
   How do you find the woman he will propose to next?
   How do you decide if w prefers m to m'?

Exercise

Design data structures for this algorithm so that all above queries can be answered in constant time.
Assume that the input is given by two $n \times n$ arrays P_M and P_W containing preferences of men and women: The i-th row of P_M is the list of woman in the order of their preference by man i (starting from the most preferred woman).