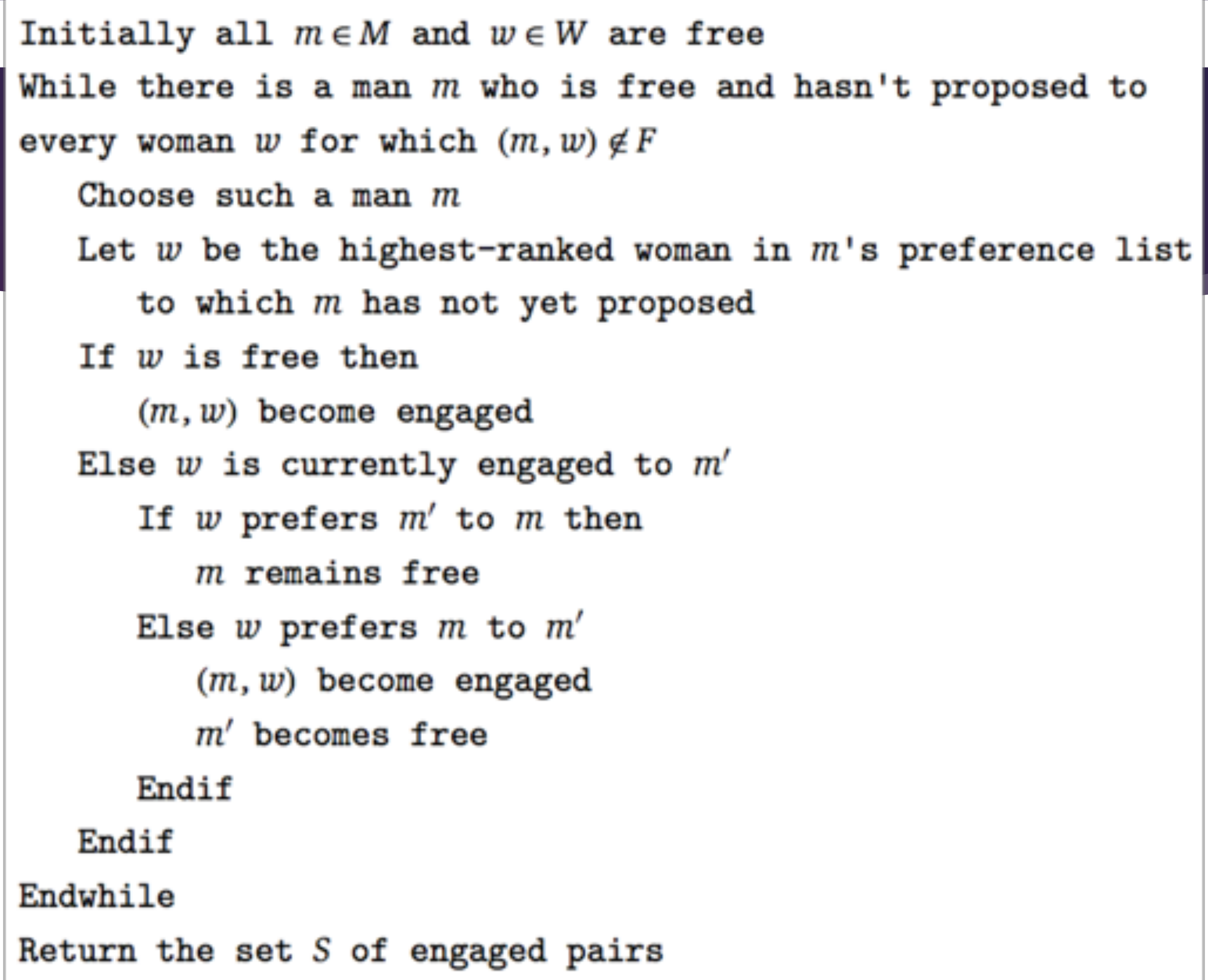
Problem Analysis Of Stable Match

YAO ZHAO



```
Initially all $m \in M$ and $w \in W$ are free
While there is a man $m$ who is free and hasn't proposed to
every woman $w$ for which $(m, w) \notin F$
    Choose such a man $m$
    Let $w$ be the highest-ranked woman in $m$'s preference list
        to which $m$ has not yet proposed
    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
            $m'$ becomes free
        Endif
    Endif
Endwhile
Return the set $S$ of engaged pairs
```

Common Problems

- What data structures are used for input and output ?
- How to find the unmatched SA efficiently?
- How to efficiently query the ranking of a SA in a student’s preference list?
- Not to test the code sufficiently

What data structures are used for input and output?

- Analysis of Input and Output Formats
- SA's name → SA's Appearance No.（Map）
- Student's name → Student's Appearance No. （Map）
- SA's Appearance No. → SA's name （Array）
- Student's Appearance No. → Student's name （Array）
- Apparently, the preference list should be a two-dimensional array. Since the Appearance No. can be easily obtained from Map, it is possible to design the preference list as int [] []
- The output is a list of student names. The i-th SA is match the i-th student. Obviously output is OK using a string array.

How to find the unmatched SA efficiently?

- Queue or Stack: O(1)
- Initial, all SA are free and add to a queue
- Each iterator pop a SA from queue, try to match, If he can steal a students from another SA，who has to go back to queue.

How to find a student of the highest rank and not be tried match before for a SA?

- Simple solution: find from head to tail every time
- But if a SA was stole a student by another SA, he should find lower rank students from the stole one.
- We can use a array to store the current preference index of SA

How to efficiently query the ranking of a SA in a student's preference list?

- Simple solution: using a loop to find the rank of a SA according the SA's Appearance No. in the student's preference list. O(n)
- More efficiently solution:

1、Maintain a reverse list of a student's preference list.

Index: SA's appearance No. → value: SA's rank

Actually, we don't need SA's rank → SA's appearance No.

2、using map to store SA's appearance No. → value: SA's rank

Data Structure List

- SA′s name → $SA's\ appearence\ No.$（Map）
- Student′s name → Student's $appearence\ No.$ （Map）
- $SA's\ appearence\ No.$ → SA′s name （Array）
- Student's $appearence\ No.$ → Student′s name （Array）
- SA's preference list（int[][]）

the first dimension: SA's appearance No.-> SA's preference list;

the second dimension: the rank of student->student's appearance No.

- Student's preference reverse list（int[][]）

the first dimension: student's appearance No.-> student's preference list;

the second dimension: SA's appearance No.-> the rank of SA

- Free SAs（Queue）
- Match status of student -> SA（Array）
- Match status of SA ->student（Array）
- When you update above variable， you should be full thought.

Pay Attention

- The problem of object copy, deep copy and shallow copy . (clone)