Sep 18

(Q1) How do we find a free woman w?

(A1) Maintain a linked list called free of free woman.

Init: Add all women to free $O(n)$

Query: Pick (say) the 1st element in free (+ delete it) $O(1)$

Update: w proposes to m

Case 1: m is free $\Rightarrow$ X

Case 2: (w', m) remain engaged $\Rightarrow$ Add w

Case 3: (w, m) get engaged $\Rightarrow$ Add w'

$O(1)$

(Q2) How would w pick her best unproposed man?

(A2) Maintain an array of size n call next

Semantics: next[w] = rank of man m that w should propose to next.

Q: What is the ID of the man w should propose to?

WomanPref [w] [next[w]] $O(1)$

Init: next [w] $\leftarrow$ 1 $\forall w \in W$ $O(n)$

Query:

Update: After w proposes next[w] $\leftarrow$ next[w]+1 $O(1)$

(Q3) How do we know who m is currently engaged to?

(A3) Array of size n called current

$$\text{current}[m] = \begin{cases} -1 & \text{o/w} \\ w & \text{if } (w, m) \text{ are engaged} \end{cases}$$

Init: current [m] $\leftarrow -1$ $\forall m \in M$ $O(n)$

Query: Read current[m] $O(1)$

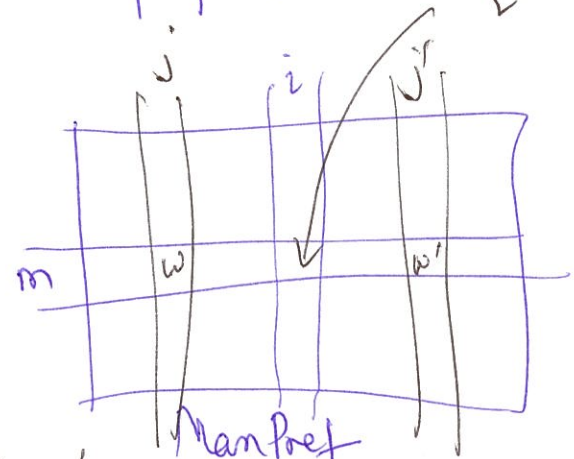
Update: If (w, m) are engaged current[m] $\leftarrow$ w $O(1)$

w proposed to m

Init (1) - (3) $O(n) + O(n) + O(n) \le O(n) \le O(n^2)$

Query/update (1)-(3) $O(1) + O(1) + O(1) \le O(1)$

(Q4) Given w, w' who does m prefer?



Do a linear scan ManPref[m]

compute j & j' s.t.

ManPref[m][j] = w

ManPref[m][j'] = w'

Check if $j < j'$ ($\Leftrightarrow$ $w > w'$ in L_m)

$O(n)$

Query $O(n)$

$\Rightarrow$ Overall $n^2 \cdot O(n) \le O(n^3)$!

A stitch in time saves nine

Solution: Design a data structure that can init in $O(n^2)$ time

but can answer Q4 in $O(1)$ time.

for j = 1 .. n

Ranking[m][ManPref ~~Ranking~~[m][j]] $\leftarrow$ j

Rank[m][w] $\leftarrow$ rank of w in m' pref list