Sep 19 Assume $M = [n]$ $W = [n]$

def $\{1, \ldots, n\}$

$[n] \Rightarrow$ { $m \rightarrow m^{th}$ man
$w \rightarrow w^{th}$ woman

i, j : ranking

Array indices start from 1

Q0) How is the input represented

A0 2D-arrays to represent prefs.

ManPref $[m][i]$ = ID of the i^{th} most preferred woman for m

WomanPref $[w][i]$ = ——— man for w

Initialization: n/a

O(1) { Query: O(1) → read a specific location in ManPref or WomanPref
Update: n/a

Q1) How do we find a free woman w?

A1 Maintain a linked list of free woman w → free

O(n) Initialization: Add all n women to free : O(n)

O(1) { Query: Pick any (say first) node in free : O(1)
(delete this node from free)

Update: [Say w proposes to m]

Case 1: (m, w) get engaged & m was free: do nothing

Case 2: m was engaged to w' &

Case 2-1: $w' > w$ in L_m : Add w to front of free → O(1)

Case 2-2: $w > w'$ in L_m : Add w' ———

Q2) How do we figure out who w's best unproposed man is?

A2: An array Next with n elements

Next[w] → rank of the man w should propose to next.

Initialization: Next[w] = 1 ∀w, O(n)

Query: WomanPref[w][Next[w]] → id of man w should propose to. : O(1)

Update: Next[w]++ : O(1)

Q3) How do we find who m is engaged to?

A3) Array Current of length n

$$Current[m] = \begin{cases} -1 & \text{if } m \text{ is free} \\ w & \text{if } (m,w) \text{ are engaged.} \end{cases}$$

Initialization: Current[m] = -1 ∀m, O(n) } O(n)

Query: Read Current[m] : O(1)

Update: (m,w) get engaged, Current[m] = w : O(1) } O(1)

(Q4): If w' > w in Lm?

Scan ManPref[m] & figure out if w' comes before w → O(n)

⇒ O(n³) overall!

"Stitch in time saves nine"

Idea: Spend O(n²) time to initialize a data structure that can answer Q4 in O(1) time.

A4: Define a 2D-array Rank

Rank[m][w] → rank of w in ManPref[m]

ManPref[m][Rank[m][w]] = w.

Initialization.

$O(n^2)$ $O(n^2)$

```
for (m = 1 .. n)
  for (j = 1 .. n)
    Rank[m][ManPref[m][j]] = j
```

$O(1)$

Query: Rank[m][w'] $\overset{?}{<}$ Rank[m][w] : $O(1)$

Update: n/a

Init (1-4) $= O(n) + O(n) + O(n) + O(n^2) = O(n^2)$

Query/Update (1-4) $= O(1) + O(1) + O(1) + O(1) = O(1)$

$\Rightarrow$ GS algo ~~to~~ can be implemented in $O(n^2)$ time

i/p size $N = \Theta(n^2)$ $\downarrow$ $O(N)$ time

linear time!