Sep 17

Notation change: Assume $M = [n] \stackrel{def}{=} \{1, .., n\}$

$W = [n]$

$m \in [n] \rightarrow$ refer to the m'th man

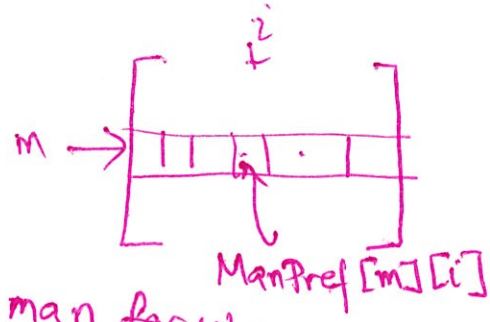
$\{m_1, .., m_n\} \longmapsto \{1, .., n\}$

→ Array indices start from 1

(Q0) How is the input represented?

2D-array ManPref, WomanPref

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



WomanPref[w][j] = ID of the j^{th} most prefered man for w.

Initialization: n/a

Query: Read value at a specific location e.g. WomanPref[w][j]

↳ O(1)

Update: n/a

Q1: How do we find a free woman w?

A1) Maintain a linked list free

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

Query: Pick say first entry in free (+ delete entry from free) [O(1)]

Update: Case 1: m was free → do nothing

Case 2: (m, w') engaged

Case 2.1: (m, w') remains engaged: Add w to free

Case 2.2: (m, w) get engaged: Add w' to free

[O(1)]

Q2: How do we figure out w's ~~next~~ best unproposed man m?

A2: Maintain an array Next of size n

Next[w] = rank of man m that she should propose to next.

Initialization: Next[w] = 1 ∀ w : O(n)

Query: Who should she propose to? WomanPref[w][Next[w]] ← O(1)

Update: Next[w]++ ← O(1)

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

A3: Array Current of length n

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

Initialize: Current[m] = -1 $\forall m$ [O(n)]

Query: Read Current[m] → O(1)

Update: If (m, w) get engaged ⇒ Current[m] = w. → O(1)

Q4) Is $w' > w$ in L_m?

A4: Scan through ManPref[m] & check if w' comes before w?

↳ O(n) ⇒ SS in $O(n^3)$ time

"stich in time saves nine"

Idea: Build another data structure in $O(n^2)$ time ⇒ Query / Update (4) is O(1)

A4' Create a 2D-array Rank

Rank[m][w] → rank of w in m's preference list ManPref[m]

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

Initialization: for m = 1..n
  for r = 1..n
    Rank[m][ManPref[m][r]] = r } $O(n^2)$

Query: Rank[m][w'] < Rank[m][w] ? ← O(1)

Update: n/a

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

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

⇒ overall ≤ Init (1-4) + n^2 · Query/Update(1-4) + $O(n^2)$

≤ $O(n^2) + n^2 \cdot O(1) + O(n)$

≤ $O(n^2) + O(n^2) + O(n) \leq O(n^2)$ ∎