Sep 16

Implementing GS

Initialization ← T_0

while (..) ← # itr = $T_1 \leq n^2$

Body ← T_2 (for each itr)

Output S ← T_3

If we could assume T_1, T_3 is $O(n^2)$, T_2 is $O(1)$

Overall runtime $\leq T_0 + T_1 \cdot T_2 + T_3$

$\leq O(n^2) + n^2 \cdot O(1) + O(n^2)$

$= O(n^2) + O(n^2) + O(n^2) = O(n^2)$

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

$W = [n]$

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

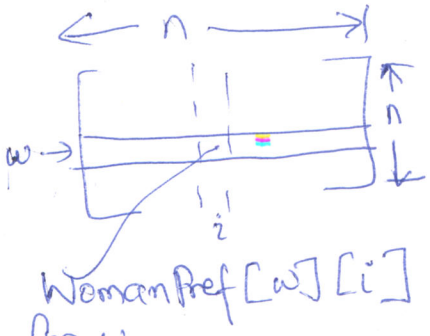
→ Array indices start at 1.

Q0) How is the input represented?

2D-Array ManPref, WomanPref

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

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



Initialization: n/a

Query: Read value at a specific location WomanPref[w][i]

↳ O(1)

Update: n/a

Q1) How do we find a free woman w?

A1) Maintain a linked list of free woman call free

Init: Add all women to free ← O(n)

Query: Pick say 1st woman in free (+ delete the entry) ↖ O(1)

Update: Case 1: m was free → do nothing

Case 2.1: (m, w') remain 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 best (proposed man m?

A2) Maintain an array Next of size n

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

Init: Next[w] = 1 $\forall w$ ← $O(n)$

Query: Who should w propose to next? WomanPref[w][Next[w]]

Update: Next[w]++ ← $O(1)$ $O(1)$

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

A3) Array Current of length n

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

Init: 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) If $w' > w$ in L_m?

Scan ManPref[m] & figure out location of w & w'.

$O(n)$!!

⇒ overall GS is $O(n^3)$.