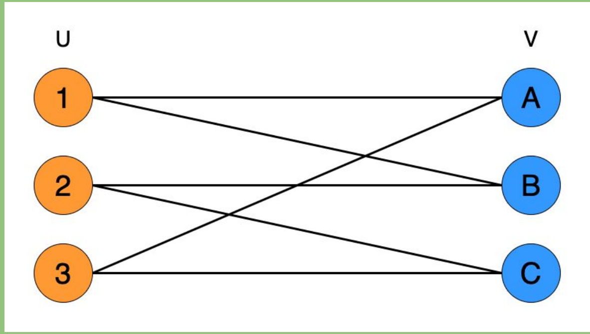
WING ALLOTMENT USING MAXIMUM BIPARTITE MATCHING



Maximising the Happiness score and allotting the wings to different wingie groups

PRESENTED BY
ROVEENA,SATYAM,GAURAV

PROBLEM STORY

- Wing allotment occurs at the end of each academic year
- During this procedure, many students feel dissatisfied since they do not receive their preferred wing

Aim:

- Implementing a system of allocating wings in such a way that the happy faces are maximised

- A wing leader is chosen from each group of wingies
- A form is floated for the wings wherein the wing leader of every wingie group needs to fill their most preferred wing names

Happiness Score 😊 - describes the happiness level of each wingie group after the allotment

Happiness Score =10 if they got their first preferred wing

Happiness Score =5 if they got their second preferred wings

Happiness Score = 0 If they didn't get any of their preferred wings

GOAL:

Allot the wings to the wingies in such a way so that the HAPPINESS SCORE is maximized

ASSUMPTIONS:

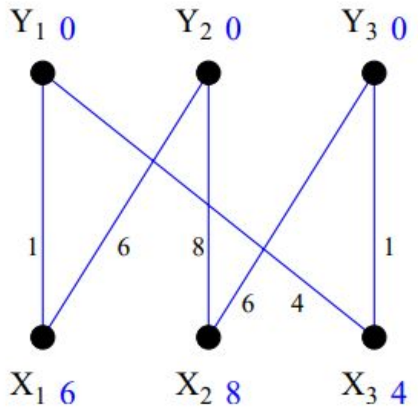
1. Every wing has the same number of rooms.
2. The number of wings is equal to the number of wingie groups.
3. No wing can be allotted to two wingie groups.
4. The wing preference form contains two choices for the most preferred wings

APPROACH

1. Construct a complete bipartite graph
   $G = (V,E)$
2. V1 as set of wingies and V2 as set of wings
3. Also we have
   $V1 \cap V2 = \varnothing$ and $V1 \cup V2 = V$
4. weight function for edges as W(e) :
   - $W(e) = 10$, $e = (v1, v2)$ where $v1 \in V1$ and $v2 \in V2$ and v1 is first preference of v2
   - $W(e) = 5$, $e = (v1, v2)$ where $v1 \in V1$ and $v2 \in V2$ and v1 is first preference of v2
   - $W(e) = 0$, otherwise
5. Finding the perfect matching inside complete bipartite graph which gives maximum happiness score.

HUNGARIAN ALGORITHM

STEP 1:

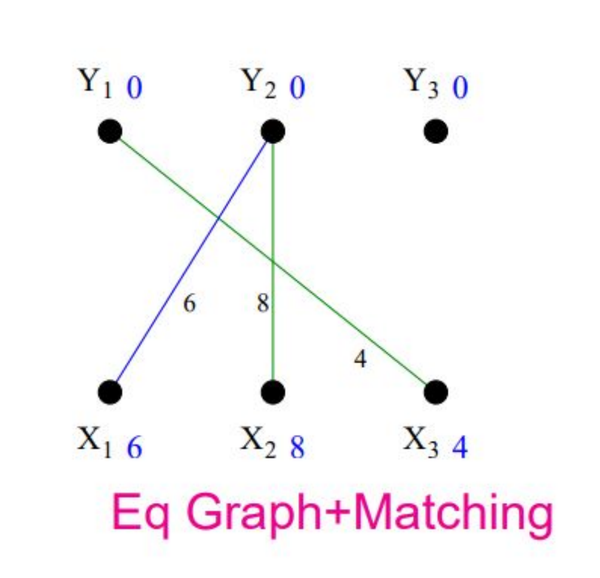


Original Graph

Finding an **initial feasible labeling** is simple. Just use:

$\forall y \in Y, \ell(y) = 0,$
$\forall x \in X, \ell(x) = \max y{\in}Y \{w(x, y)\}$

STEP 2:



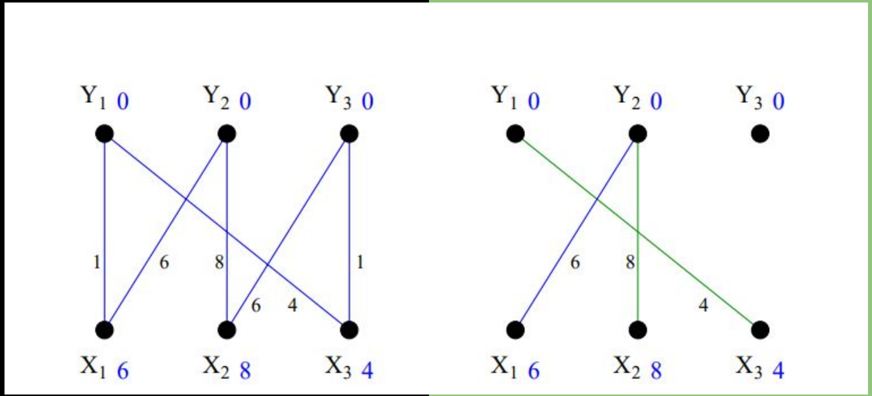
Finding an equality graph with the label matchings generated previously.

If M perfect, stop.
Otherwise pick free vertex $u \in X$.(The free vertex u tells us the conflicting vertices)
Set $S = \{u\}$, $T = \varnothing$.

Here:
$S = \{x1\}$, $T = \varnothing$.

EXAMPLE CONTINUED……

- Since $N_\ell(S)$!= T, do step 4.
- Choose y2 ∈ $N_\ell(S)$ − T.
- y2 is matched so grow tree by adding (y2, x2)
- S = {x1, x2}, T = {y2}
- At this point $N_\ell(S)$ = T, so goto 3.



CASES:

STEP 3/CASE 1:

If $N_\ell(S) = T$, update labels (forcing $N_\ell(S) \neq T$)

$$\alpha_\ell = \min_{s\in S,\ y\notin T} \{\ell(x) + \ell(y) - w(x,y)\}$$

$$\ell'(v) = \begin{cases} \ell(v) - \alpha_\ell & \text{if } v \in S \\ \ell(v) + \alpha_\ell & \text{if } v \in T \\ \ell(v) & \text{otherwise} \end{cases}$$

STEP 4/CASE 2:

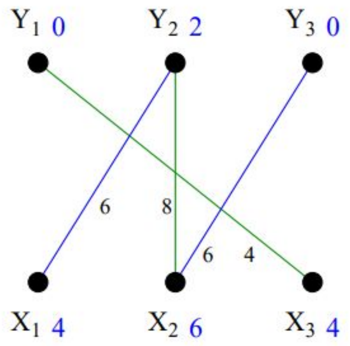
If $N_\ell(S) \neq T$, pick $y \in N_\ell(S) - T$.

- If y free, $u - y$ is augmenting path.
  Augment M and go to 2.

- If y matched, say to z, extend alternating tree:
  $S = S \cup \{z\}$, $T = T \cup \{y\}$. Go to 3.

EXAMPLE CONTINUED:

UPDATING LABEL VALUES:



new Eq Graph

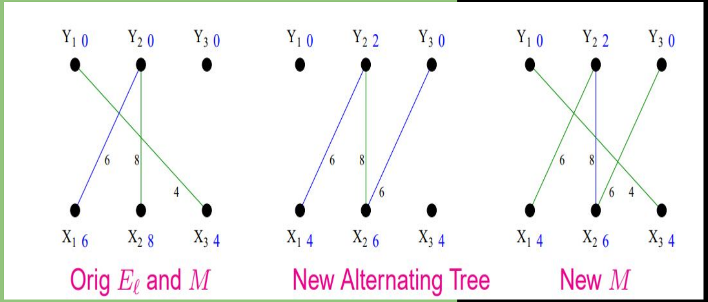
- Calculate α_ℓ

$$\begin{aligned}\alpha_\ell &= \min_{x\in S, y\notin T}\begin{cases} 6+0-1, & (x_1, y_1)\\ 6+0-0, & (x_1, y_3)\\ 8+0-0, & (x_2, y_1)\\ 8+0-6, & (x_2, y_3)\end{cases}\\ &= 2\end{aligned}$$

- Reduce labels of S by 2;
  Increase labels of T by 2.

- Now $N_\ell(S) = \{y_2, y_3\} \neq \{y_2\} = T$.

FINAL STEP:



- $S = \{x_1, x_2\}$, $N_\ell(S) = \{y_2, y_3\}$, $T = \{y_2\}$

- Choose $y_3 \in N_\ell(S) - T$ and add it to T.

- y_3 is **not** matched in M so we have just found an alternating path x_1, y_2, x_2, y_3 with two free end-points. We can therefore augment M to get a larger matching in the new equality graph. This matching is perfect, so it must be optimal.

- Note that matching (x_1, y_2), (x_2, y_3), (x_3, y_1) has cost $6 + 6 + 4 = 16$ which is exactly the sum of the labels in our final feasible labelling.

COMPLEXITY

- augment() runs for n loops.
- In each loop:
  - Slack is initialized in n iterations
  - In step 3, min delta can be calculated in n iterations and this will be done for at max n iterations, so maximum n*n iterations in this step.
  - In step 4, while adding vertices to S, each time we update slack which takes n iterations, so at max n*n iterations in this step.
- So overall, n*n*n iterations in worst case. Hence the complexity is O(n^3).

THANK YOU!!