Bipartite Maximum Matching :

Motivating Ex:

J jobs C Candidates.

The idea is we have given list of jobs that candidate can do.

L_i : List of jobs candidate i can do.

Also we have been given the constraint:

Constraint: Each candidate must be given at most 1 job, and each job must be assigned to at most 1 candidate

Goal: Assign candidates to jobs such that maximum no. of jobs are filled.

The Bipartite Maximum Matching Problem:

(i) To define this problem
(ii) To relate with this job & candidate problem.

In this problem:
our Input:

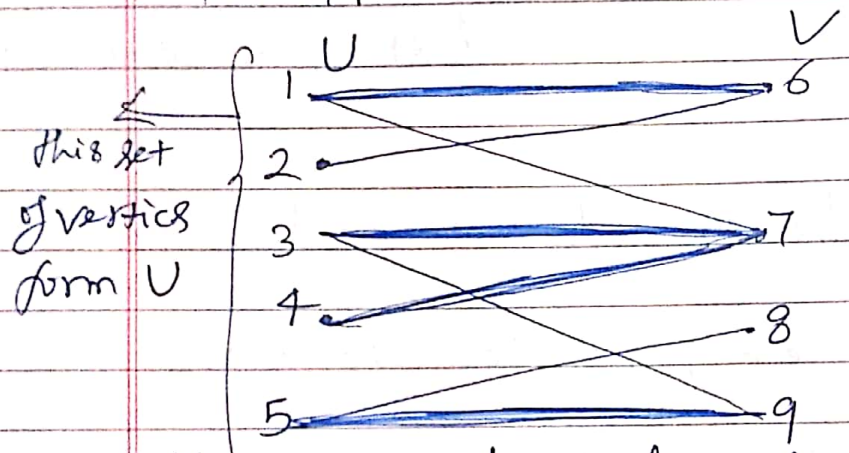
Input: Bipartite Graph $G = (U, V, E)$ → G is composed of two vertex sets (U & V) and cardinality of U is n_1 & V is n_2
where U & V are vertex sets with $|U| = n_1$, $|V| = n_2$, and E the set of edges.

Scanned by CamScanner

What is Bipartite Graph is ?

A Bipartite graph is a simply a graph in which vertex set is in two parts U & V and edges only go between U & V.

For Ex!



→ You can see, edges only go from some vertex in U to some vertex in V.
→ No edges within U, or within V.

⇒ Let me define what is matching is:

matching is the subset of edges.

matching : $M \subseteq E$ is said to be a matching if at most one edge from M is incident on any vertex (in U or in V).

→ In above ex. the set of highlighted edges would be the matching.

If I add one more highlighted 4 - 7, this should not be a matching, because we have the conflict at vertex 7. Here would be two edges of matching of so called match having is being incident and that is not

Scanned by CamScanner

Goal:

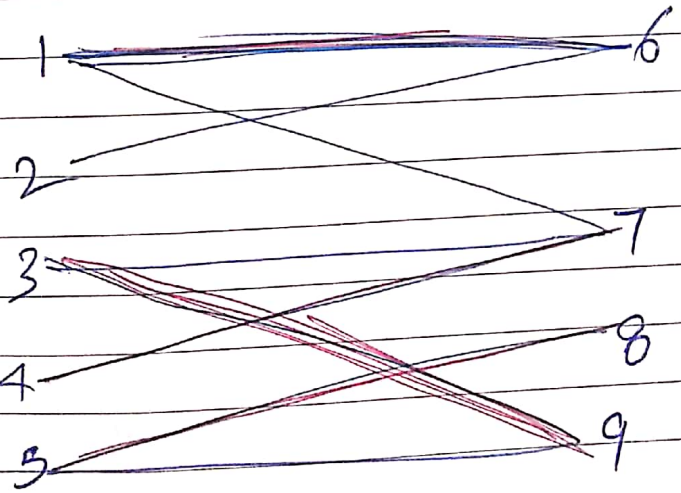
We require output:

output: Matching of maximum possible size.

> size of matching is defined as the no. of edges connect.

In previous Ex:

The highlighted matching which consists of three edges. then size is 3. This however is not the max. sized matching.



This red matching for example is the maximum size matching & infact it contaiNt 4 edges.

Maximum matchings are not unique. In above example considering the edge 2-6 in place of 1-6 would be the case of max. matching.

Scanned by CamScanner

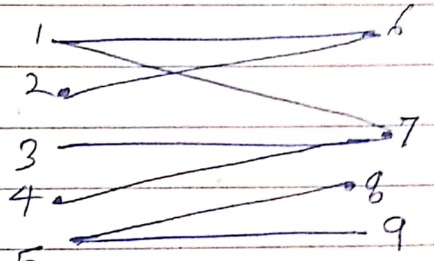
Relate this matching problem to candidate & job problems.

→ So first set of vertices called U, can be thought of the candidates:

So candidate : 1, 2, 3, 4, 5

→ And jobs are represented by the second set which we called V.

job: 6, 7, 8, 9

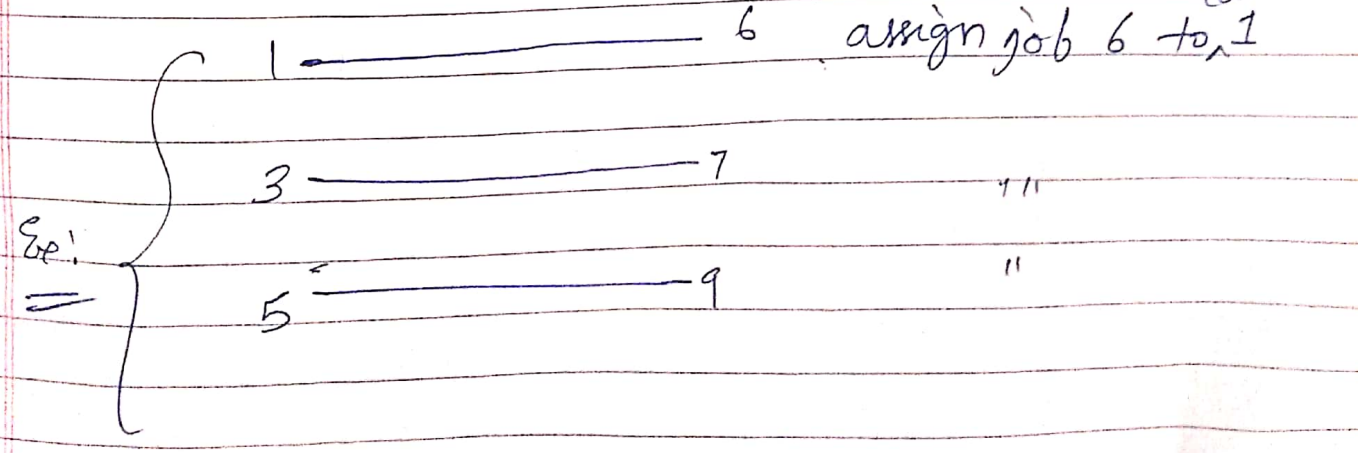


→ Edge (U, V): Candidate u can do job v.

In above Ex: Candidat 1 and do the job 6 & 7.

→ matching: job assignment

goal: maximize the no. of job assigned.



Scanned by CamScanner

classmate
Date
Page

Outline of Lecture:

(1) goining to begain with algo design idea

(2) Augmenting Pathe — This turns to be very important concept in the whole matching area.

(3) High level Algo — understanding augmenting path will lead us to reasonabely clean simple high level algo.

(4) Corrections: Berge's Theorem

its correctness simply depends on Berge's Theorem. So we will state & prove that theorem.

(5) Effecient Implementation — How to efficienty implement this algo? & we will estimate it's time.

→ Let us start with really simple minded Idea:

→ How would you find the largest size matching?

→ If we talk about the largest size, perhaps we naturally consider Greedy Idea.

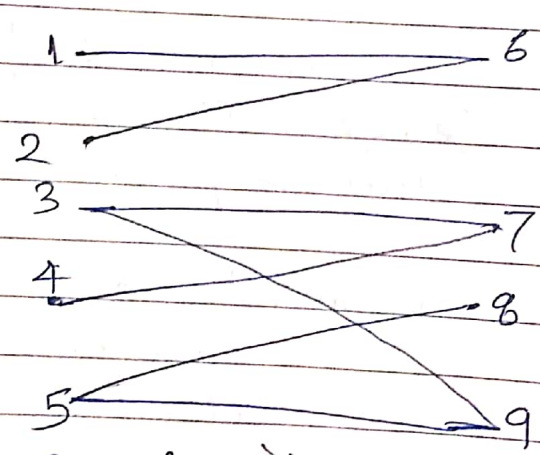
So here is the strategy:

Scanned by CamScanner

An Idea:

Greedy: keep on adding edges into m till no more edges can be added.

ofcourse, if we add the edges into m, then we make sure that there is no conflict or there are no two edges which ever are incident on the same vertex which are placed in our set m.



In above case if we try this idea then what would be happend ?

- well Start with edge 1-6 and added no problem
- then we may try to add edge 2-6, but it would produce the conflict at 6.
- So we take next edge 3-7, then we try to add 3-9, but introduce the conflict at 3.
- and 4-7 can not be added as it will product the conflict at 7.

Scanned by CamScanner

- Try for next edge addition (5-9), it is OK no conflicts.
- After this you can see that no more additions are possible.

Observation: ⇒ Have we reached best possible matching? No.

Because in Red line there are 4 matching instead of 3.

Need: So we clearly need better than simple Greedy idea.

→ Here we call the Kho Kho Idea:

Kho-Kho Idea:

There is a chaser who keeps running but who occasionally goes and knocks on the backs of the person who is sitting down, then the person who is originally sitting down starts running and start chasing the members of the opposite team until he or she again goes behind and knocks ~~behind~~ on the back of another person who is sitting and who takes over.

→ This is the idea we are going to explore:

Scanned by CamScanner

So I need a definition for that:

Kho-Kho Idea:

⇒ Free vertex for M: not an endpoint of any edge in M.

Let us take an example: First to Understand Idea

⇒ Match a free vertex; Distribute matching, Remove conflict, Repeat.

Idea is something like this:

Suppose we have a free edge, that indicate may be we can agument or we can increase the size of our matching, may be by throughing an edge by considering an edge outof that vertex and may be trying to include that edge into the matching that we have found so far.
This may succeed if it succeed ~~greats~~ great, then we have a bigger matching. If it doesnot succeed, then there are conflict with another edge ~~that~~ which is already present.

→ this is where kho-kho idea come:

Scanned by CamScanner

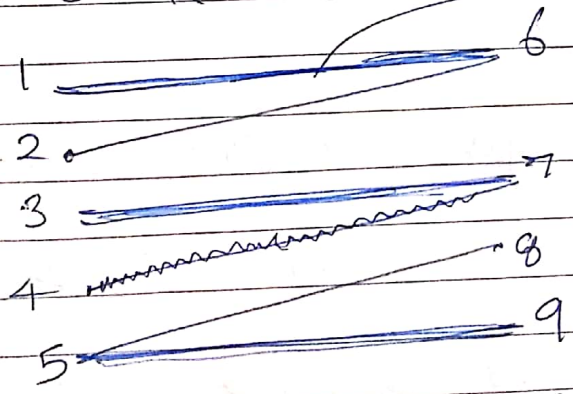
old edge is present that will be knocked out, & the new edge will sit in our matching.

→ Once knocked out the old edge then what will be happened?

Well the other end point of that will become free no we will try to match that. And we try to do this until we find that there is no conflict and if succeed then we have increased the size of our matching a little bit.

Lets start:

First check the Free Vertices: → edges of matching



Vertices 2, 4, and 8 are free for M.

Let us start with vertex 4 and see what happened if we try to match it with somewhere (i.e. 7). So 4 can be only be matched with 7.

Therefore edge 4-7 is the candidate for inclusion in old matching.

Scanned by CamScanner

→ If we try this (i.e. 4-7) then two edges incident at 7. So there is a conflict.

→ If conflict then remove 3-7 edge. then vertex 3 becomes free.

→ Now try another new matching, connect 3 to 9. but it produces conflict at 9. How do we eliminate this conflict?

→ Remove edge (5-9). Now vertex 5 is free. So we try to match it.

→ Connect 5 to 8, and there is no conflict.

⇒ At the end:

Added 3 edges (i.e (3-9), (4-7), (5-8))

Removed 2 edges (i.e. (3-7) & (5-9))

Conclusion: Earlier we have 3 edges in the matching now we have 4 " " " matching.

⇒ How we traversed the graph in entire process?

We really traversed the path in the graph

Path traversed: 4-7-3-9-5-8

Scanned by CamScanner

→ the path we traced is so important that is given a name, it's called the augmenting path.

Augmenting path for a matching M:

Sequence P of vertices $v_1, v_2, v_3, \dots v_k$ such that

- $v_1 \in U$ and $v_k \in V$ are free in M

and intermediate vertices.

like

$(v_1, v_2), (v_3, v_4) \dots (v_{k-1}, v_k) \in E - M$ — go forward

↓

orginally in the graph but not in the matching

$(v_2, v_3), (v_4, v_5) \dots (v_{k-2}, v_{k-1}) \in M$

↳ Backward edge

So we go forward, go backward ....

& we do this several times until we end with the vertex in the set V. that is the agumenting path.

Scanned by CamScanner

Operation of taking an agumenting path P & a matching M, generating New, bigger matching $= M \oplus P$

↳ Symmetric difference

Ex: $Q \oplus R$ = set of elements in Q, or in R, but not in both

then Q & R are different from each other

⇒ In the example what is included in new matching? the symmetric difference between the path & old matching.

→ So agumenting path plays a big role when you wanted to increase the size of matching.

⟹ Edmonds' Algorithm: → called the Founder of Analysis of Algo.

M = empty matching

while there is an augmenting path P for M

$M = M \oplus P$

Output M

Scanned by CamScanner

Description of Algo:

⇒ Old problem which we had with our greedy idea as well.

imp. ⇒ What if we discover that there is no agumenting path can we stop or there is a scope of bigger matching which is not being find with this idea.

⇒ This possibility can be ruled out and this is done by a theorem attributed to Berge.

Berge's theorem ⇒

A matching M in a bipartite graph is maximum if and only if there does not exist an augumenting path for M.

→ This theorem justify or proof the correctness of Edmonds algo.

Proof this theorem:

The proof is in two part.

(1) Only - if case: obvious

Scanned by CamScanner

if- case: Assume contrary, i.e. suppose

M has no augmenting path, and ∃ matching N with $|N| > |M|$ (∃ there exists).

We have in our hand matching M → we can not find the augmenting path neither have we got the best matching. ~~we are assuming, that there is bigger matching N, somewhere exist that.~~

Consider $R = M \oplus N$

Claim → R = paths and cycles (composed of.) ⟺ degree of every vertex in $R \leq 2$

What is the diff. between M & N

How we constructed R: We constructed R by taking some edges of M & some edges of N, but note that degree of every vertex in M & N at most one.

⇒ Degree of every vertex in M, N ≤ 1

The edges of M & N are such that on every vertex at most one edge from either M or N is incident.

⇒ Even if we take the union, forget about the symmetric difference, the degree of every vertex in M & N is at most one.
⇒ The degree of the union will be at most 2.

Scanned by CamScanner

Interesting Fact about R } ⇒

→ Paths/cycle edges alternate between M, N.

↓

R consists of Path & cycle but the edges in R alternate between M & N.

Some more properties: ⇒

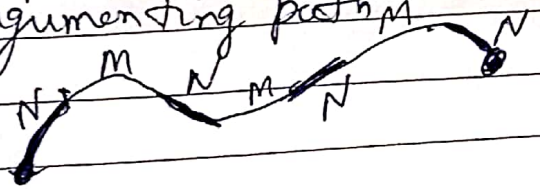
Fact → $|R| = |N| + |M| - |N \cap M|$ → R has more edges from N

Union

→ We just have to examine the implication of this } Let us look the cycle first:
Each cycle consists of equal number of edges from both M, N. If we just look at cycle this fact cannot be Explained.

→ What is happend?
∃ path P which must contain more edges from N.

First & last edges of P must be from N. First & last endpoint of P must also be free in M.
P is an agumenting path in



So we have proved that augmenting path exists and in other words this (M) can be improved.

We started on saying that M has no augmenting path, so we have gone to contradiction. This proves Berge's theorem.

Scanned by CamScanner