Lecture Notes: Maximum Matching in Bipartite Graphs

Objectives

- Be familiar with maximum matching problems and applications.
- Be able to solve maximum matching problem using Alternating Path Algorithm

Key Ideas

- Bipartite graphs
- Matching edges
- Maximum matching
- Perfect matching
- Cardinality of maximum matching
- Alternating paths
- Augmenting paths

1 Definitions

- A *bipartite graph* is a graph that can be partitioned (divided) into two sets (L and R) such that no two nodes in the same set are connected.
- A *matching edge* is an edge connecting one node from L to another node in R and a node cannot be "matched" by more than one node.
- A *matching* in a bipartite graph is a set of matching edges.
- A *maximum matching* is a matching with maximum cardinality (size).
- An *alternating path* is a path along which matching edges and non-matching edges alternate.
- An *augmenting path* is an alternating path starting with an exposed node in one side of a bipartite graph and ending at another exposed node in the other side.

2 Matching in Bipartite Graphs

Alternating Path Algorithm

1. Pick an *exposed* node (a node that has no matching edges adjacent to it) from either set of nodes (left or right). If there is none, we are done and the current matching is maximum.

2. Build an alternating tree starting from this node.

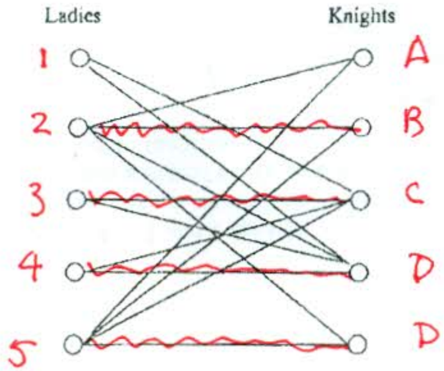
3. If another exposed node is reached, improve the matching by exchanging the matching and non-matching edges along the alternating path. This path is now called *augmenting path* between the two exposed nodes. Start over with the improved matching.

4. If no exposed node is reached, then repeat the above steps with another exposed node, but don't consider the nodes that are in the already built alternating trees (becasue we already know we cannot reach any exposed nodes from them)

A matching in a bipartite graph is called *perfect* if the matching edges cover all the nodes. Observe that we can have a perfect matching only if the number of nodes in the two parts of the node set is the same. The following classic story was adapted from *Matching Theory, by Lovász and Plummer.*

In the court of King Arthur there dwelt 150 knights and 150 ladies-in-waiting. Arthur decided to marry them all off, but the trouble was that whatever pairing he tried there were always some couples that didn't like each other. So he summoned Merlin, the Wizard and ordered him to find a pairing in which every pair was willing to marry. Merlin had supernatural powers (or just knew a little about optimization, who knows) and he saw at once that no such pairing was possible. How could he convince the king (who would surely order Merlin to be beheaded if the wizard started babbling about bipartite graphs, flows and cuts)?

Below is a simple example with 5 ladies and 5 knights. Convince yourself that there is no perfect matching in this case. How could you convince King Arthur about this? (Lines join nodes corresponding to ladies and knights who like each other.)



There are 6 individuals (A, B, C, D, E, F) and 8 jobs (1, 2, 3, 4, 5, 6, 7, 8), below is a list indicating who is qualified to do which job. Assign individuals to jobs (each person can get at most one job and each job can have at most one person assigned to it), so that the number of assignments is maximized. Draw a graph model of the problem next to the data.

A can do 1, 2, 3, 4, 6

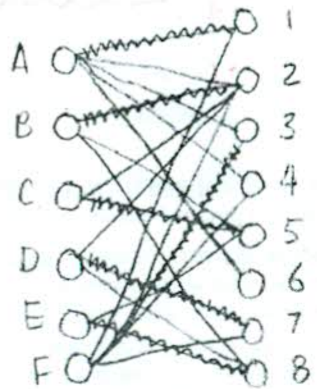
B can do 2, 5, 8

C can do 2, 5

D can do 2, 7, 8

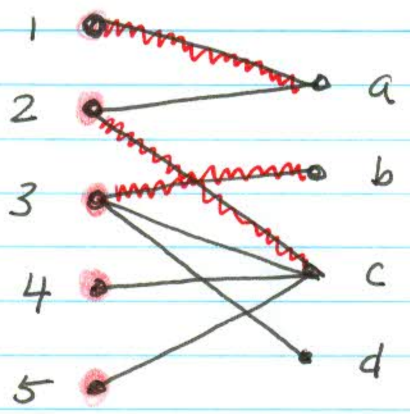
E can do 5, 8

F can do 1, 2, 3, 4, 7

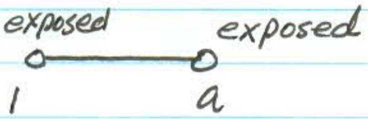


Maximum matching

p 733 example

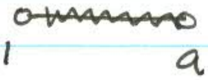


1 Start with the first exposed node '1' and build alternating path.



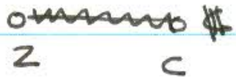
In case, it takes one edge to reach another exposed node 'a'

Convert into augmenting path



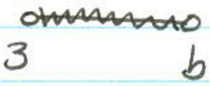
2 start the second exposed node '2'

$\Longrightarrow$

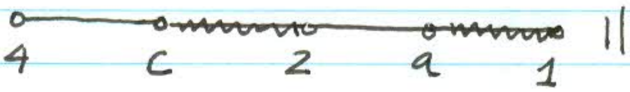


3. Start with '3'

$\Longrightarrow$

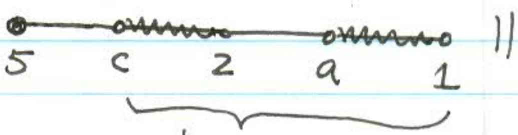


4 start w/ node '4'



no exposed ~~to~~ reached

5 start w/ node '5'.



no exposed reached

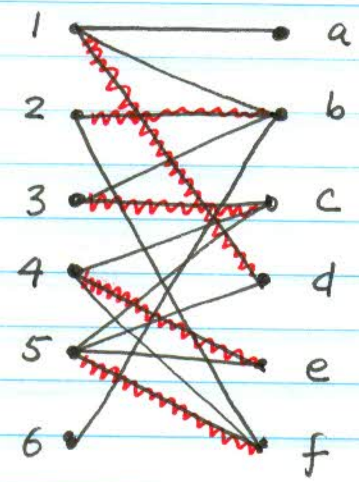
This part has been tried in step 4

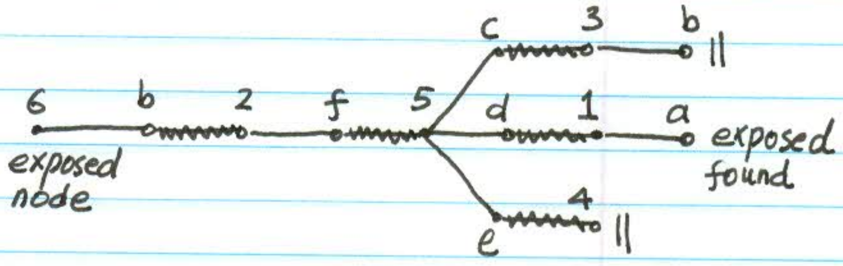
No more exposed node to start with. Done

Three matching edges are found.

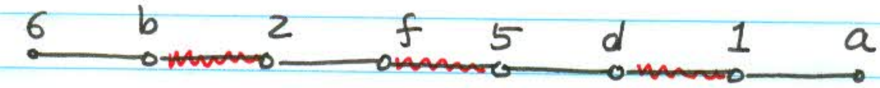
Bibartite graphs — max matching

Alternating path Algorithm





1. Suppose 5 matching edges marked in red have been found so far. There is one exposed node '6'. Start from node 6, build alternating path tree using BFS

2. 

(alternating path)

3 matching edges

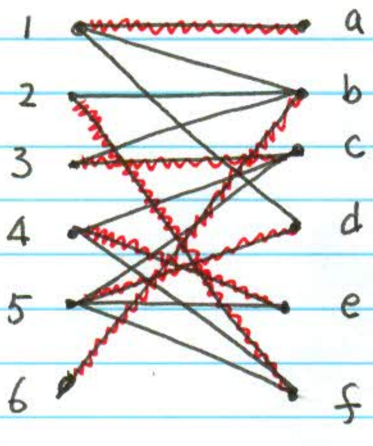
↓ flip edges

3. 

(augmenting path)

4. update the graph with newly found matching edges

now we have 4 matching edges



Now no exposed nodes are left. Done!

We have 6 matching edges marked in red.