CS 341 – Algorithms

Lecture 15 – Bipartite Matching

14 July 2021

Today’s Plan

1. Problem

2. Augmenting Path Algorithm

3. Finding an Augmenting Path

Introduction

The bipartite matching problem is an important problem both in practice and in theory.

We will learn a new algorithmic technique called the “augmenting path method” to solve the problem.

This is an important technique that underlies many algorithms for combinatorial optimization problems, including the network flow problem that you can learn in CO351.

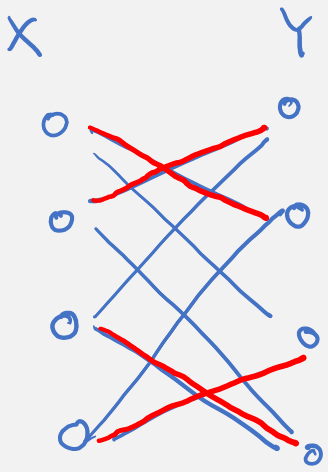
This technique can be understood more broadly as a “local search method”, in which we keep moving to a better solution using some simple operations.

All of these can be understood using the general framework of linear programming.

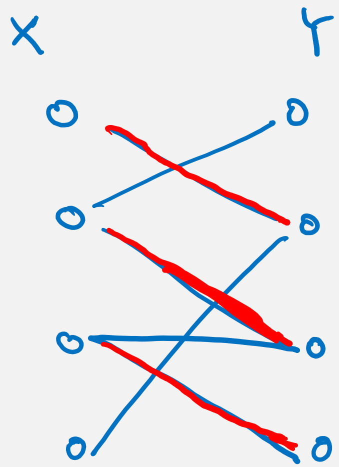
Bipartite Matching

Input: A bipartite graph $G = (X, Y; E)$.

Output: A maximum cardinality subset of edges that are vertex disjoint.



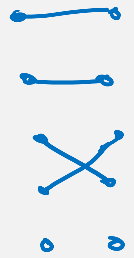
perfect matching



maximum matching is of size 3

Terminology

A subset of edges $M \subseteq E$ is called a matching if edges in M are pairwise vertex disjoint.



Given a matching M, we say a vertex v is matched if v is the endpoint of some edge $e \in M$;
otherwise we say v is unmatched or free.

We say a matching M is a perfect matching if every vertex is matched in M.

Clearly, a perfect matching is the best one can hope for in the bipartite matching problem.

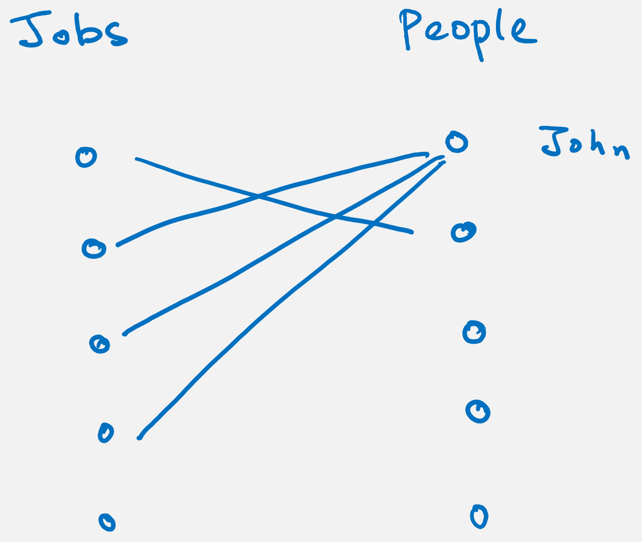
In the next lecture, we will see a nice characterization of when a graph does not have a perfect matching.

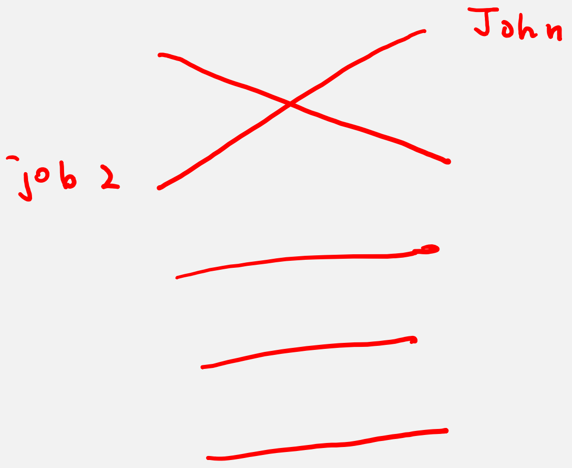
Job Assignment

A standard application of bipartite matching is the job assignment problem.

<u>Input</u>: We are given n jobs and m people. Each person is only capable of doing a subset of jobs.

<u>Output</u>: Our task is to assign all the jobs to people, without assigning more than one job to a person.



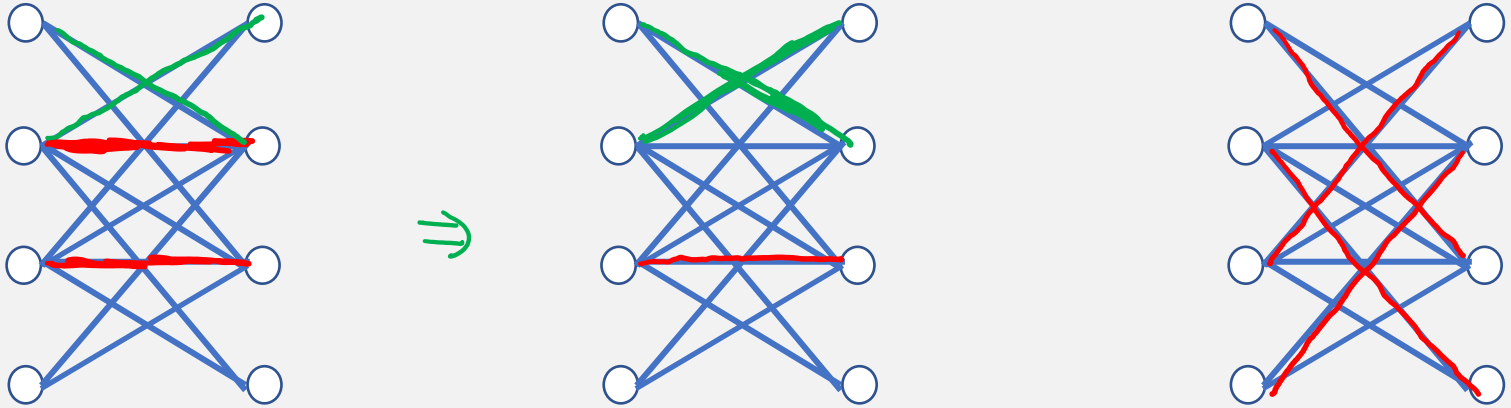


Today’s Plan

1. Problem
2. Augmenting Path Algorithm
3. Finding an Augmenting Path

Greedy Algorithm

A first natural approach is to go greedy: Always add a “free” edge to the current partial solution.



greedy solution

matching of size 3

perfect matching

local search

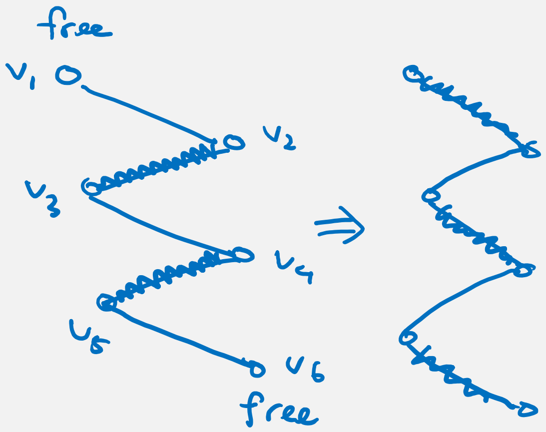
Augmenting Path

A path $v_1, v_2, \ldots, v_{2k}$ is an <u>augmenting path</u> with respect to a matching M if

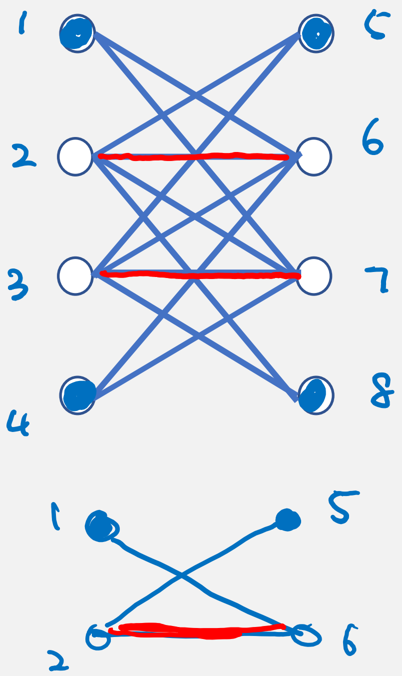
1) v_1 and v_{2k} are free vertices,

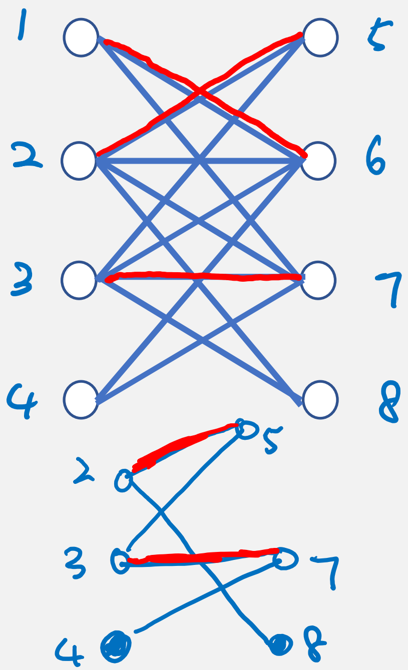
2) $v_{2i-1}v_{2i} \notin M$ for $1 \leq i \leq k$, odd edges not in the matching

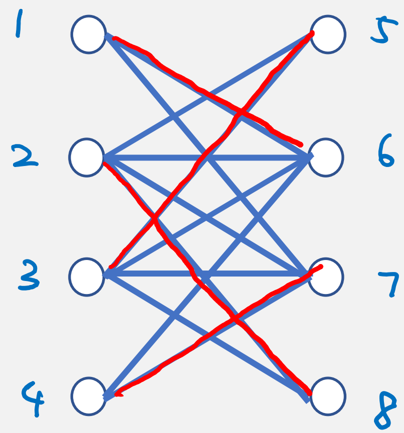
3) $v_{2i}v_{2i+1} \in M$ for $1 \leq i \leq k-1$. even edges are in the matching



free





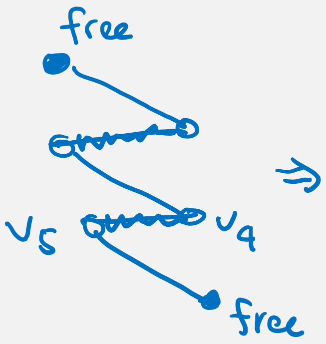


Main Observation

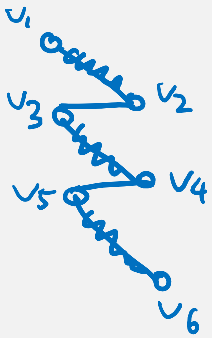
Proposition. M is a maximum matching if and only if there is no augmenting path with respect to M.

Equivalently, M is not a maximum matching if and only if there is an augmenting path with respect to M.

($\Leftarrow$) easy



$\Rightarrow$



$$M' \leftarrow M - \{v_{2i}v_{2i+1} \mid 1 \le i \le k-1\}$$
$$+ \{v_{2i-1}v_{2i} \mid 1 \le i \le k\}$$

$$|M'| = |M| + 1$$

v_1, v_{2k} were free before, now degree 1

$v_2, \dots v_{2k-1}$ still have degree 1.

So M' is a bigger matching then M.

Main Observation

Proposition. M is a maximum matching if and only if there is no augmenting path with respect to M.

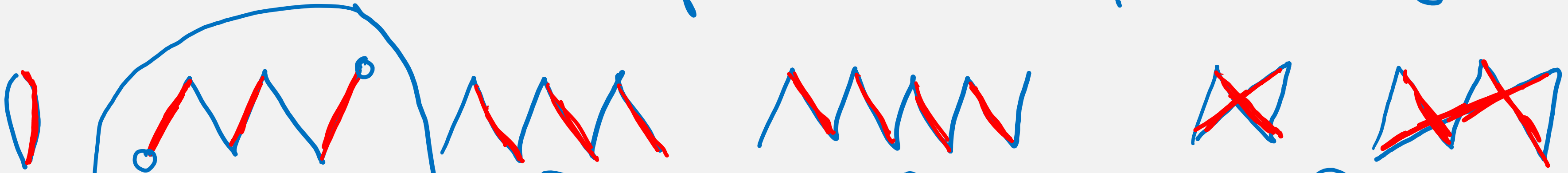
Equivalently, M is not a maximum matching if and only if there is an augmenting path with respect to M.

$\Rightarrow$) interesting direction

— M^* maximum matching, so by assumption $|M^*| > |M|$

consider $M \cup M^*$, the union has max deg ≤ 2.

$\Rightarrow$ each component is either a path or a cycle.



1 common edges

2 path with more red than blue

3 path with same red and blue

4 path with more blue than red

5 cycles

type 2 is an augmenting path

type 2 must exist because # red > # blue

$\square$

Algorithm

The proposition suggests a "local search" algorithm for finding a maximum matching.

$M = \emptyset$ // Start from an empty matching

While there is an augmenting path $P = v_1, v_2, \ldots, v_{2k}$ of M do

$$M \leftarrow M - \{v_{2i}v_{2i+1} \mid 1 \le i \le k-1\} + \{v_{2i-1}v_{2i} \mid 1 \le i \le k\}$$

Return M.

Correctness: If $\nexists$ augmenting path of M,

then M is maximum. ← by proposition.

Time Complexity

Let $T(m, n)$ be the time complexity to find an augmenting path of M if it exists,

or report that no such paths exist, in a graph with n vertices and m edges.

If $\exists$ an augmenting path. then matching size increases by 1.

matching size $\leq \frac{n}{2}$ $\Rightarrow$ $O(n)$ iterations

$\Rightarrow$ total time : $O(n \cdot T(m,n))$

will show than $T(m,n) = O(m+n)$ $\Rightarrow$ total $O(nm)$.

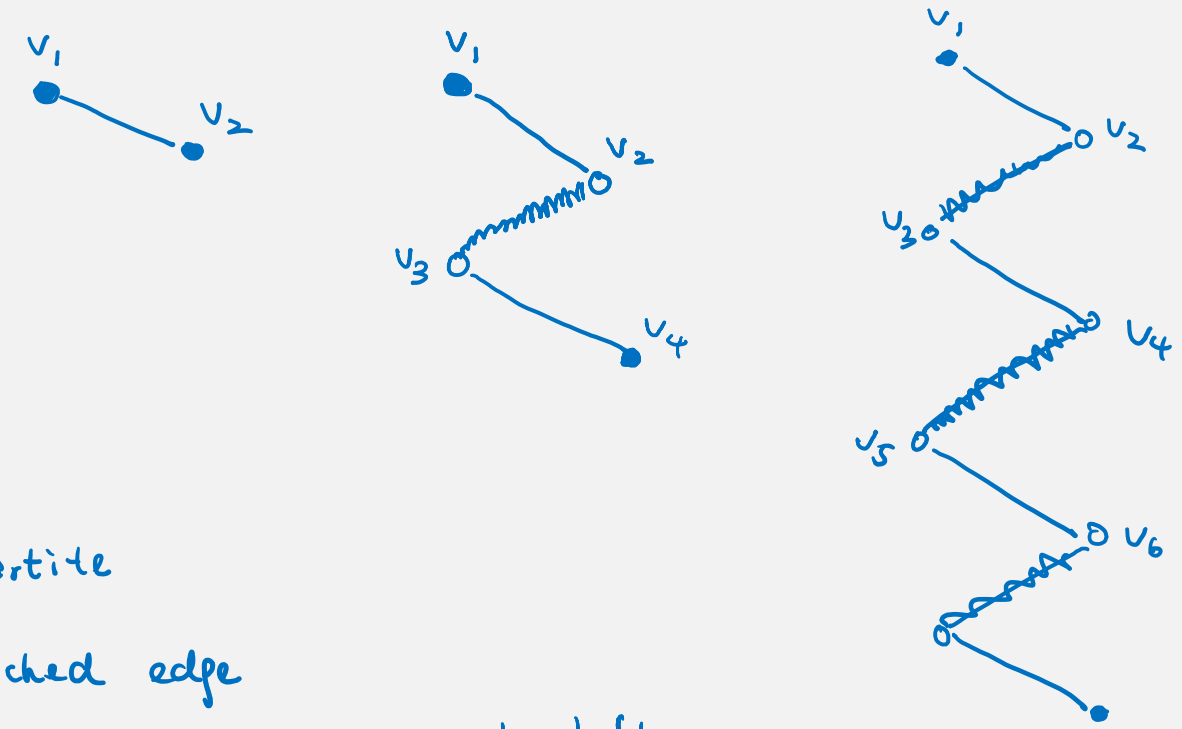
<u>Faster Algorithm</u>: There is an algorithm by Edmonds and Karp which solves the problem in $O(m\sqrt{n})$ time.

Today's Plan

1. Problem
2. Augmenting Path Algorithm
3. Finding an Augmenting Path

Idea

The bipartite graph structure allows us to design a simple algorithm to find an augmenting path.



obs: bipartite

matched edge

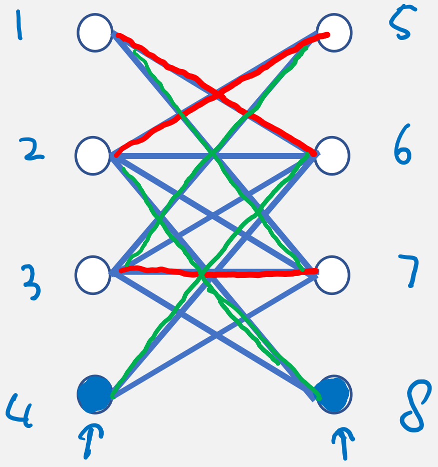
to go from right to left

idea: use direction to encode color

Directed Graph

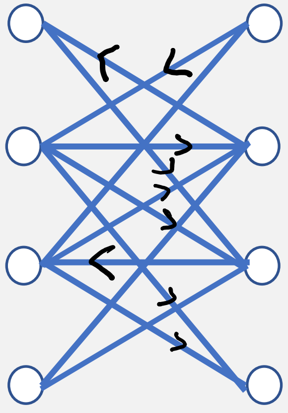
The idea is to encode the color information on the edges by directions.

Claim $\exists$ an augmenting path of M in G iff
$\exists$ a directed path from a free vertex on the left to a free vertex on the right.



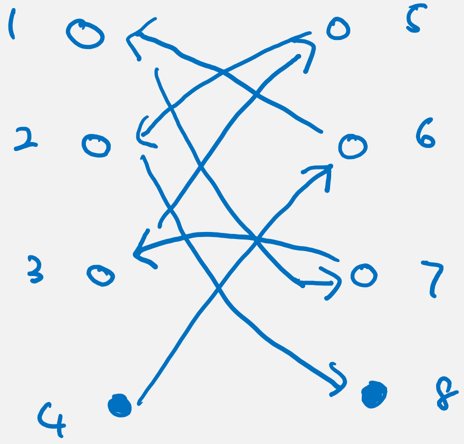
G

$\vec{G}_M$



all matched edges from right to left

all unmatched edges from left to right

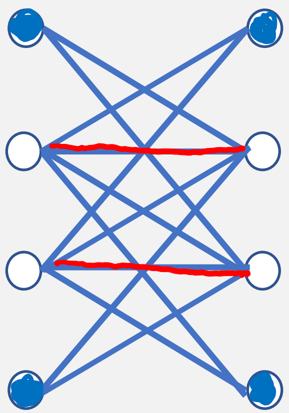


Reachability

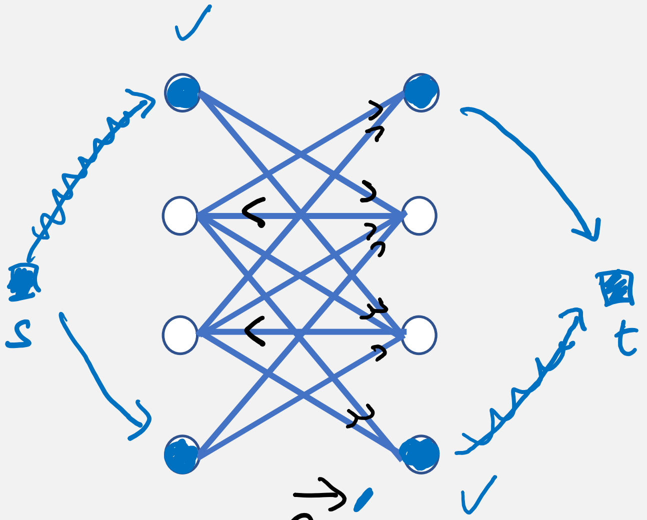
The claim allows us to reduce the problem of finding an augmenting path to a reachability problem.

naive : do a BFS/DFS on each free vertex on left

$O(n \cdot m)$



G



$\vec{G}_M'$

Claim. $\exists$ an augmenting path of M in G

$\Leftrightarrow$ $\exists$ a directed path from s to t in $\vec{G}_M'$

Algorithm

Input: a bipartite graph $G=(X,Y;E)$, and a matching $M \subseteq E$.

Output: an augmenting path P of M in G, or report that no such paths exist.

1. Construct the directed graph $\vec{G}_M'$ as described above.

2. Use BFS/DFS to determine if there exists a directed path P in $\vec{G}_M'$ from s to t.

If yes, return P (

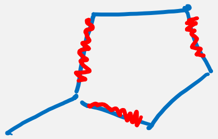
If no, return "No".

Concluding Remarks

<u>Time Complexity</u>: $O(m+n)$. augmenting path $\Rightarrow$ $O(mn)$ for maximum matching

<u>Challenge</u>: Can you solve the maximum matching problem in general (non-bipartite) graphs?

Which step of the bipartite matching algorithm breaks?



<u>History</u>: Edmonds designed a famous "blossom" algorithm to solve the maximum matching problem.

Tutte also done important work in the maximum matching problem. algebraic approach

characterization