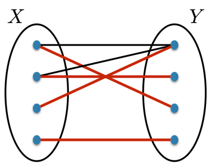
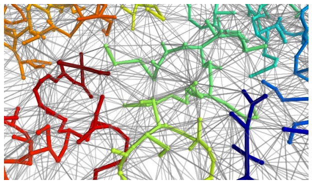
15-251
Great Ideas in
Theoretical Computer Science

Lecture 13:
Graphs III: Maximum Matchings





October 10th, 2017

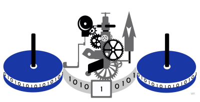
Some motivating real-world examples

matching machines and jobs

Job 1

Job 2

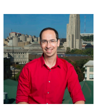
⋮ ⋮



Job n

Some motivating real-world examples

matching professors and courses



⋮

15-110

15-112

15-122

15-150

15-251

⋮

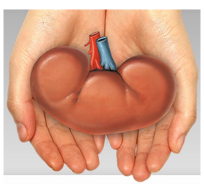
Some motivating real-world examples

matching rooms and courses

GHC 4401	15-110
DH 2210	15-112
GHC 5222	15-122
WEH 7500	15-150
DH 2315	15-251
⋮	⋮

Some motivating real-world examples

matching kidney donors and patients



How do you solve a problem like this?

1. Formulate the problem
2. **Ask**: Is there a trivial algorithm?
3. **Ask**: Is there a better algorithm?
4. Find and analyze

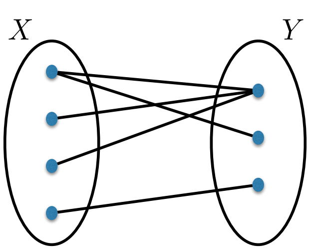
Remember the CS life lesson

First step: Formulate the problem

Purpose:

- Get rid of all the distractions, identify the crux.
- Get a clean mathematical model that is easier to reason about.
- Solutions often generalize to other settings.

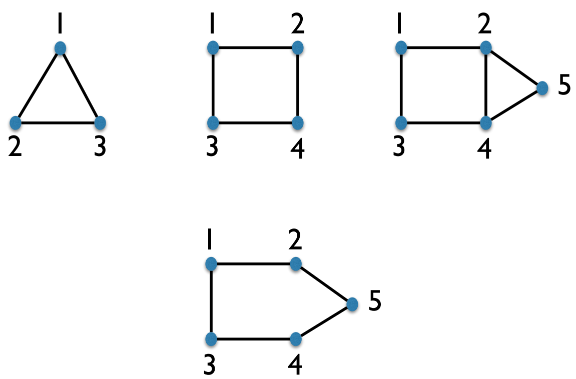
Bipartite Graphs



$G = (V, E)$ is **bipartite** if:

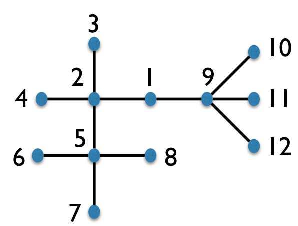
Bipartite Graphs

Given a graph $G = (V, E)$, we could ask, is it bipartite?



Poll

Is this graph bipartite?



- Yes
- No
- Beats me

Important Characterization

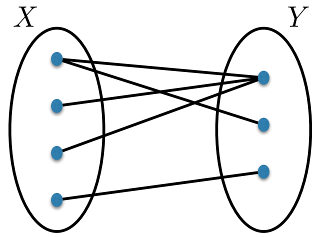
An obstruction for being bipartite:

Contains a cycle of odd length.

Is this the only type of obstruction?

Theorem:

Bipartite Graphs



Often we write the bipartition explicitly:

$$G = (X, Y, E)$$

Bipartite Graphs

Great at modeling relations between two classes of objects.

Examples:

X = machines, Y = jobs
An edge $\{x, y\}$ means x is capable of doing y.

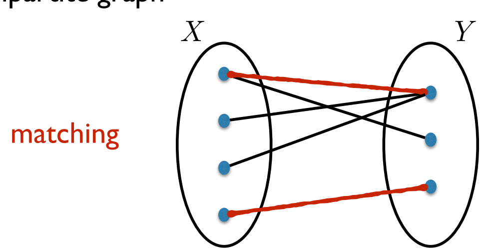
X = professors, Y = courses
An edge $\{x, y\}$ means x can teach y.

X = students, Y = internship jobs
An edge $\{x, y\}$ means x and y are interested in each other.

⋮

Matchings in bipartite graphs

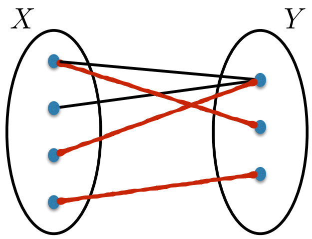
Often, we are interested in finding a **matching** in a bipartite graph



A **matching** :

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

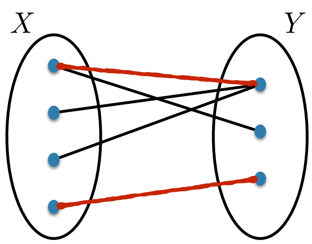


maximum matching

Maximum matching:

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

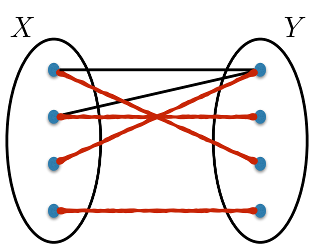


maximal matching

Maximal matching:

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

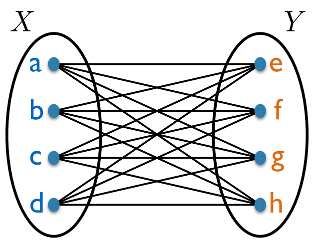


perfect matching

Perfect matching:

Poll

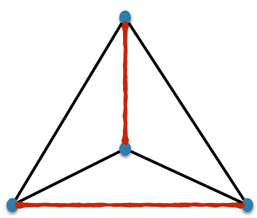
How many different perfect matchings does the graph have (in terms of n)?



$$|X| = |Y| = n$$

Important Note

We can define matchings for <u>non-bipartite</u> graphs as well.



Maximum matching problem

The problem we want to solve is:

Maximum matching problem

Input: A graph $G = (V, E)$.
Output: A maximum matching in G.

Bipartite maximum matching problem

Actually, we want to solve the following restriction:

Bipartite maximum matching problem

Input: A *bipartite* graph $G = (X, Y, E)$.
Output: A maximum matching in G.

How do you solve a problem like this?

1. Formulate the problem
2. **Ask**: Is there a trivial algorithm?
3. **Ask**: Is there a better algorithm?
4. Find and analyze

Bipartite maximum matching problem

Bipartite maximum matching problem

Input: A *bipartite* graph $G = (X, Y, E)$.
Output: A maximum matching in G.

Is there a (trivial) algorithm to solve this problem?

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm?

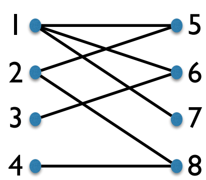
3. **Ask**: Is there a better algorithm?

4. Find and analyze

Bipartite maximum matching problem

A good first attempt:

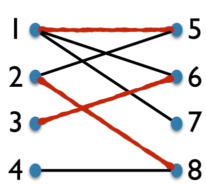
What if we picked edges "greedily"?



Bipartite maximum matching problem

A good first attempt:

What if we picked edges "greedily"?



maximal matching

but not maximum

Is there a way to get out of this *local optimum*?

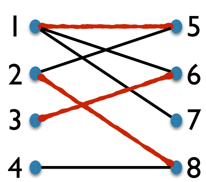
Important Definition: Augmenting paths

Let **M** be some matching.

An ***alternating path*** with respect to **M** is a path in **G** such that:

An ***augmenting path*** with respect to **M** is an alternating path such that:

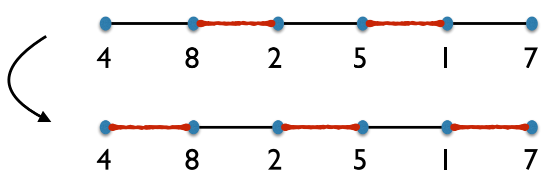
Important Definition: Augmenting paths



matching = red edges

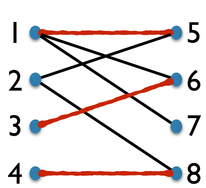
Augmenting path:

4-8-2-5-1-7



augmenting path $\Longrightarrow$ can obtain a bigger matching.

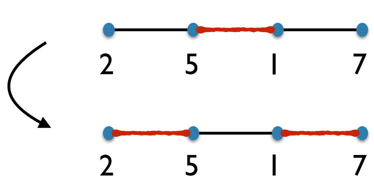
Important Definition: Augmenting paths



matching = red edges

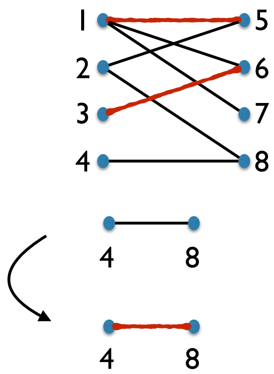
Augmenting path:

2-5-1-7



augmenting path $\Longrightarrow$ can obtain a bigger matching.

Important Definition: Augmenting paths



matching = red edges

Augmenting path:

4-8

augmenting path $\Longrightarrow$ can obtain a bigger matching.

Augmenting paths and maximum matchings

augmenting path $\Longrightarrow$ can obtain a bigger matching.

In fact, it turns out:

no augmenting path $\Longrightarrow$ maximum matching.

Theorem:

Augmenting paths and maximum matchings

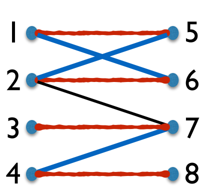
Proof:

If there is an augmenting path with respect to **M**, we saw that **M** is not maximum.

Want to show:

If **M** not maximum, there is an augmenting path w.r.t. **M**.

Let **M*** be a maximum matching. |**M***| > |**M**|.

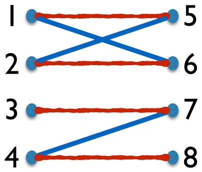


Let **S** be the set of edges contained in **M*** or **M** but not both.

S = (**M*** $\cup$ **M**) - (**M** $\cap$ **M***)

Augmenting paths and maximum matchings

Proof (continued):

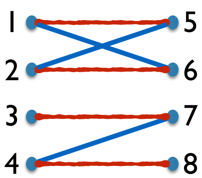


$\mathbf{S} = (\mathbf{M^*} \cup \mathbf{M}) - (\mathbf{M} \cap \mathbf{M^*})$

(will find an augmenting path in **S**)

Augmenting paths and maximum matchings

Proof (continued):



$\mathbf{S} = (\mathbf{M^*} \cup \mathbf{M}) - (\mathbf{M} \cap \mathbf{M^*})$

(will find an augmenting path in **S**)

Augmenting paths and maximum matchings

Theorem:

A matching **M** is maximum **if and only if** there is **no** augmenting path with respect to **M**.

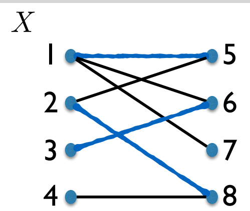
Summary of proof:

Algorithm to find maximum matching

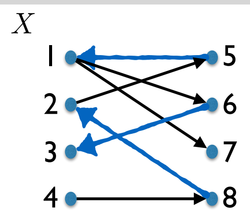
Theorem:
A matching **M** is maximum **if and only if** there is **no** augmenting path with respect to **M**.

Algorithm to find max matching:

Finding augmenting paths in bipartite graphs



Finding augmenting paths in bipartite graphs



Algorithm:

Running time:

How do you solve a problem like this?

1. Formulate the problem
2. **Ask**: Is there a trivial algorithm?
3. **Ask**: Is there a better algorithm?
4. Find and analyze