

# Matchings II: Finding the maximum

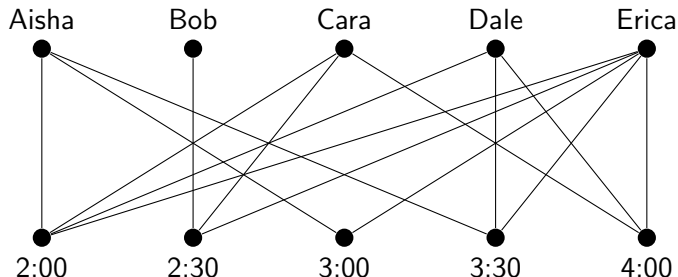
## COMS20010 (Algorithms II)

John Lapinskas, University of Bristol

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

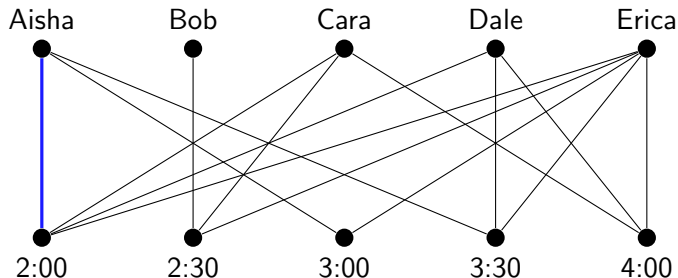
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

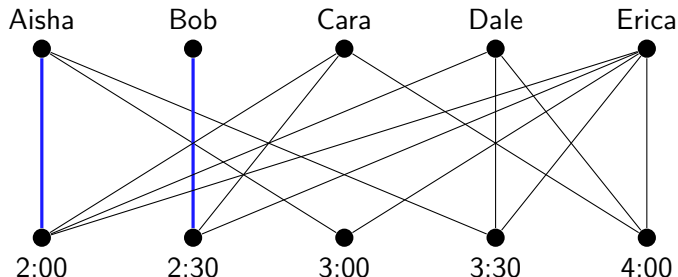
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

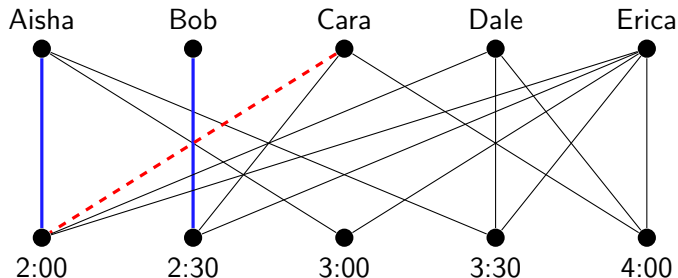
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

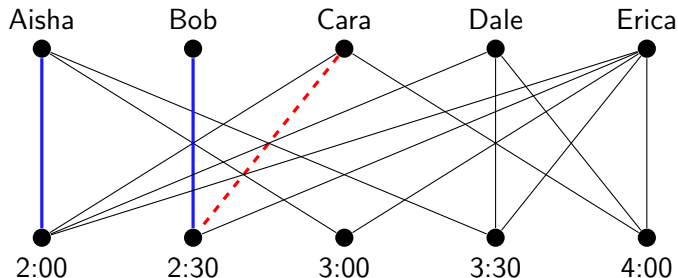
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

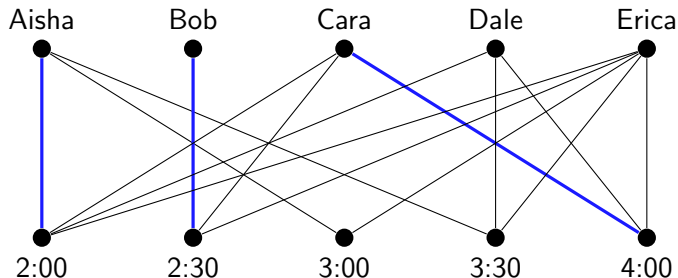
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

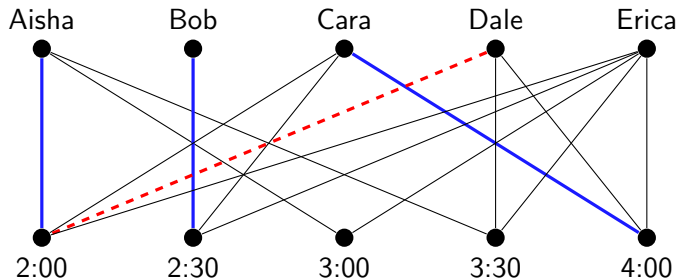
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

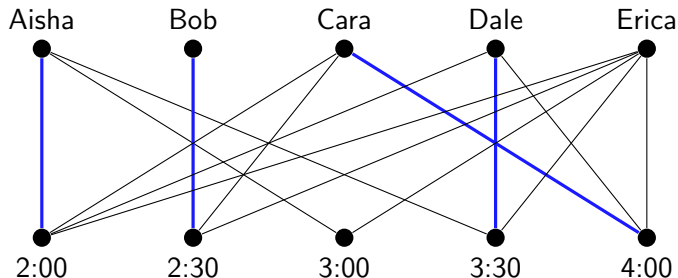




# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

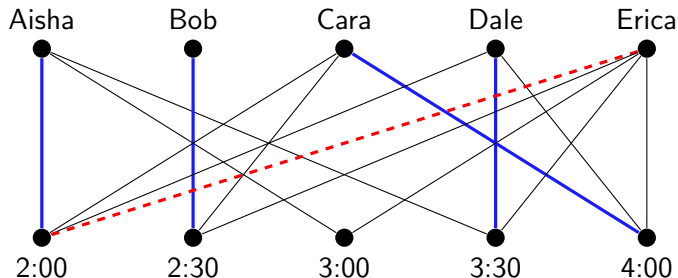
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

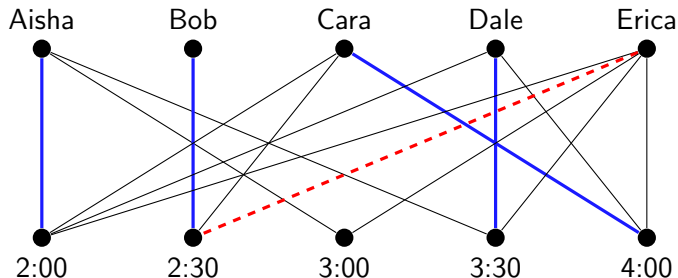
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

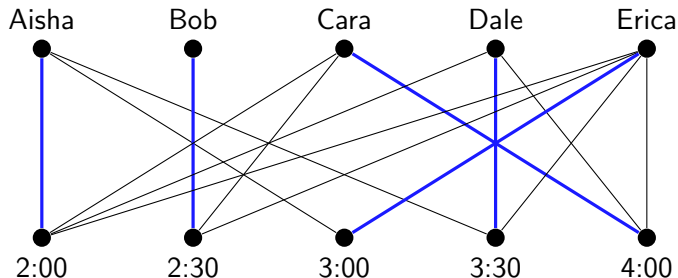
Can we form  $M$  by greedily adding edges? E.g.:



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

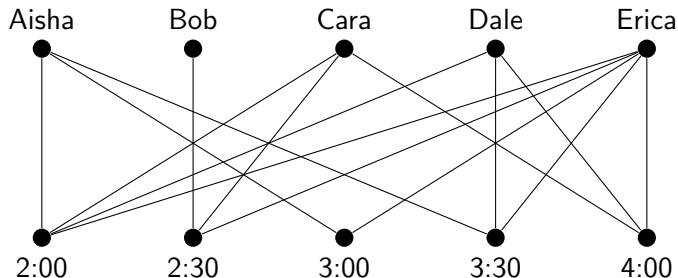


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

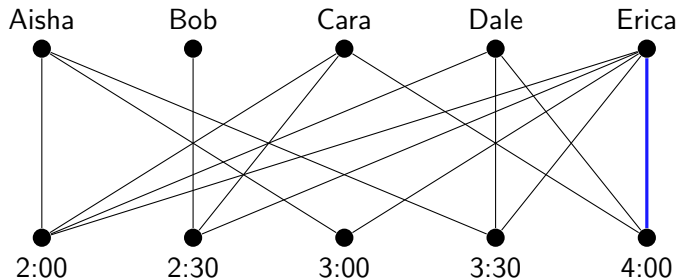


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

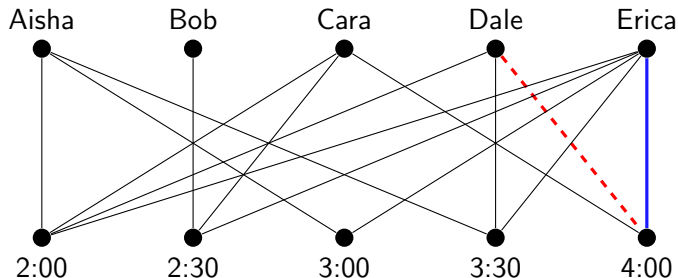


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

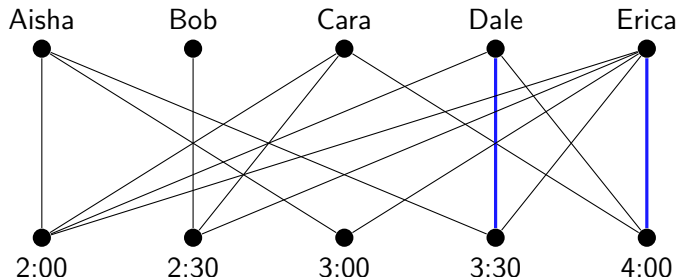


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:



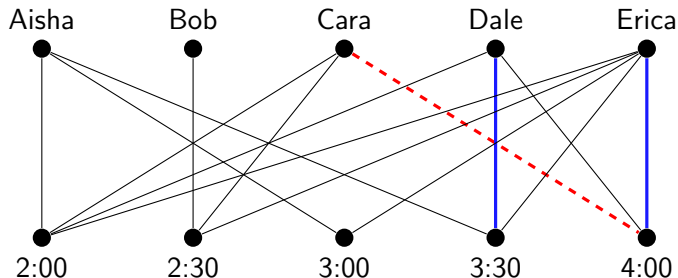
But if we had considered edges a different order...



# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

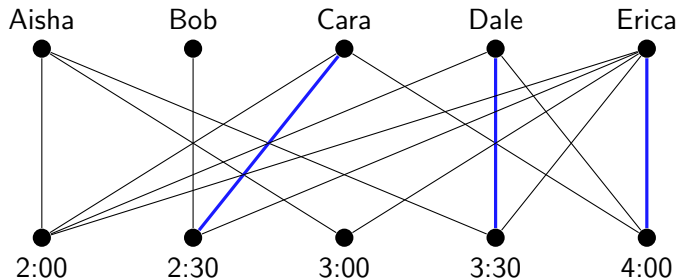


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

Can we form  $M$  by greedily adding edges? E.g.:

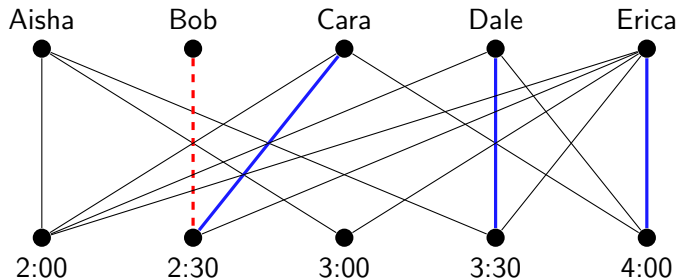


But if we had considered edges a different order...

# An algorithm for maximum matchings

**General problem statement:** Given a bipartite graph  $G = (V, E)$ , output a matching  $M$  which is as large as possible (i.e. **maximum**).

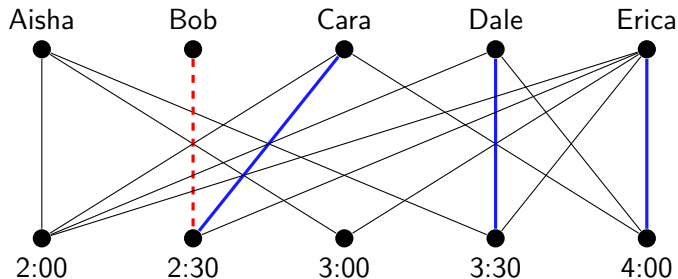
Can we form  $M$  by greedily adding edges? E.g.:



But if we had considered edges a different order... we wouldn't have been able to match Bob! So this algorithm **fails**.

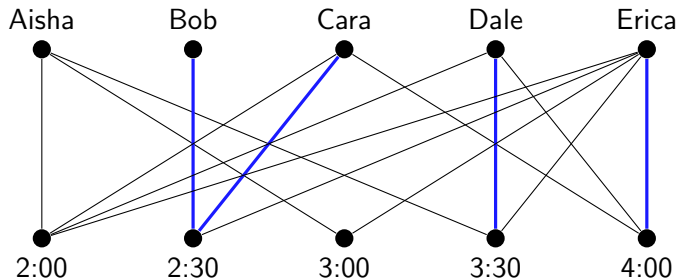
## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



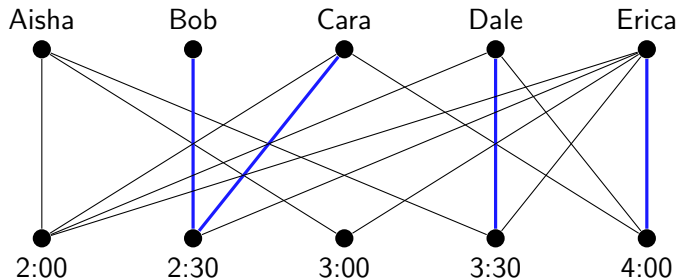
## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



## Repairing poor decisions: an example

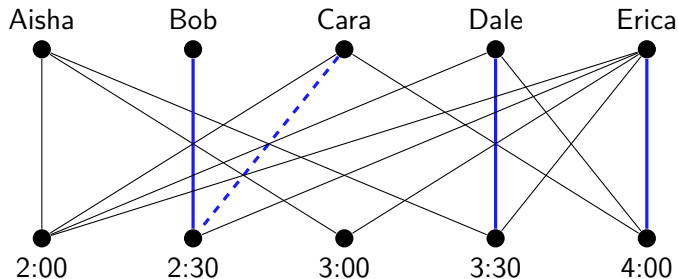
But maybe all is not lost... Say we try to force Bob into the matching.



This forces us to unmatched 2:30...

## Repairing poor decisions: an example

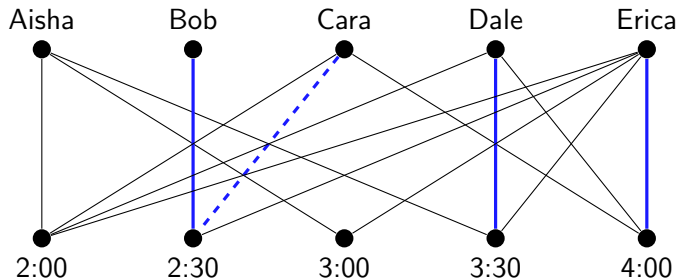
But maybe all is not lost... Say we try to force Bob into the matching.



This forces us to unmatched 2:30...

## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



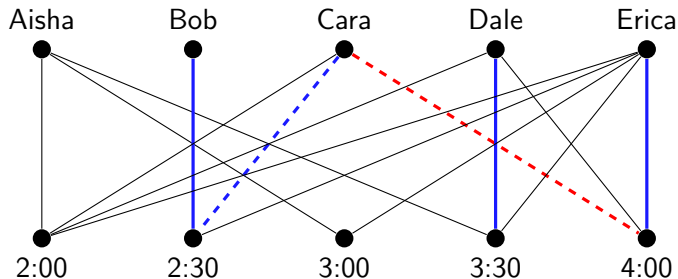
This forces us to unmatched 2:30...

Which leaves us free to rematch Cara...



## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.

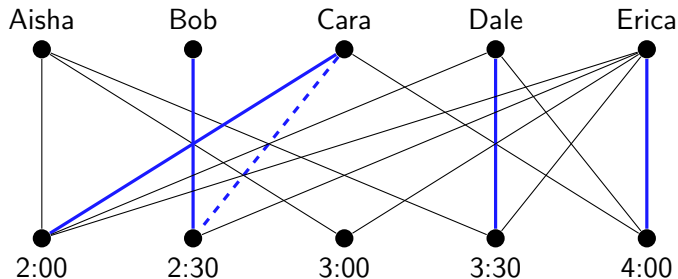


This forces us to unmatched 2:30...

Which leaves us free to rematch Cara...

## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.

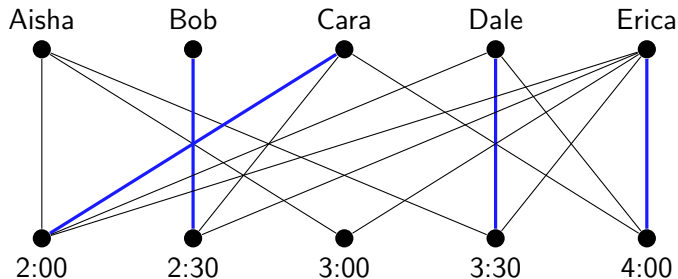


This forces us to unmatched 2:30...

Which leaves us free to rematch Cara...

## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



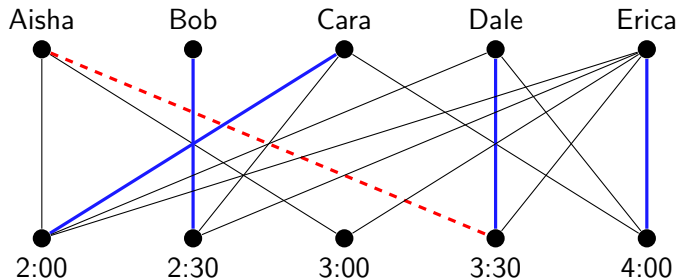
This forces us to unmatched 2:30...

Which leaves us free to rematch Cara...

Who can still meet us at 2:00. So we succeeded in matching Bob!

## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



This forces us to unmatched 2:30...

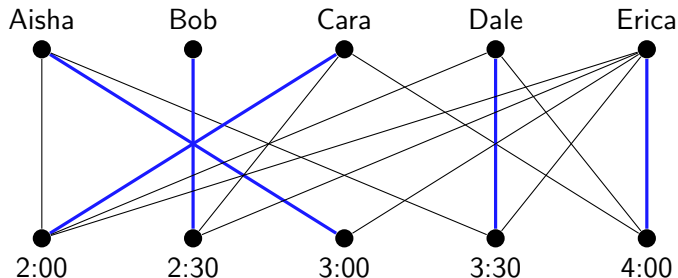
Which leaves us free to rematch Cara...

Who can still meet us at 2:00. So we succeeded in matching Bob!

And now we continue as before, and get a perfect matching.

## Repairing poor decisions: an example

But maybe all is not lost... Say we try to force Bob into the matching.



This forces us to unmatched 2:30...

Which leaves us free to rematch Cara...

Who can still meet us at 2:00. So we succeeded in matching Bob!

And now we continue as before, and get a perfect matching.

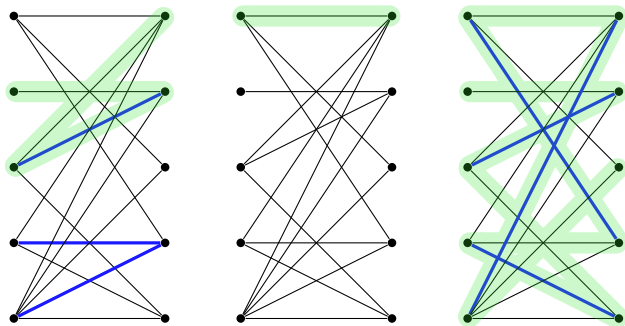
## Repairing poor decisions: the general method

Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path**  $P$  for  $M$  is a path in  $G$  which alternates between matching and non-matching edges, and which begins and ends with unmatched vertices.

# Repairing poor decisions: the general method

Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path**  $P$  for  $M$  is a path in  $G$  which alternates between matching and non-matching edges, and which begins and ends with unmatched vertices.

Formally, writing  $P = v_0 \dots v_k$ , we require  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ . For example:



Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path** for  $M$  is a path  $P = v_0 \dots v_k$  such that:

- $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ;
  - $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ ;
  - $v_0, v_k \notin \bigcup_{e \in M} e$ .
-



Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path** for  $M$  is a path  $P = v_0 \dots v_k$  such that:

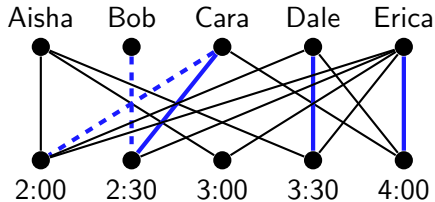
- $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ;
- $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ ;
- $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

If  $P$  is an augmenting path for  $M$ , we define

$$\text{Switch}(M, P) = M - \{\{v_i, v_{i+1}\} : i \text{ is odd}\} \cup \{\{v_i, v_{i+1}\} : i \text{ is even}\}.$$

Then  $\text{Switch}(M, P)$  is a matching containing one more edge than  $M$ .



Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path** for  $M$  is a path  $P = v_0 \dots v_k$  such that:

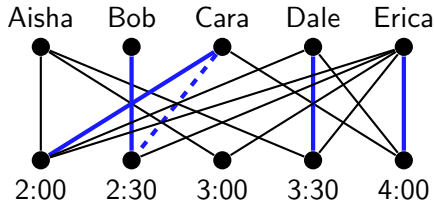
- $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ;
- $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ ;
- $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

If  $P$  is an augmenting path for  $M$ , we define

$$\text{Switch}(M, P) = M - \{\{v_i, v_{i+1}\} : i \text{ is odd}\} \cup \{\{v_i, v_{i+1}\} : i \text{ is even}\}.$$

Then  $\text{Switch}(M, P)$  is a matching containing one more edge than  $M$ .



Given a matching  $M$  in a bipartite graph  $G$ , an **augmenting path** for  $M$  is a path  $P = v_0 \dots v_k$  such that:

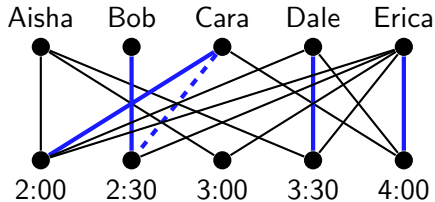
- $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ;
- $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ ;
- $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

If  $P$  is an augmenting path for  $M$ , we define

$$\text{Switch}(M, P) = M - \{\{v_i, v_{i+1}\} : i \text{ is odd}\} \cup \{\{v_i, v_{i+1}\} : i \text{ is even}\}.$$

Then  $\text{Switch}(M, P)$  is a matching containing one more edge than  $M$ .



This suggests a new greedy algorithm!

# A correct algorithm for maximum matchings

---

**Algorithm:** MAXMATCHING (SKETCH)

---

**Input** : A bipartite graph  $G = (V, E)$ .

**Output** : A list of edges forming a matching in  $G$  of maximum size.

```
1 begin
2   Initialise  $M \leftarrow []$ , the empty matching.
3   while  $G$  contains an augmenting path for  $M$  do
4     Find an augmenting path  $P$  for  $M$ .
5     Update  $M \leftarrow \text{Switch}(M, P)$ .
6   Return  $M$ .
```

---

To make this work, we need to do two things:

# A correct algorithm for maximum matchings

---

**Algorithm:** MAXMATCHING (SKETCH)

---

**Input** : A bipartite graph  $G = (V, E)$ .

**Output** : A list of edges forming a matching in  $G$  of maximum size.

```
1 begin
2   Initialise  $M \leftarrow []$ , the empty matching.
3   while  $G$  contains an augmenting path for  $M$  do
4     Find an augmenting path  $P$  for  $M$ .
5     Update  $M \leftarrow \text{Switch}(M, P)$ .
6   Return  $M$ .
```

---

To make this work, we need to do two things:

- Find an efficient way to find an augmenting path whenever one exists.

# A correct algorithm for maximum matchings

---

**Algorithm:** MAXMATCHING (SKETCH)

---

**Input** : A bipartite graph  $G = (V, E)$ .

**Output** : A list of edges forming a matching in  $G$  of maximum size.

1 **begin**

2     Initialise  $M \leftarrow []$ , the empty matching.

3     **while**  $G$  contains an augmenting path for  $M$  **do**

4         Find an augmenting path  $P$  for  $M$ .

5         Update  $M \leftarrow \text{Switch}(M, P)$ .

6     Return  $M$ .

---

To make this work, we need to do two things:

- Find an efficient way to find an augmenting path whenever one exists.
- Prove that if  $M$  has no augmenting paths, then  $M$  is maximum.

## Finding augmenting paths efficiently

If we search by brute force, this could take  $\Theta(|V|!)$  time! Let's not.

## Finding augmenting paths efficiently

If we search by brute force, this could take  $\Theta(|V|!)$  time! Let's not.

One general theme of this course: solve a complex problem by applying an algorithm for a simple problem in a clever way. We call this **reducing** the complex problem to the simple one.



## Finding augmenting paths efficiently

If we search by brute force, this could take  $\Theta(|V|!)$  time! Let's not.

One general theme of this course: solve a complex problem by applying an algorithm for a simple problem in a clever way. We call this **reducing** the complex problem to the simple one.

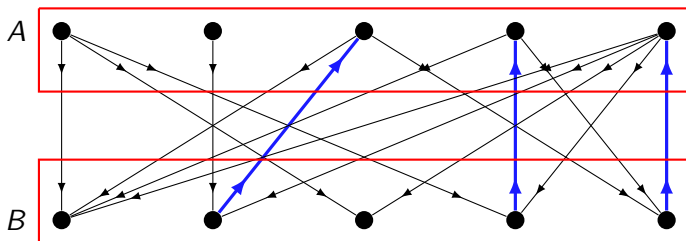
Here, we reduce the problem of finding an augmenting path to a problem we can already solve: finding a path from one set to another in a **directed** graph, via breadth-first search.

(For how to apply breadth-first search to sets instead of vertices, see last week's problem sheet — this is itself a reduction!)

Suppose  $G = (V, E)$  has a matching  $M$  and a bipartition  $(A, B)$ .  
 Turn  $G$  into an auxiliary digraph  $D_{G,M}$  by directing non-matching edges from  $A$  to  $B$  and matching edges from  $B$  to  $A$ . Formally:

$$V(D_{G,M}) := V,$$

$$E(D_{G,M}) := \{(a, b) : a \in A, b \in B, \{a, b\} \in E \setminus M\} \cup \\ \{(b, a) : a \in A, b \in B, \{a, b\} \in M\}.$$



$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

**Proof:** First note any augmenting path in  $G$  has endpoints in  $U \cap A$  and  $U \cap B$ , since it has an odd number of edges and  $G$  is bipartite.

So let  $P = v_0 \dots v_k$  be any path in  $G$  with  $v_0 \in U \cap A$ ,  $v_k \in U \cap B$ . We show  $P$  is augmenting for  $M$  iff it is also a path in  $D_{G,M}$ .

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

**Proof:** First note any augmenting path in  $G$  has endpoints in  $U \cap A$  and  $U \cap B$ , since it has an odd number of edges and  $G$  is bipartite.

So let  $P = v_0 \dots v_k$  be any path in  $G$  with  $v_0 \in U \cap A$ ,  $v_k \in U \cap B$ . We show  $P$  is augmenting for  $M$  iff it is also a path in  $D_{G,M}$ .

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

**Proof:** First note any augmenting path in  $G$  has endpoints in  $U \cap A$  and  $U \cap B$ , since it has an odd number of edges and  $G$  is bipartite.

So let  $P = v_0 \dots v_k$  be any path in  $G$  with  $v_0 \in U \cap A$ ,  $v_k \in U \cap B$ . We show  $P$  is augmenting for  $M$  iff it is also a path in  $D_{G,M}$ .

- $G$  is bipartite  $\Rightarrow v_i \in A$  for all even  $i$  and  $v_i \in B$  for all odd  $i$ . So:



$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

**Proof:** First note any augmenting path in  $G$  has endpoints in  $U \cap A$  and  $U \cap B$ , since it has an odd number of edges and  $G$  is bipartite.

So let  $P = v_0 \dots v_k$  be any path in  $G$  with  $v_0 \in U \cap A$ ,  $v_k \in U \cap B$ . We show  $P$  is augmenting for  $M$  iff it is also a path in  $D_{G,M}$ .

- $G$  is bipartite  $\Rightarrow v_i \in A$  for all even  $i$  and  $v_i \in B$  for all odd  $i$ . So:
- $\{v_i, v_{i+1}\} \in M$  for all odd  $i \Leftrightarrow (v_i, v_{i+1}) \in E(D_{G,M})$  for all odd  $i$ ;

$D_{G,M}$  is defined by directing edges outside  $M$  from  $A$  to  $B$ ,  
and edges in  $M$  from  $B$  to  $A$ .

$P = v_0 \dots v_k$  is **augmenting** if  $\{v_i, v_{i+1}\} \in M$  for all odd  $i$ ,  
 $\{v_i, v_{i+1}\} \notin M$  for all even  $i$ , and  $v_0, v_k \notin \bigcup_{e \in M} e$ .

---

Let  $U = V \setminus \bigcup_{e \in M} e$  be the set of vertices not matched by  $M$ .

**Lemma:** A path in  $G$  is augmenting for  $M$  if and only if it's also a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$ .

**Proof:** First note any augmenting path in  $G$  has endpoints in  $U \cap A$  and  $U \cap B$ , since it has an odd number of edges and  $G$  is bipartite.

So let  $P = v_0 \dots v_k$  be any path in  $G$  with  $v_0 \in U \cap A$ ,  $v_k \in U \cap B$ . We show  $P$  is augmenting for  $M$  iff it is also a path in  $D_{G,M}$ .

- $G$  is bipartite  $\Rightarrow v_i \in A$  for all even  $i$  and  $v_i \in B$  for all odd  $i$ . So:
- $\{v_i, v_{i+1}\} \in M$  for all odd  $i \Leftrightarrow (v_i, v_{i+1}) \in E(D_{G,M})$  for all odd  $i$ ;
- $\{v_i, v_{i+1}\} \notin M$  for all even  $i \Leftrightarrow (v_i, v_{i+1}) \in E(D_{G,M})$  for all even  $i$ .  $\square$

---

**Algorithm:** MAXMATCHING

---

**Input** : A bipartite graph  $G = (V, E)$ .

**Output** : A list of edges forming a matching in  $G$  of maximum size.

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

---

---

**Algorithm:** MAXMATCHING

---

**Input** : A bipartite graph  $G = (V, E)$ .

**Output** : A list of edges forming a matching in  $G$  of maximum size.

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

---

**Invariant:** At the start of the  $i$ th loop iteration,  $M$  is a matching with  $i - 1$  edges.  $M$  can have at most  $|V|/2$  edges in total, so MAXMATCHING outputs a matching with no augmenting paths.

---

**Algorithm: MAXMATCHING**

---

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

---

---

**Algorithm: MAXMATCHING**

---

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

---

- Steps 2, 4 and 6 can all be done in  $O(|E|)$  time. (Exercise!)

---

**Algorithm: MAXMATCHING**

---

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

- 
- Steps 2, 4 and 6 can all be done in  $O(|E|)$  time. (Exercise!)
  - Step 5 can be done in  $O(|E|)$  time using breadth-first search, if  $G$  is in adjacency-list form.

---

**Algorithm: MAXMATCHING**

---

```
1 begin
2   Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .
3   repeat
4     Form the graph  $D_{G,M}$ .
5     Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.
6     Otherwise, break.
7     Update  $M \leftarrow \text{Switch}(M, P)$ .
8   Return  $M$ .
```

---

- Steps 2, 4 and 6 can all be done in  $O(|E|)$  time. (Exercise!)
- Step 5 can be done in  $O(|E|)$  time using breadth-first search, if  $G$  is in adjacency-list form.
- Steps 4–6 repeat at most  $|V|$  times.



---

**Algorithm: MAXMATCHING**

---

1 **begin**

2 Find a bipartition  $(A, B)$  of  $G$ . Initialise  $M \leftarrow []$ .

3 **repeat**

4 Form the graph  $D_{G,M}$ .

5 Set  $P$  to be a path from  $U \cap A$  to  $U \cap B$  in  $D_{G,M}$  if one exists.

Otherwise, **break**.

6 Update  $M \leftarrow \text{Switch}(M, P)$ .

7 **Return**  $M$ .

---

- Steps 2, 4 and 6 can all be done in  $O(|E|)$  time. (Exercise!)
- Step 5 can be done in  $O(|E|)$  time using breadth-first search, if  $G$  is in adjacency-list form.
- Steps 4–6 repeat at most  $|V|$  times.

So overall the running time is  $O(|E||V|)$ .

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

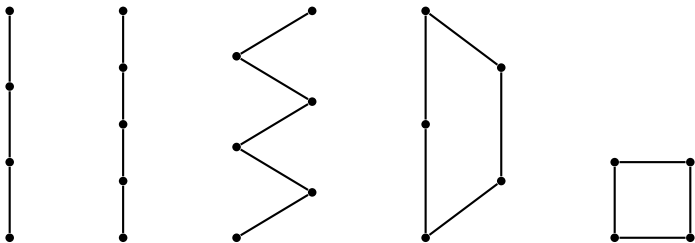
**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.  
We suppose  $M$  is **not** maximum, and find an augmenting path.

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



Since each vertex is in at most one  $M$  edge and at most one  $M'$  edge,  $S$  has maximum degree at most 2.

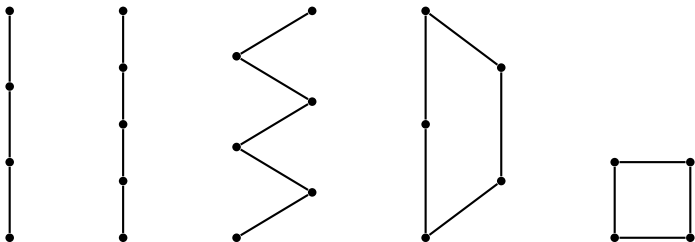
So  $S$  is a disjoint union of path and cycle components.

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



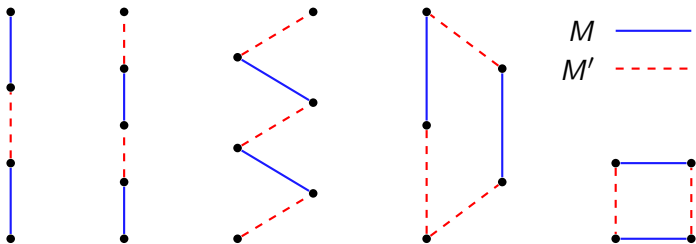
Since  $M$  and  $M'$  are matchings, each component's edges must alternate between  $M'$  and  $M$ . (In particular, no odd cycles!)

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



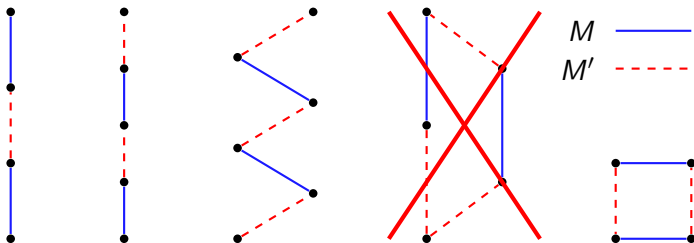
Since  $M$  and  $M'$  are matchings, each component's edges must alternate between  $M'$  and  $M$ . (In particular, no odd cycles!)

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



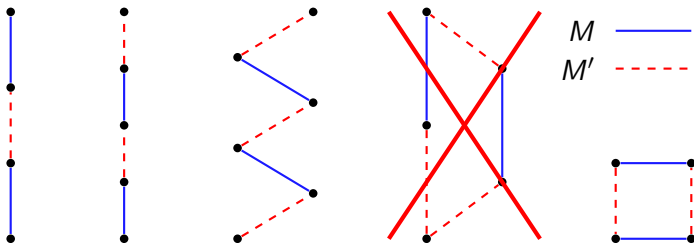
Since  $M$  and  $M'$  are matchings, each component's edges must alternate between  $M'$  and  $M$ . (In particular, no odd cycles!)

**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



Since  $|M'| > |M|$ , some component  $C$  has more  $M'$ -edges than  $M$ -edges. Since  $M'$ -edges and  $M$ -edges alternate, it has exactly **one** more  $M'$ -edge.

$G$  is bipartite, so it has no odd cycles, so  $C$  must be a path starting and ending with an  $M'$ -edge — an augmenting path.  $\square$

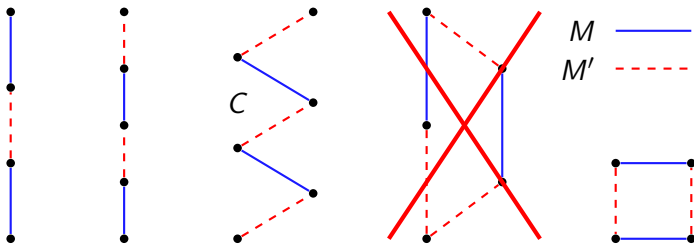


**Berge's Lemma:**  $M$  has no augmenting paths  $\Rightarrow M$  is maximum.

We suppose  $M$  is **not** maximum, and find an augmenting path.

Let  $M'$  be another matching which **is** maximum, so  $|M'| > |M|$ .

Consider the symmetric difference  $S = M \triangle M'$ , i.e. the graph formed of edges contained in either  $M$  or  $M'$  but not both.



Since  $|M'| > |M|$ , some component  $C$  has more  $M'$ -edges than  $M$ -edges. Since  $M'$ -edges and  $M$ -edges alternate, it has exactly **one** more  $M'$ -edge.

$G$  is bipartite, so it has no odd cycles, so  $C$  must be a path starting and ending with an  $M'$ -edge — an augmenting path.  $\square$

Recall that given a graph  $G$ , we proved that `MAXMATCHING` returns a matching  $M$  for  $G$  with no augmenting path in time  $O(|E||V|)$ .

Berge's Lemma tells us that  $M$  is maximum, so we're done!

Recall that given a graph  $G$ , we proved that `MAXMATCHING` returns a matching  $M$  for  $G$  with no augmenting path in time  $O(|E||V|)$ .

Berge's Lemma tells us that  $M$  is maximum, so we're done!

Let us celebrate with a matching pair of kittens.



D'awww.