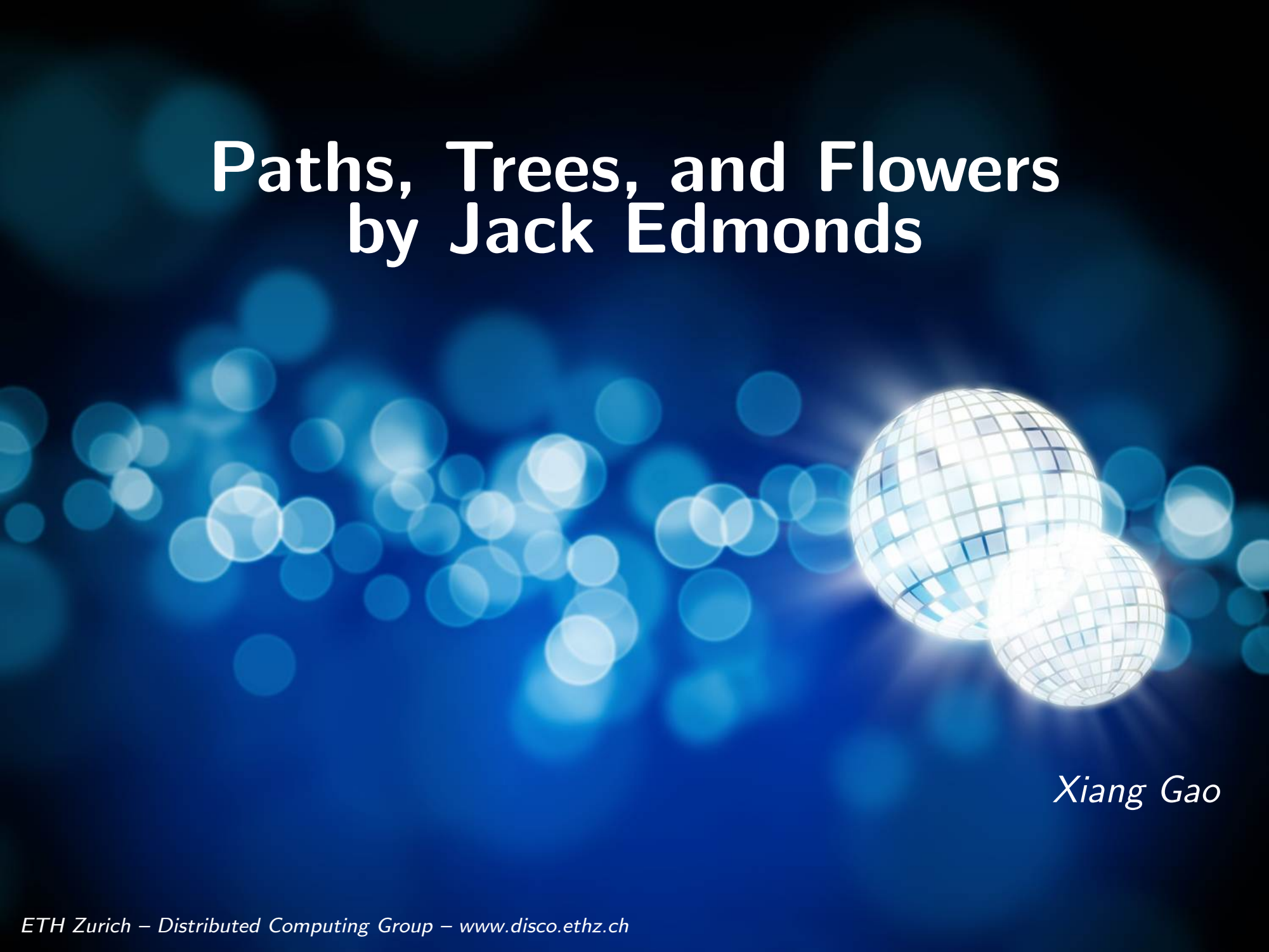
Paths, Trees, and Flowers by Jack Edmonds



Xiang Gao

ETH Zurich – Distributed Computing Group – www.disco.ethz.ch

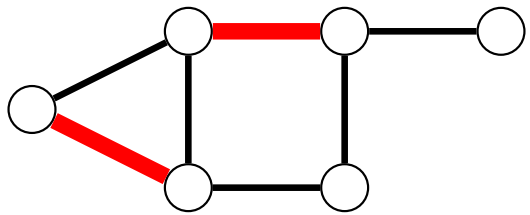
Introduction and background

Edmonds maximum matching algorithm

Matching-duality theorem

Matching

Matching in a graph is a set of edges, no two of which meet at a common vertex.

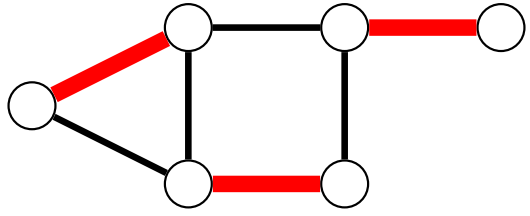


Maximum matching is a matching of maximum cardinality.

Matching

Matching in a graph is a set of edges, no two of which meet at a common vertex.

Maximum matching is a matching of maximum cardinality.

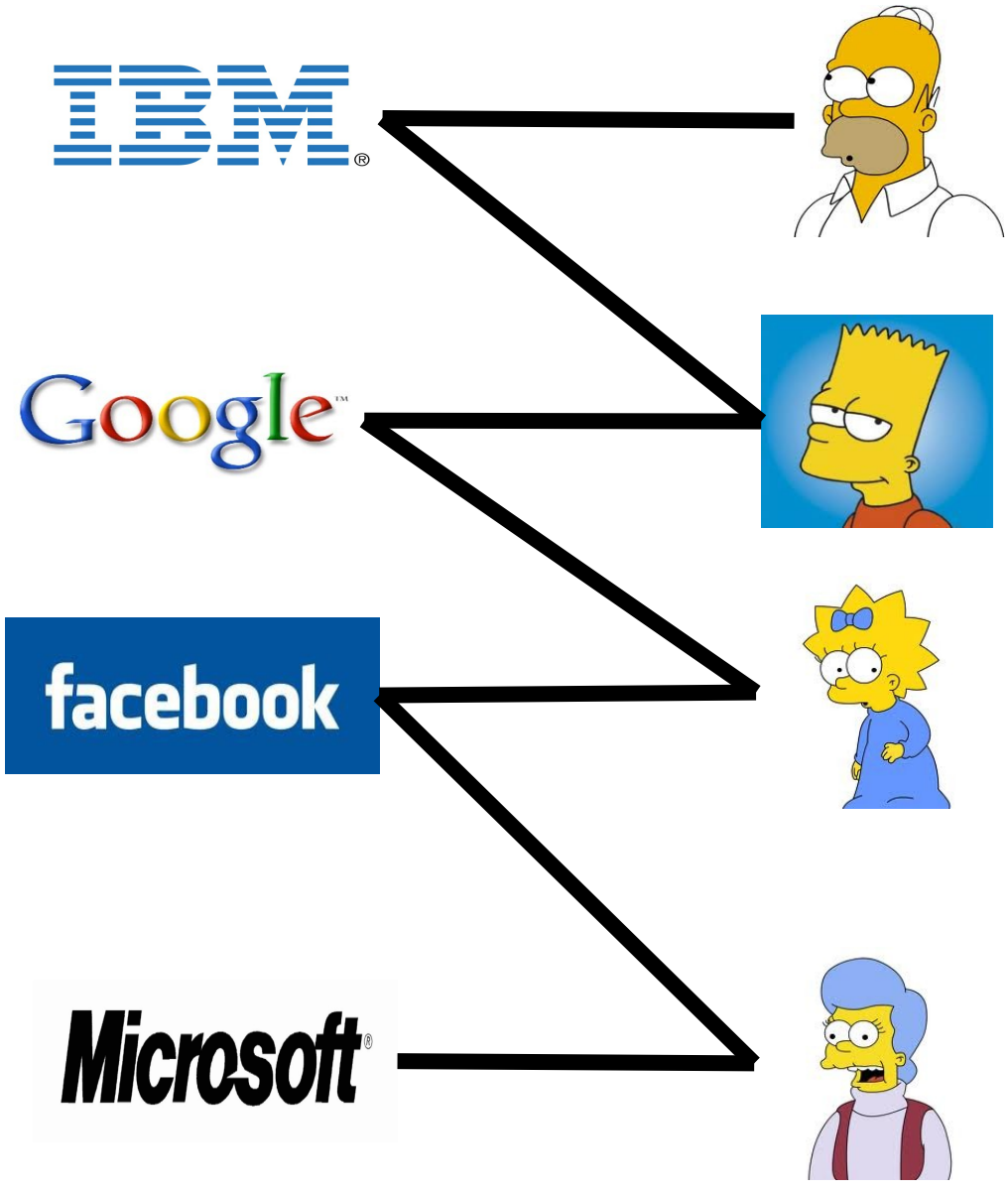


Question

Why do we want to study matching problems?

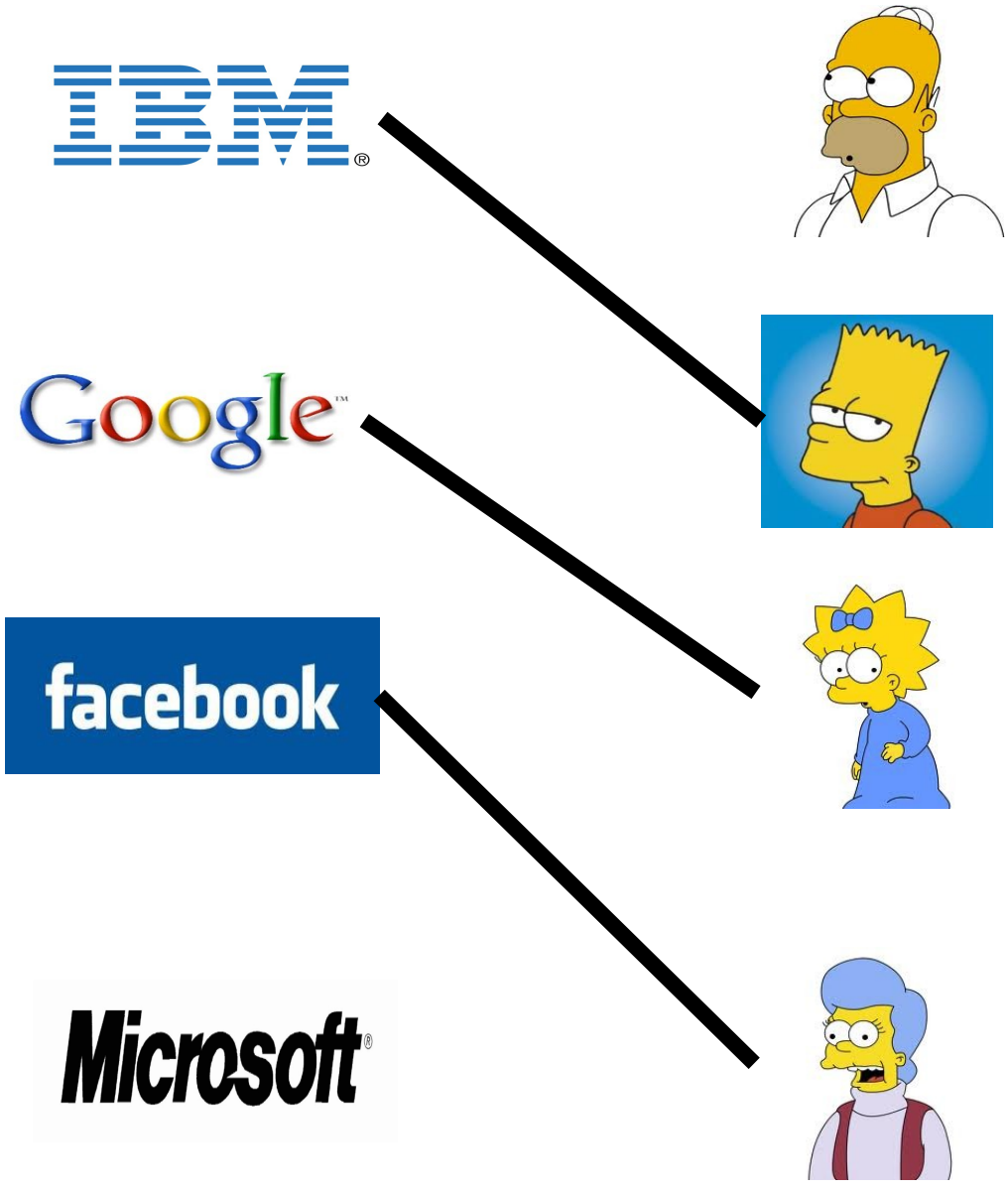
A lot of applications

Job recruitment process.



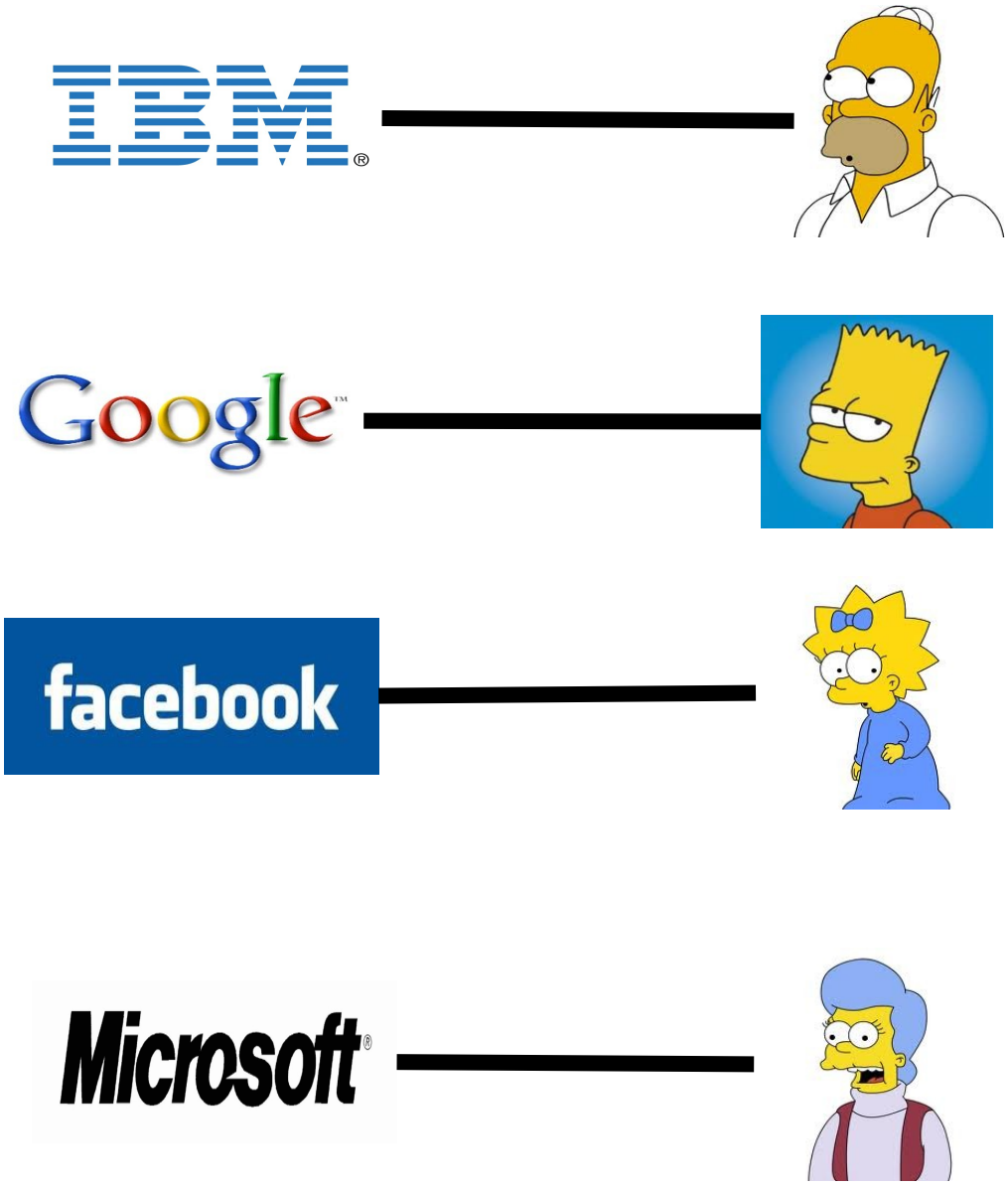
A lot of applications

Then someone is not happy.



A lot of applications

Everyone is happy now.



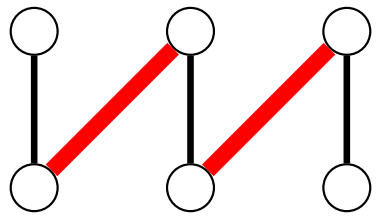
Question

You don't want to do it manually.

So, how can you find the maximum matching?

First try: greedy method

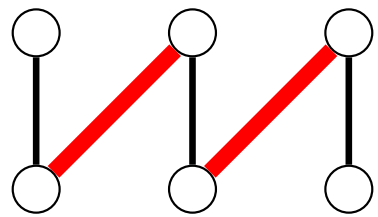
Enumerate every edge, if both end points are not covered by the matching, add the edge to the matching.



Not always work. May reach local optimal (maximal matching).

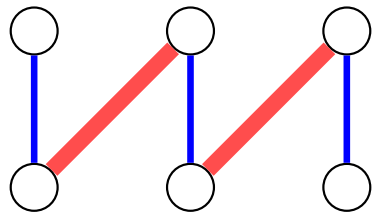
Question

How to improve the matching?



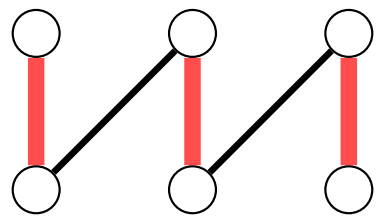
Augment the matching

If we flip the edges...



Augment the matching

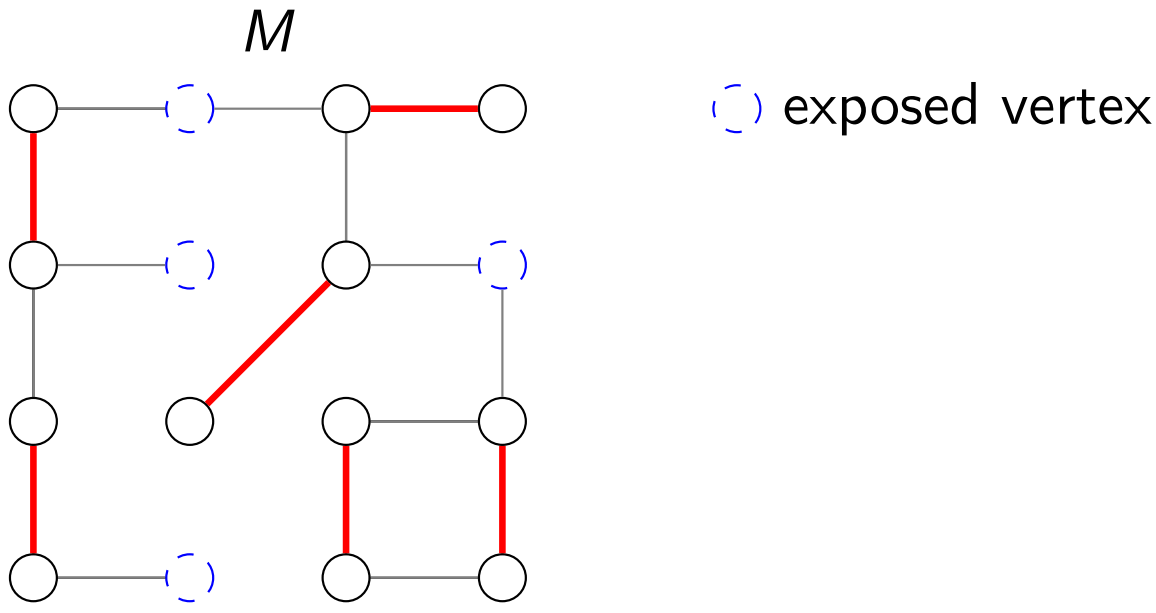
Seems we find a way to improve the matching!



Then we want to generalize the idea.

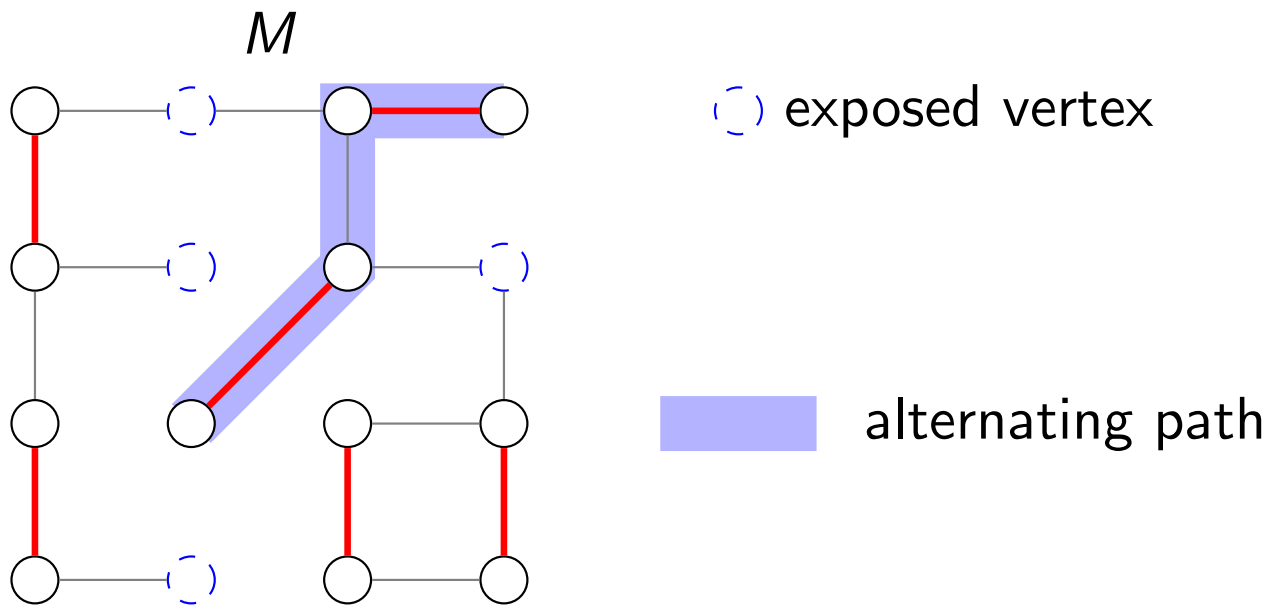
Exposed vertex

Exposed(free) vertex is a vertex that is not incident with any edge in the matching M



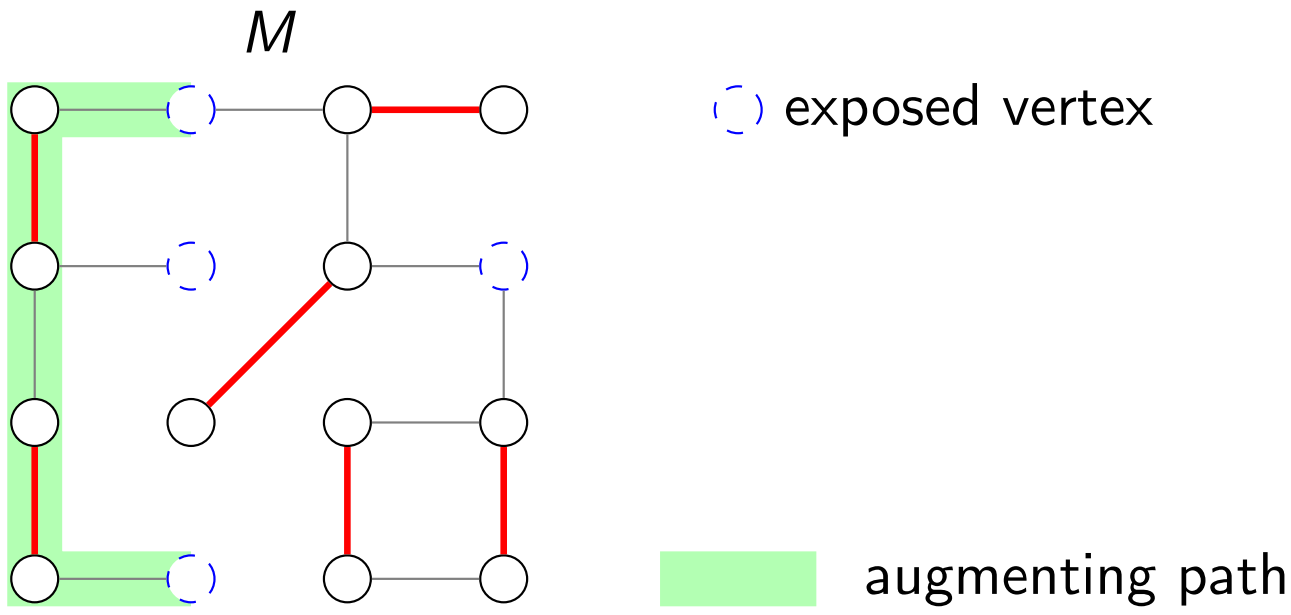
Alternating path

Alternating path is a path whose edges are alternately in M and $\overline{M}$



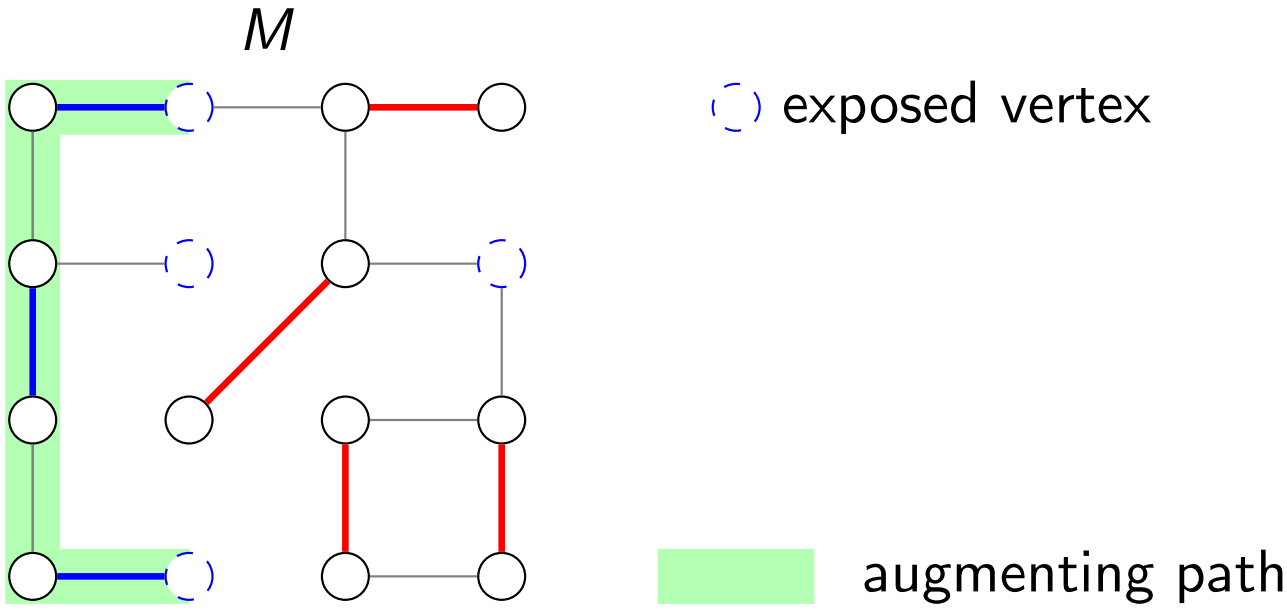
Augmenting path

Augmenting path is a simple alternating path between exposed vertices



Augmenting path

We can find a larger matching in the augmenting path!

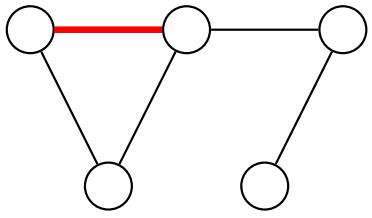


This operation is called symmetric difference $\oplus$.

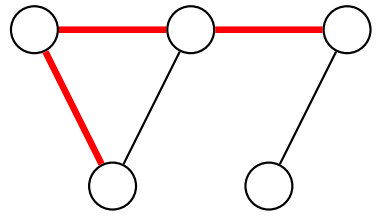
Symmetric difference

Symmetric difference of two sets D and E is defined as $D \oplus E = (D - E) \cup (E - D)$

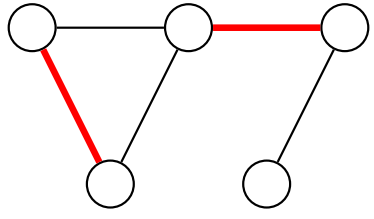
D



E



$D \oplus E$



The result of the symmetric difference between a matching and an augmenting path on the graph is a larger matching.

Question

Then our algorithm can try to find augmenting paths to improve the known mathcing.

Question

Can we stop when there is no augmenting paths?

What is the optimality condition for the algorithm to terminate?

Answer

Berge theorem says if there is no augmenting path, we can stop!

Berge's theorem

Theorem - Berge(1957)

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

- Proof:
  We have already seen that if there is an augmenting path we can find a larger matching.

  So we only need to prove if the matching is not maximum, there must exist an augmenting path.

Berge's theorem

If we have a larger matching, how to find an augmenting path?

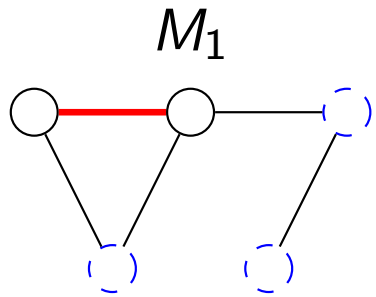
Berge's theorem

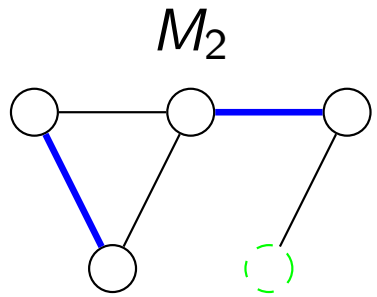
Recall that we use $\oplus$ to find a larger matching.

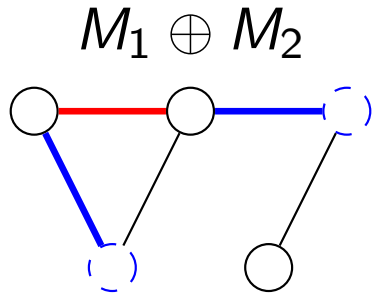
Now we reverse the step. If we apply $\oplus$ between matching M_1 and a larger matching M_2, what do we get?

Berge's theorem

The results are paths and circuits which are alternating for M_1 and M_2.







We have found alternating paths, which is very close to find augmenting paths!

Berge's theorem

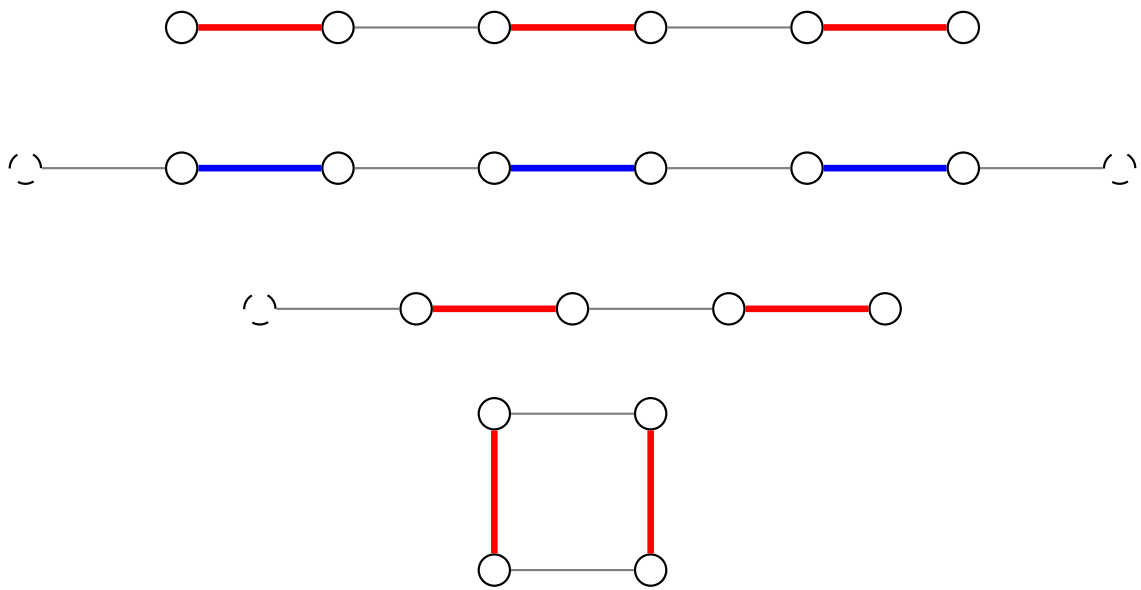
Since the matching M_2 is larger, there must be a component has one more edge in M_2 than in M_1.

This component is an augmenting path.

Why?

Berge's theorem

Because there are only four kinds of alternating paths. The only case when edges not in the matching are more than edges in the matching is an augmenting path!



Algorithm

Then we have a general algorithm, which keeps finding augmenting paths and improving the matching until there is no more augmenting path.

MAXIMUM-MATCHING(G)

```
1  M = ∅
2  repeat
3       if there is an augmenting path P with respect to M
4            M = M ⊕ P
5  until there isn't an augmenting path with respect to M
6  return M
```

Algorithm

How can we perform the search for an augmenting path efficiently? By brute force, the search space could be exponential.

And... what is an efficient algorithm?

Digression

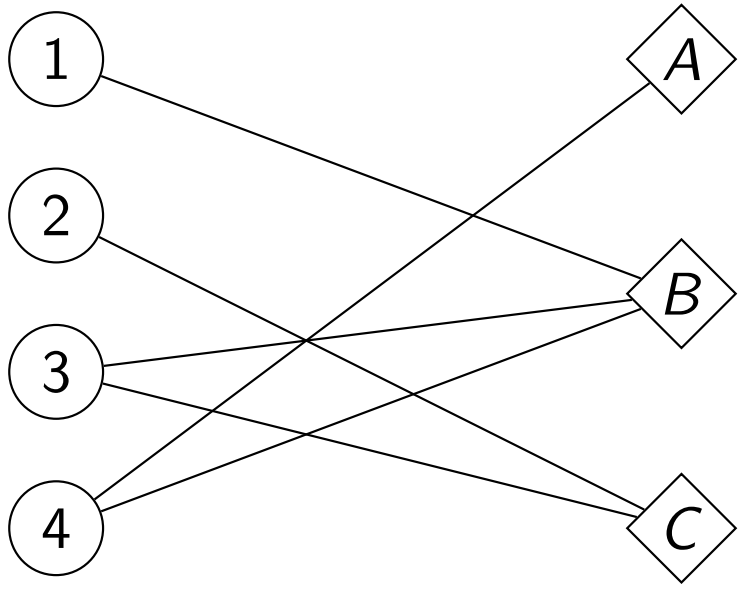
Edmonds defined that an efficient should has polynomial running time in this paper, which is earlier than Stephen Cook defined $P\&NP$.

Previous work: bipartite matching

A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.

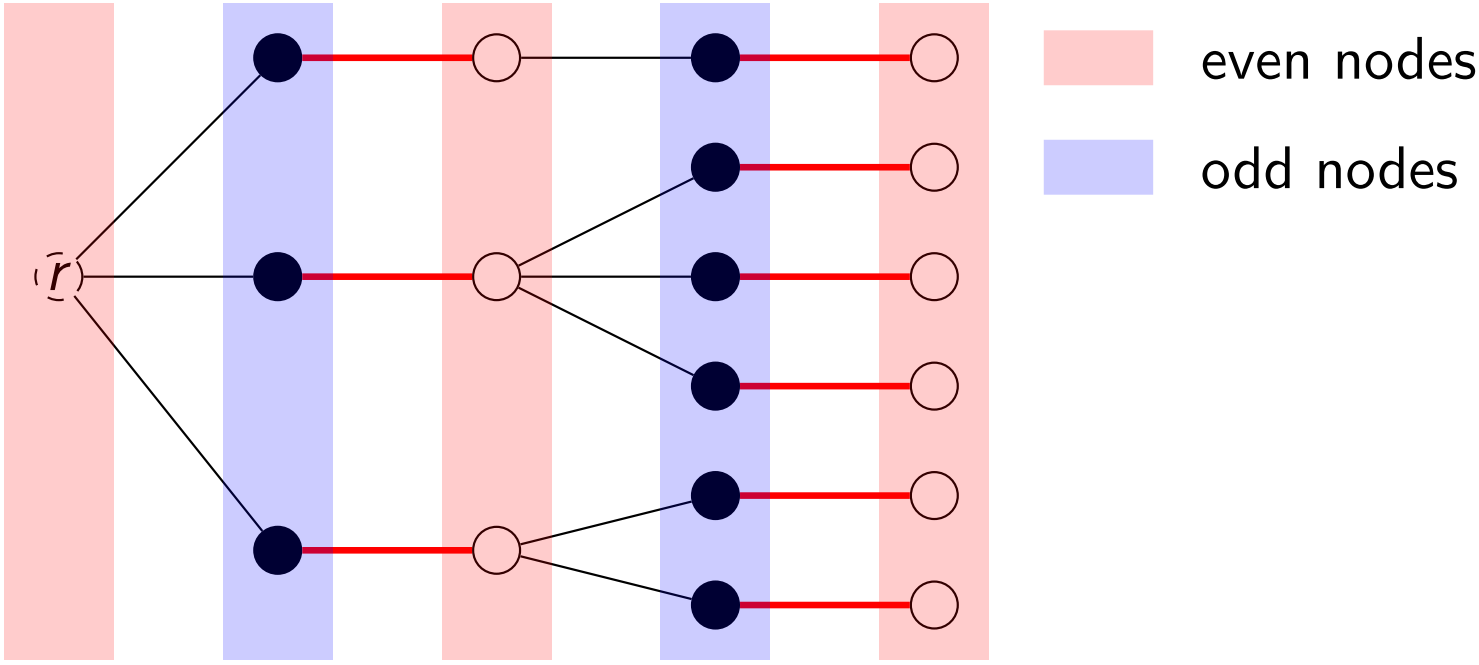
Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles!

There are polynomial time algorithms for maximum matching on bipartite graphs!



Find an augmenting path using BFS

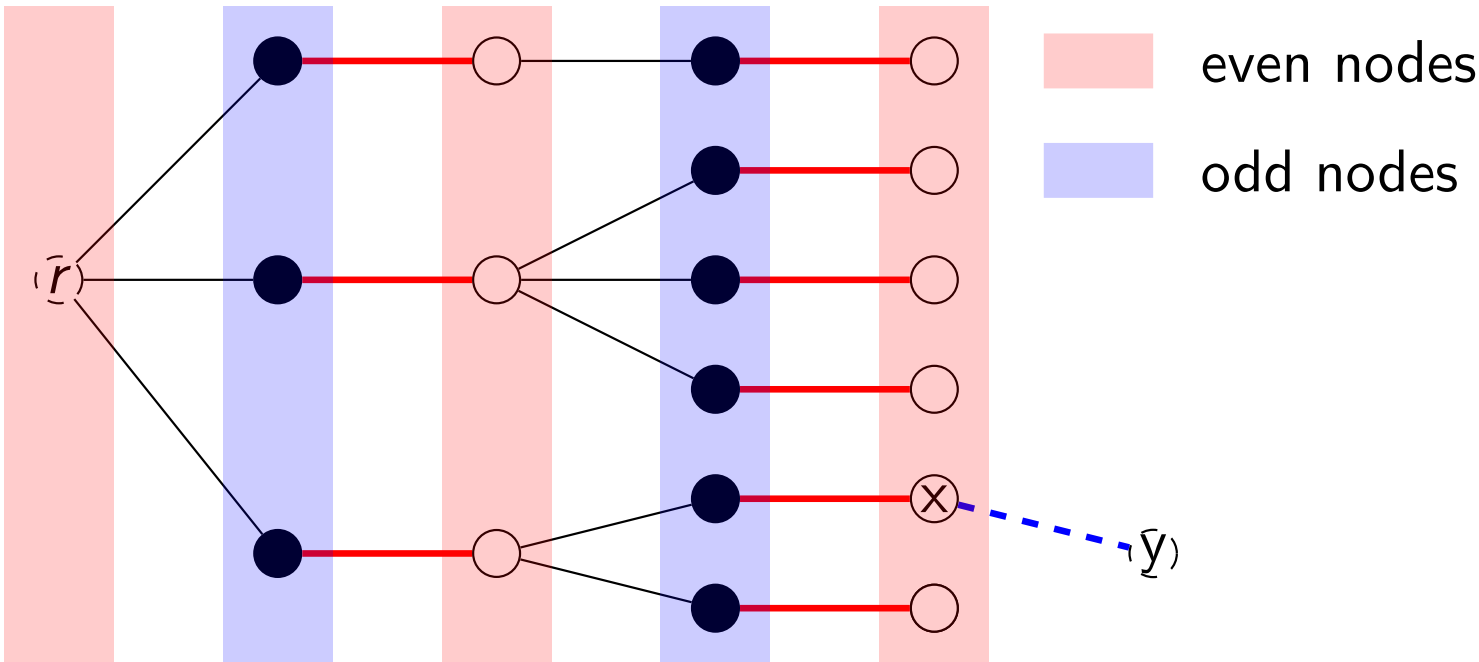
We can use some greedy heuristic to find an initial matching. Then we start from an exposed vertex to grow an alternating tree, since the augmenting path is an alternating path.



Find an augmenting path using BFS

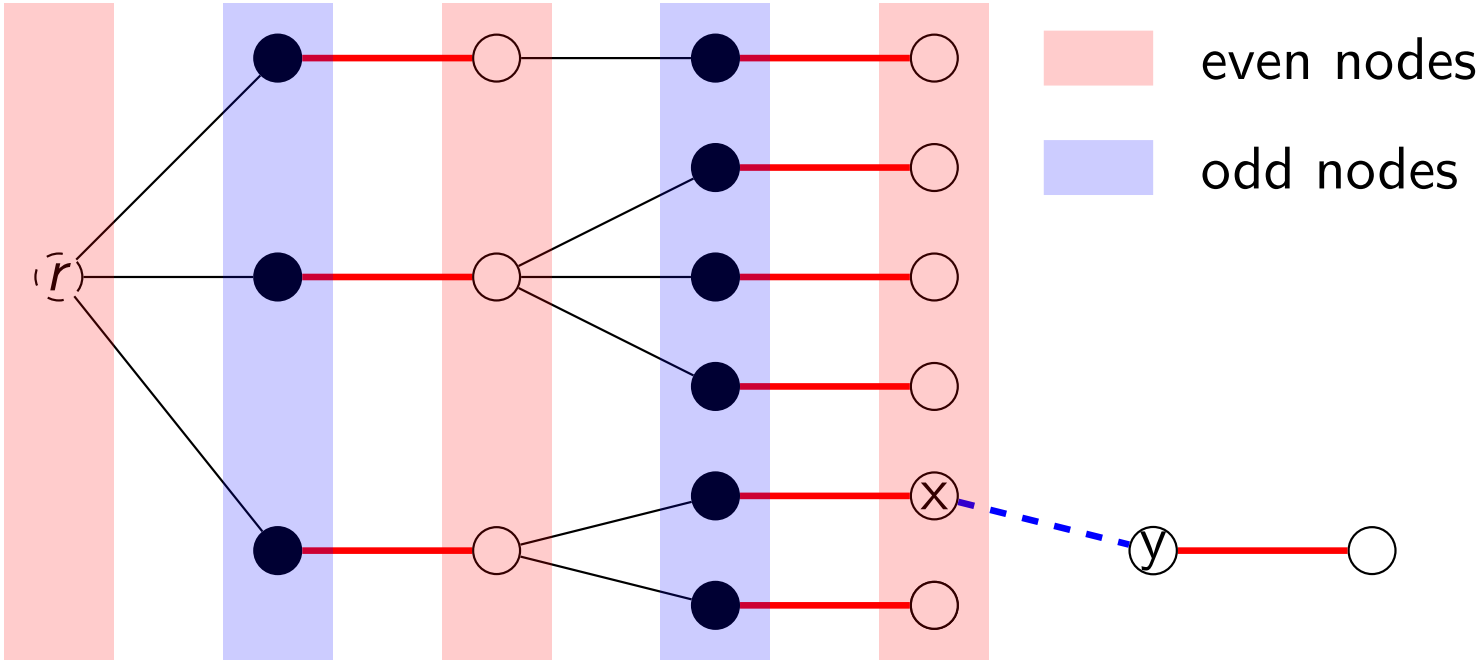
Case 1, y is an exposed vertex not contained in T.

We found an augmenting path!



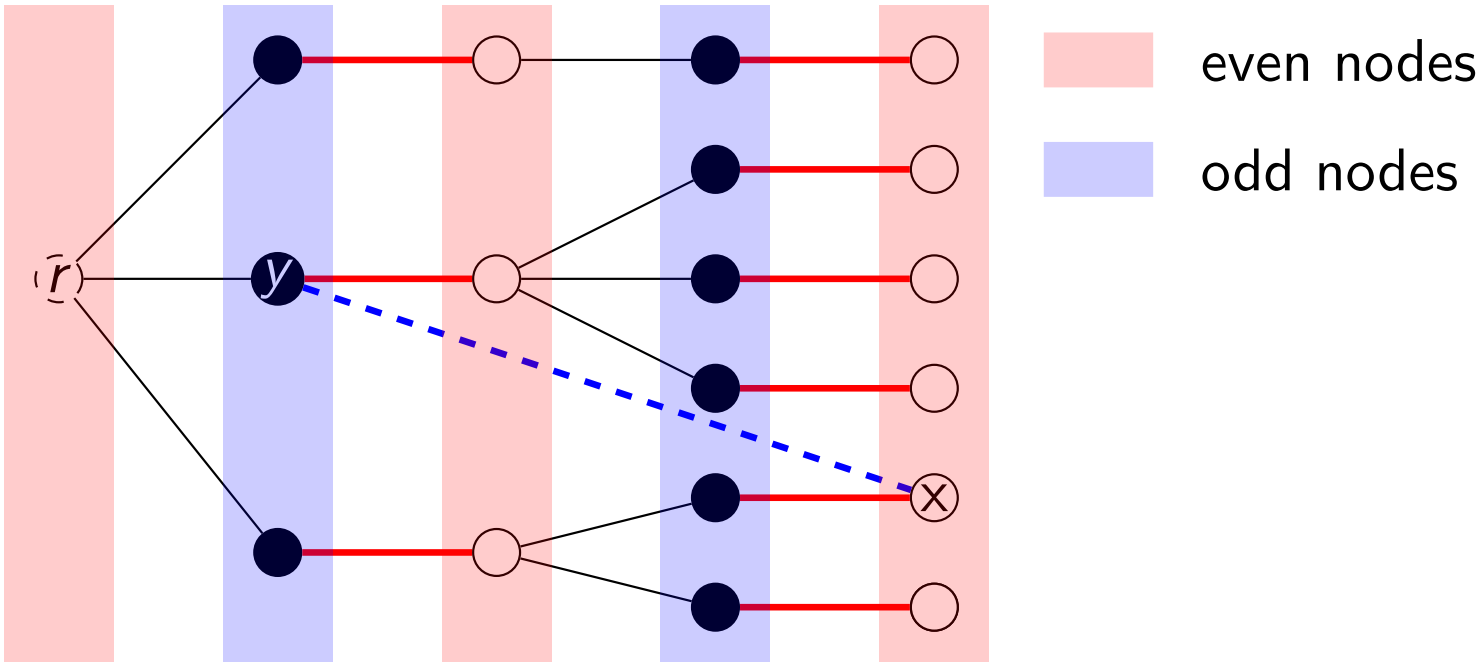
Find an augmenting path using BFS

Case 2, y is matched vertex not in T, grow the tree.



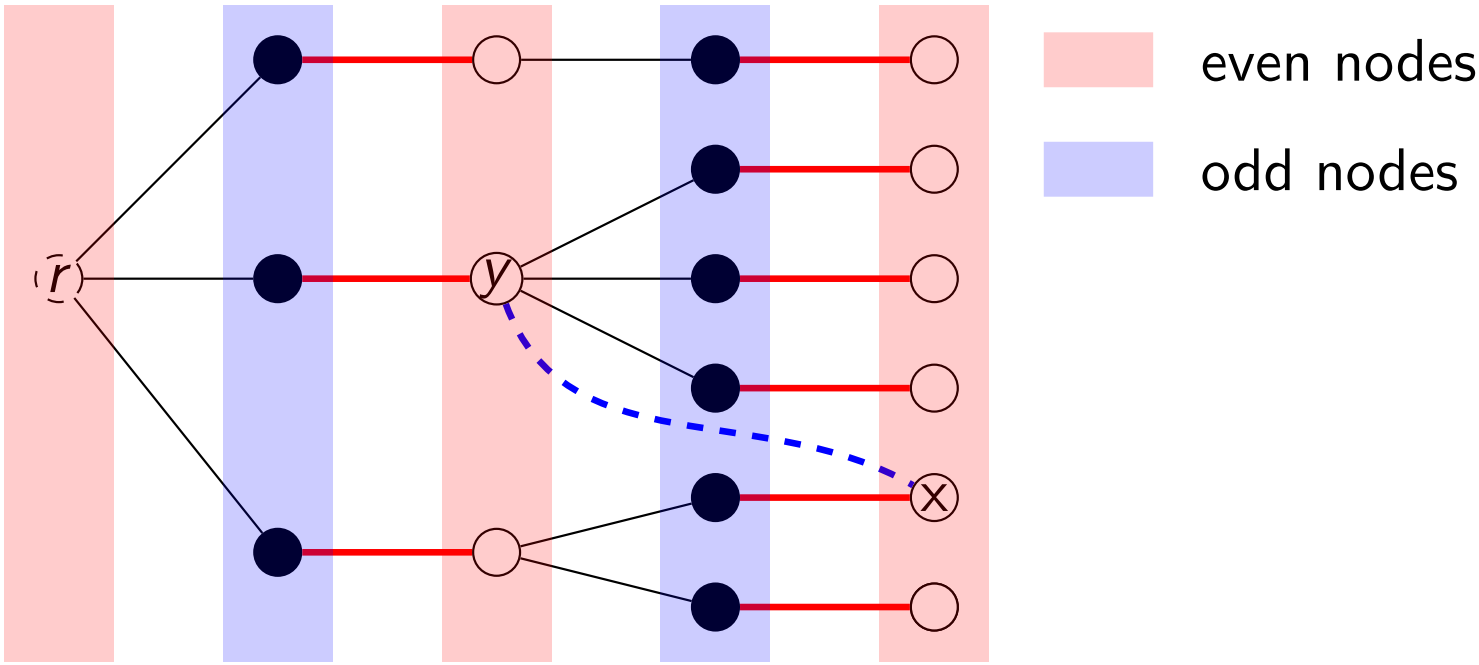
Find an augmenting path using BFS

Case 3, y is already contained in T as an odd vertex.
Ignore it, no worry! Why?



Find an augmenting path using BFS

Case 4, y is already contained in T as an even vertex.
Can't ignore y, but it doesn't happen on a bipartite graph (no odd cycle)!



Summary

The algorithm grow an alternating tree from an exposed vertex, if we find an exposed vertex, we can enlarge the matching and start from another exposed vertex growing the tree again.

After growing trees from all the exposed vertices, the algorithm terminates.

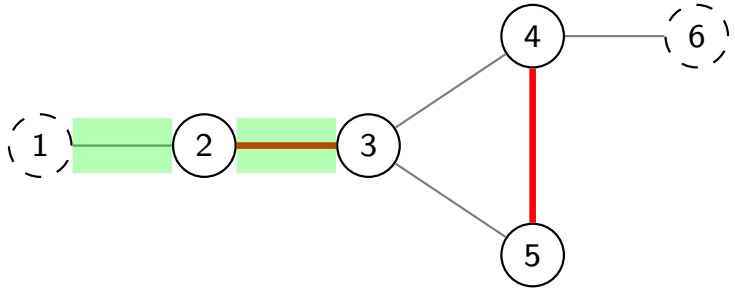
Time complexity $O(nm)$, n is the number of vertices, m is the number of edges.

Question

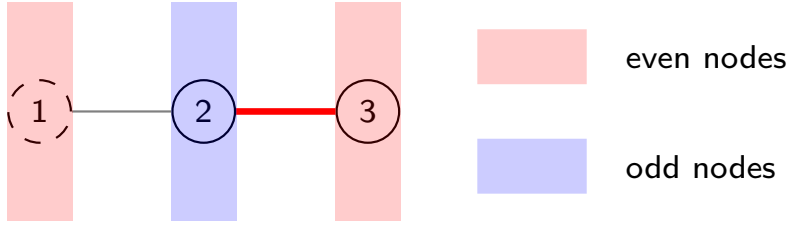
Why will odd cycles be a trouble?

Examples of BFS failure on non-bipartite graphs

- Initial graph

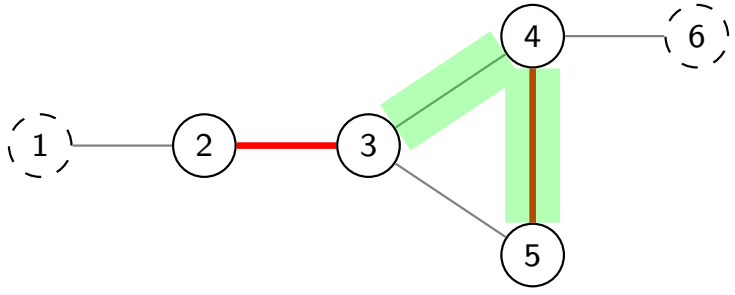


- Start from vertex 1, we add vertex 2 and 3.

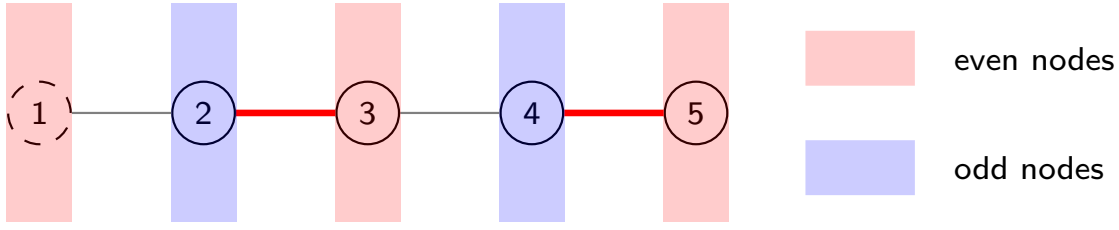


Examples of BFS failure on non-bipartite graphs

- Initial graph

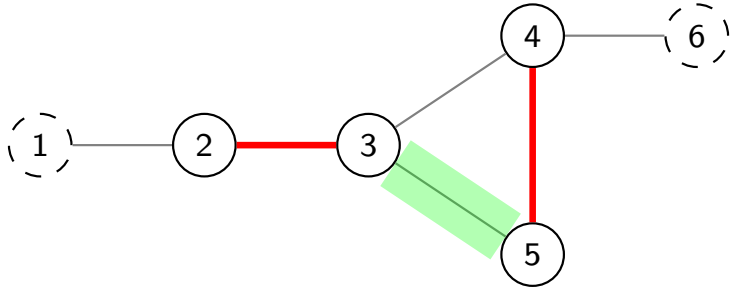


- Two vertices 4 and 5 could be visited. If we first visit 4...

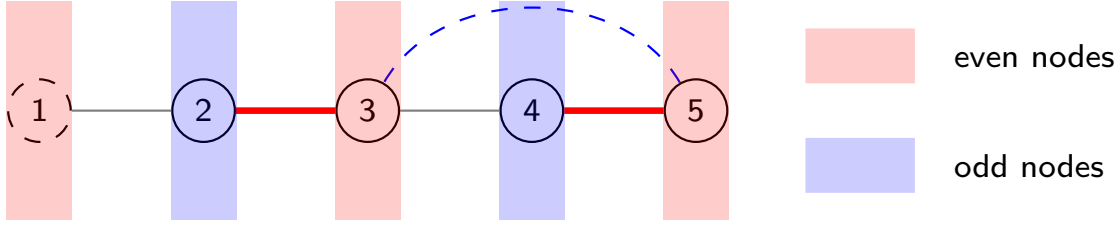


Examples of BFS failure on non-bipartite graphs

- Initial graph

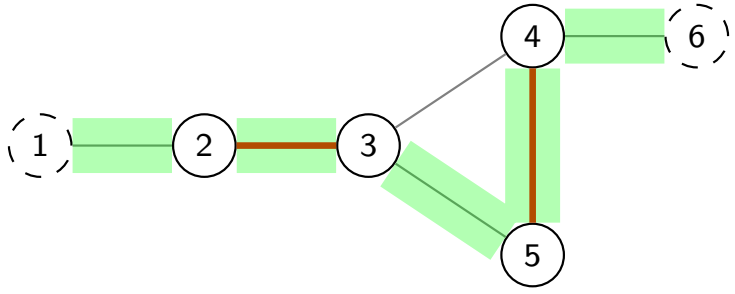


- From 5, we find 3 which is contained in T, if we ignore it, we will stop here.

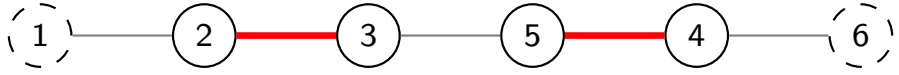


Examples of BFS failure on non-bipartite graphs

- Initial graph

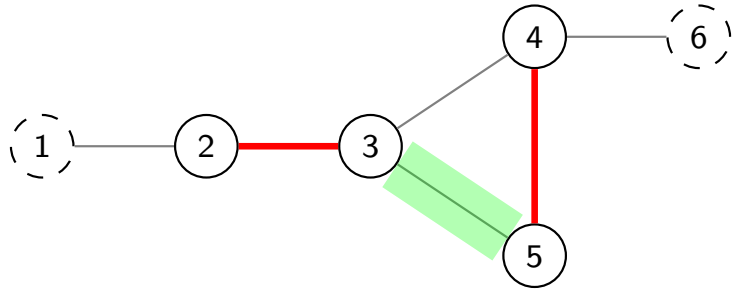


- But actually there is an augmenting path!

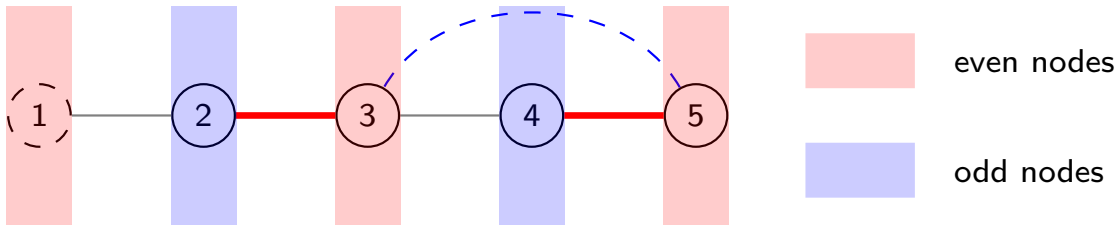


Examples of BFS failure on non-bipartite graphs

- Initial graph

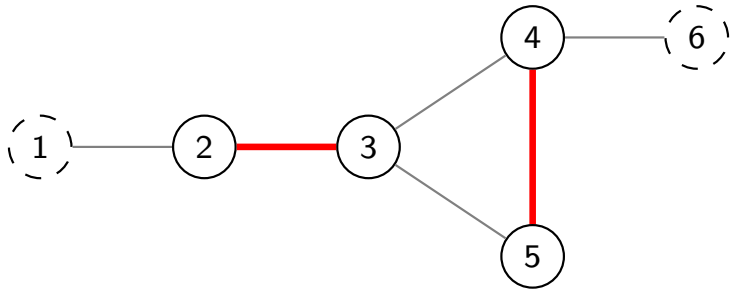


- The problem is that 3 already been visited in even layer, and 5 is also in even layer. But the tree must alternate between different layers.

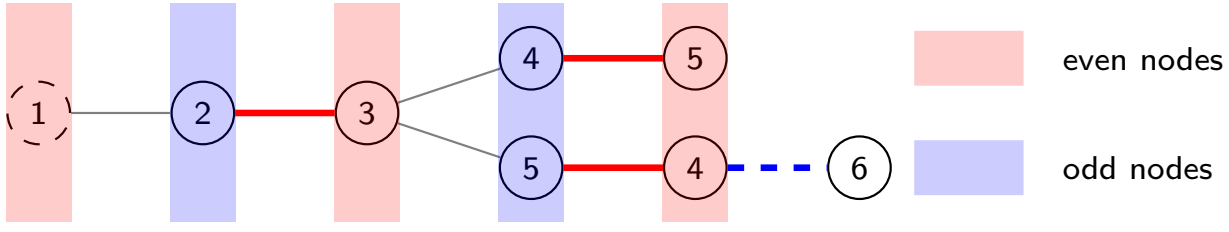


Examples of BFS failure on non-bipartite graphs

- Initial graph

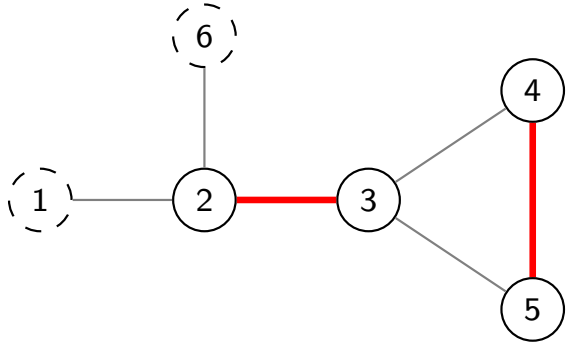


- First idea: allow vertices to be visited both even and odd layers.

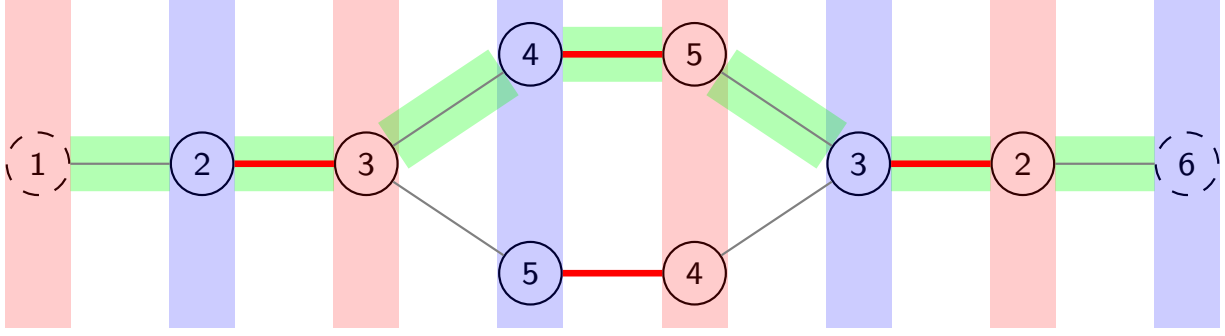


Examples of BFS failure on non-bipartite graphs

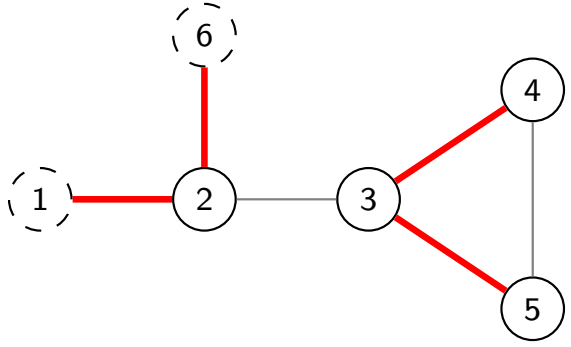
- Initial graph



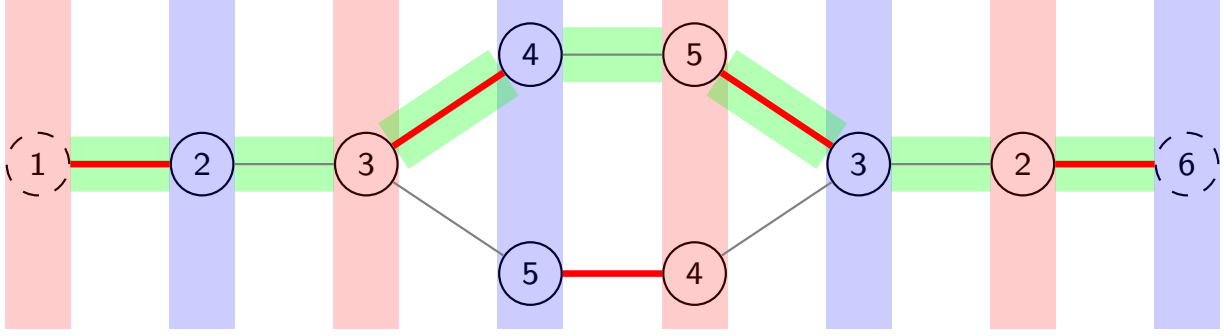
- But the augmenting path we find may not be a simple path



Examples of BFS failure on non-bipartite graphs

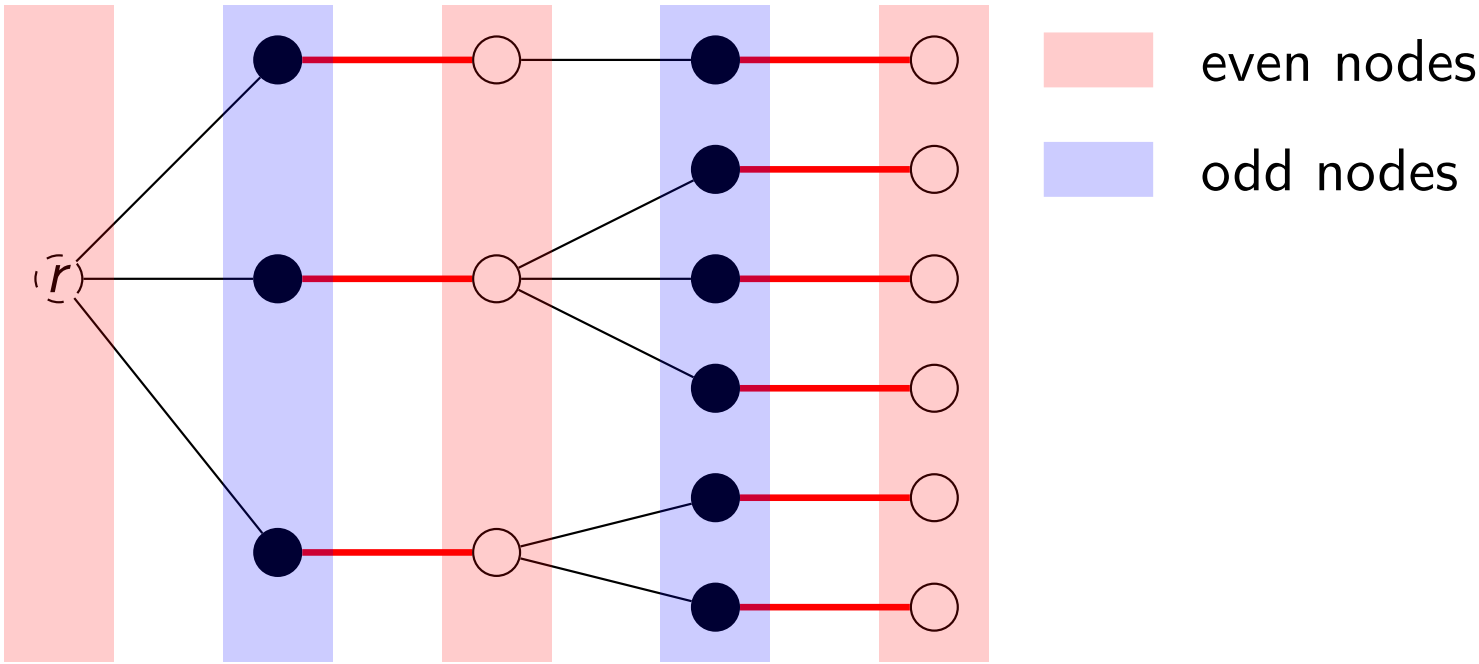


- If we flip the edges...



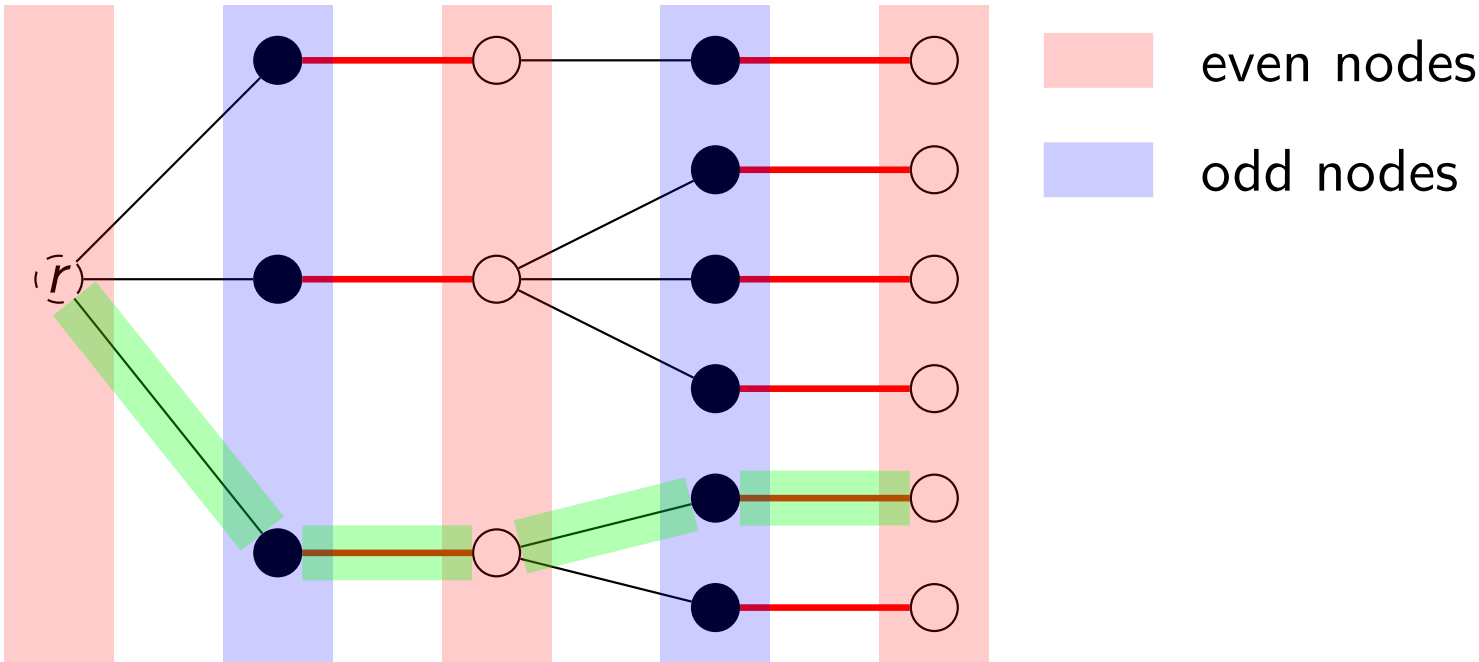
Review

The frontier of the BFS consist of only even nodes, from where we extend the search tree.



Review

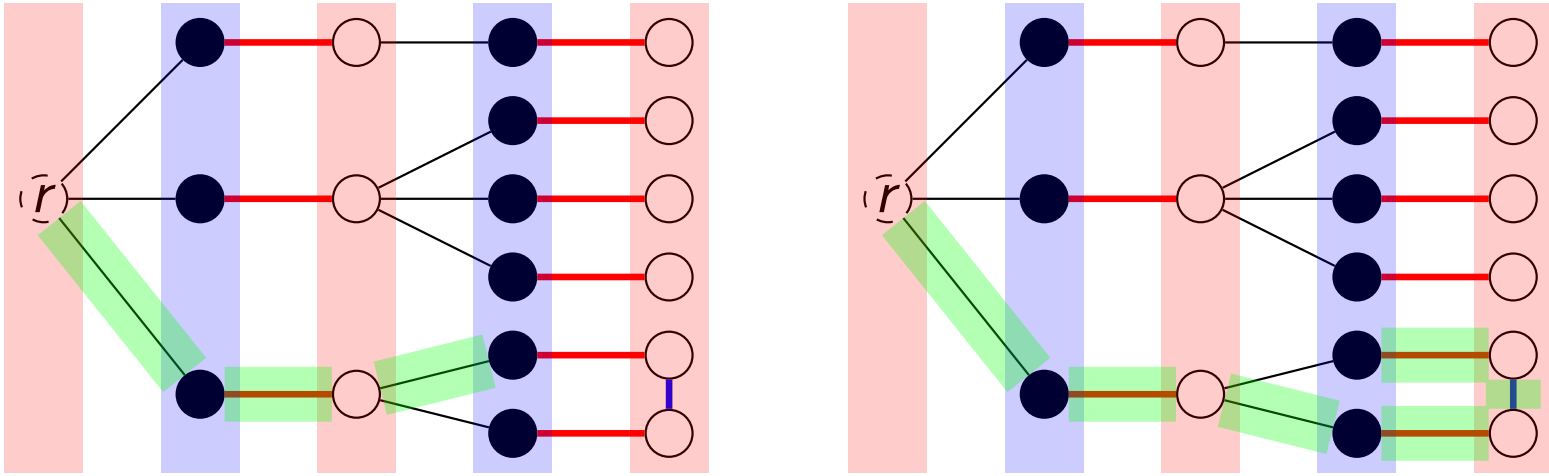
For any node in the tree, there is a unique path from root to the node.
For odd node, the path length is odd. For even node, the path lenght is even.



Review

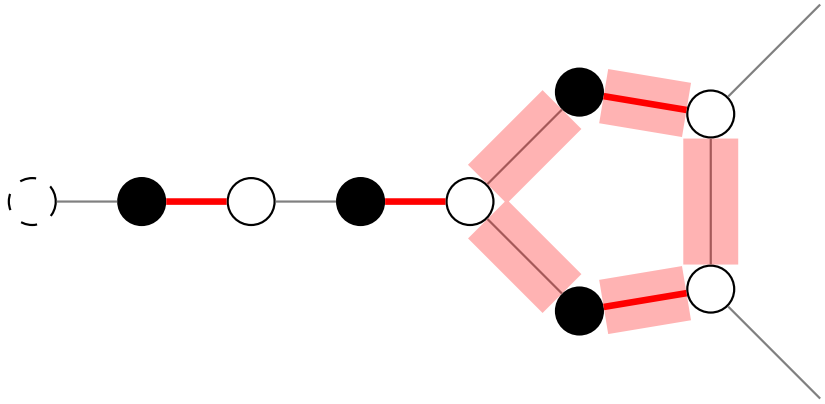
If there is an edge connecting two even nodes (equivalently there is an odd cycle), there are now two paths from root to every node in the odd cycle, one is odd length, the other is even length.

Then the previous odd nodes can also become even nodes, from which we can continue growing the tree!

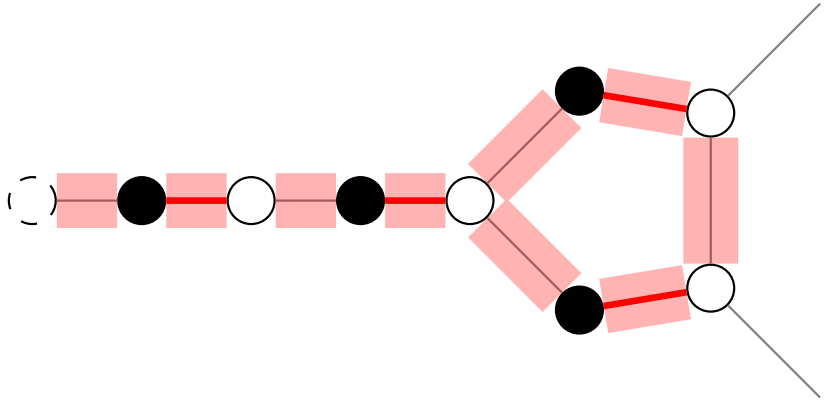


Edmonds' idea

He called the odd cycle blossom.

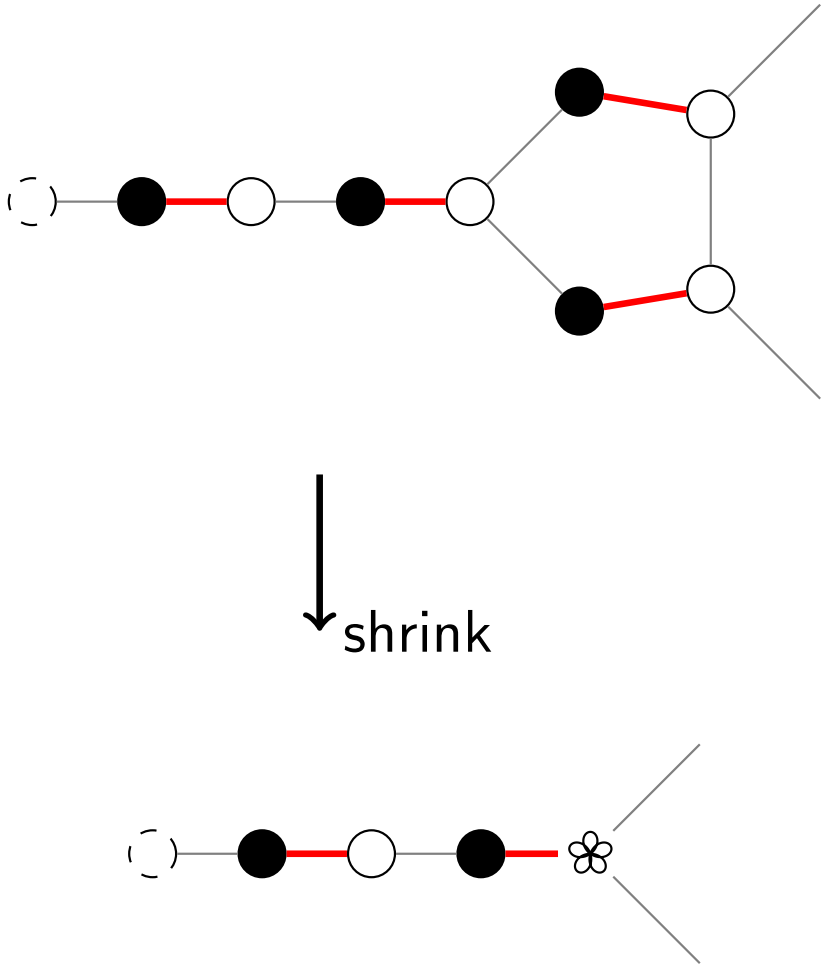


The alternating path from root plus the blossom is called flower.



Edmonds' idea

Since all the nodes in the odd cycle can become even nodes, we shrink all the nodes to an even pseudo vertex, all edges connected to the nodes in the odd cycle now connect to the pseudo vertex.



Edmonds' idea

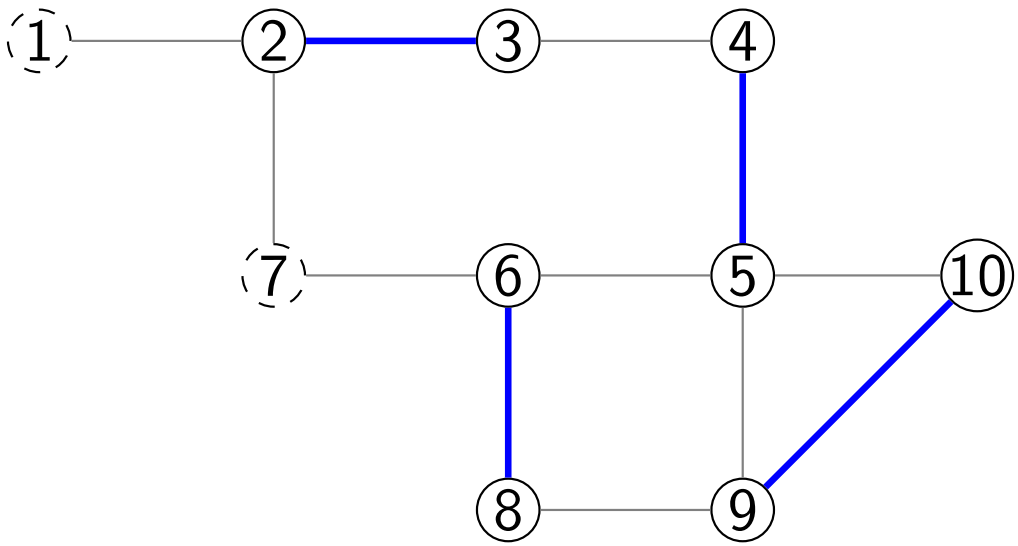
- Whenever a blossom B is found, we shrink it to a pseudo vertex, and recursively run the algorithm to find maximum matching of the new graph G/B.
- Edmonds proved that the maximum matching of G is equal to the maximum matching of G/B plus the maximum matching of B.
- Since there are at most n exposed vertices, we need $O(n)$ augmentations.
- Each augmentation will shrink at most $O(n)$ blossoms.
- Constructing the alternating tree takes at most n^2 if we use adjacency matrix.
- $\therefore O(n^4)$.

Other implementations

- Actually we do not need to contract the blossom. We can also put all the odd nodes in the blossom to the queue of the BFS. But mainting the alternating path to every vertex still needs some trick (We need to change the odd path to the odd node with the other even length path).
- With some careful implementaion, Gabow[73] reduced the time complexity to $O(n^3)$.
- Currently the best time complexity is $O(n^{2.5})$ by Micali and Vazirani[80].

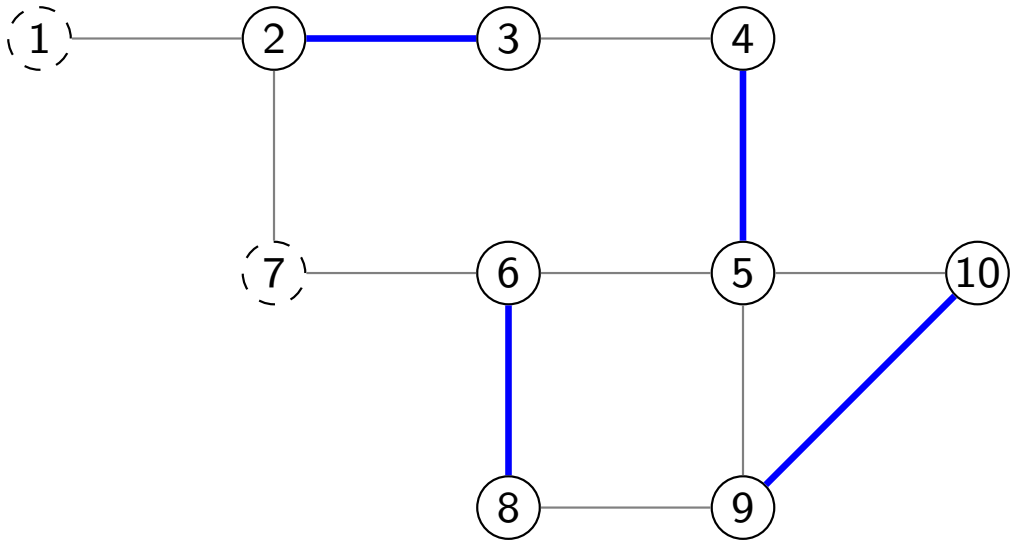
Examples

A graph $G = (V, E)$ and a matching M.



Examples

A graph $G = (V, E)$ and a matching M.

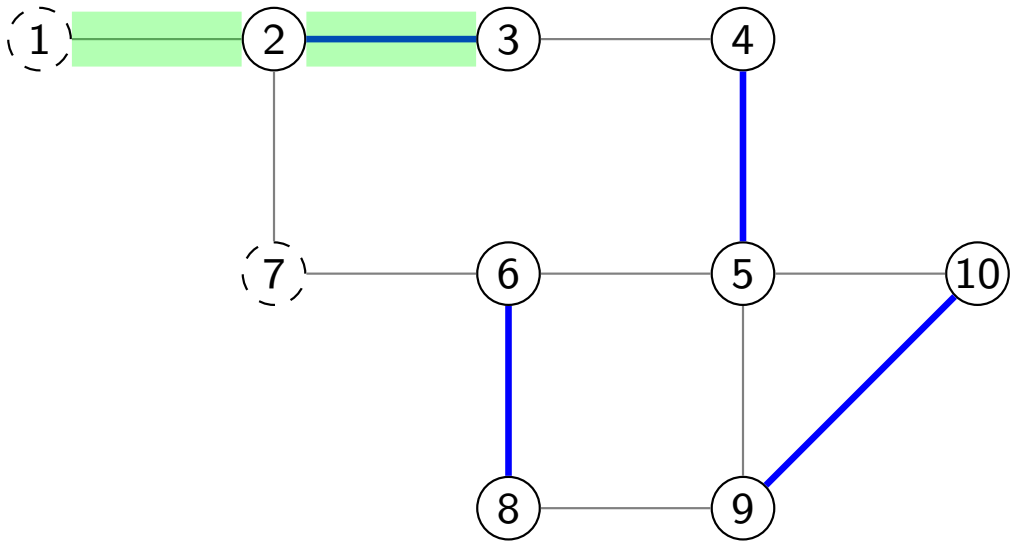


Start a BFS from 1.

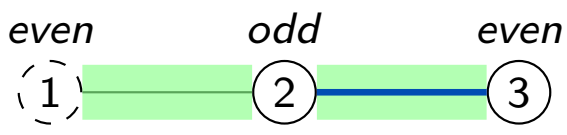
even

Examples

A graph $G = (V, E)$ and a matching M.

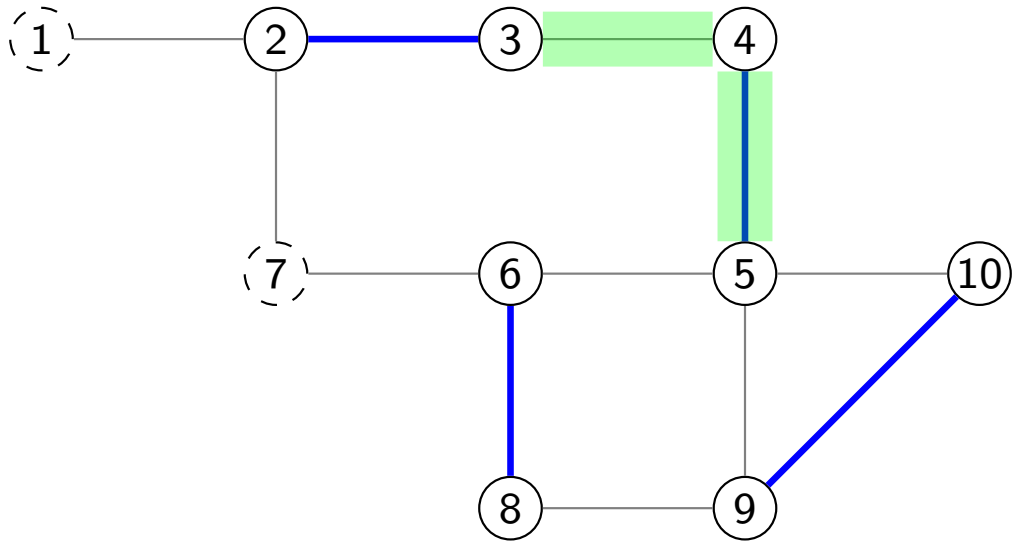


Grow the tree.

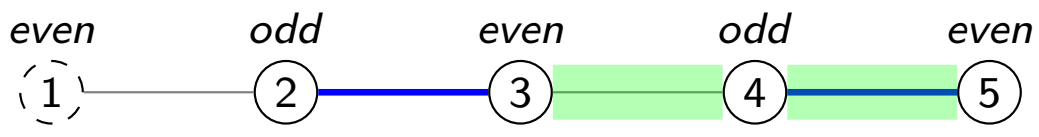


Examples

A graph $G = (V, E)$ and a matching M.

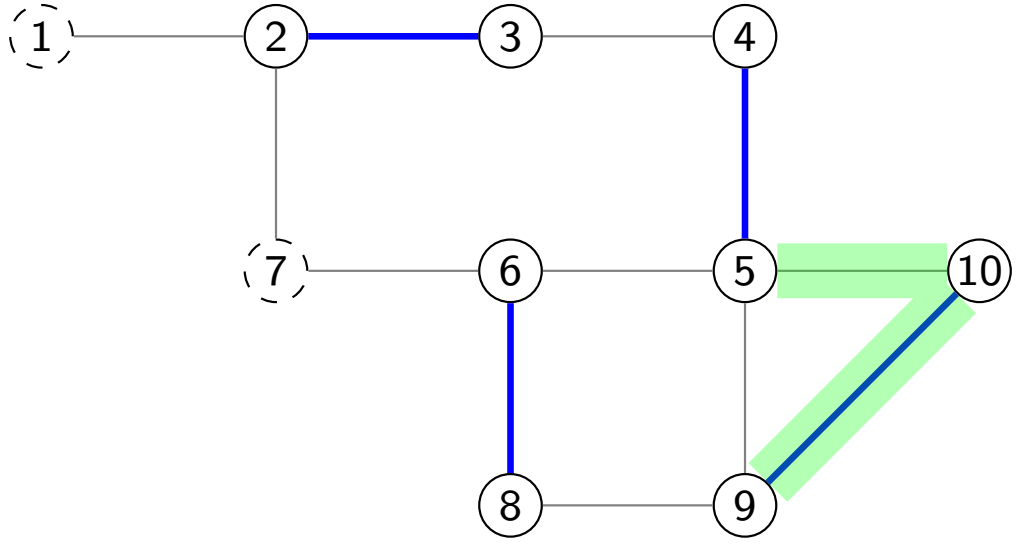


Grow the tree..

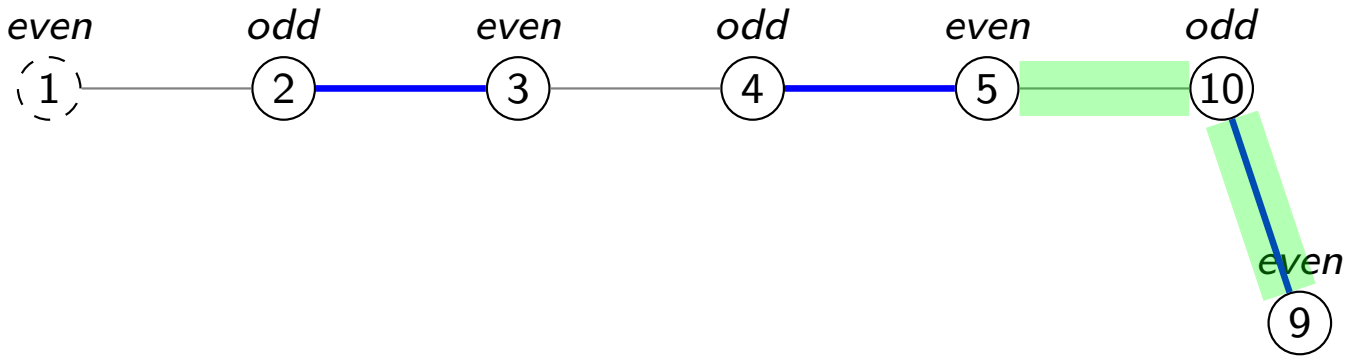


Examples

A graph $G = (V, E)$ and a matching M.

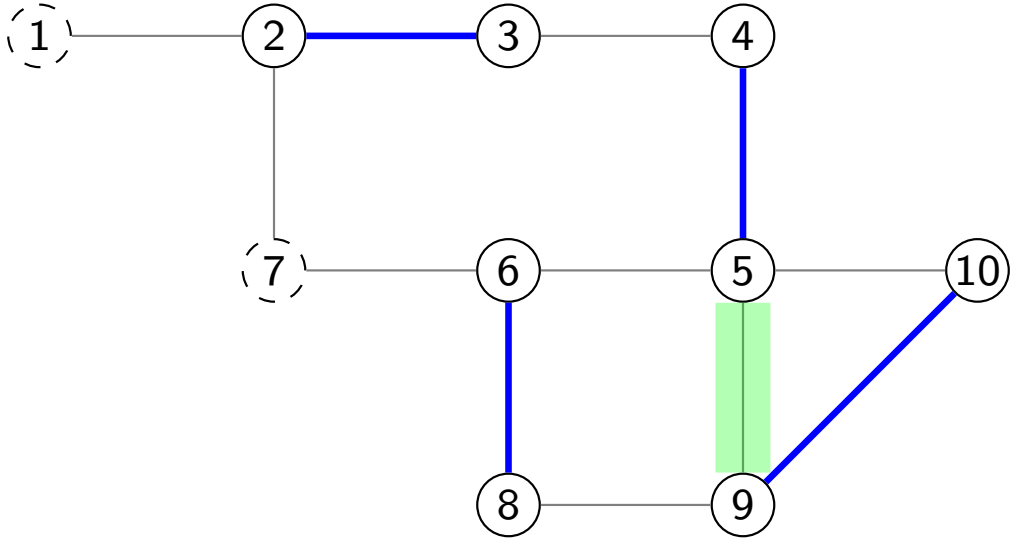


Grow the tree....

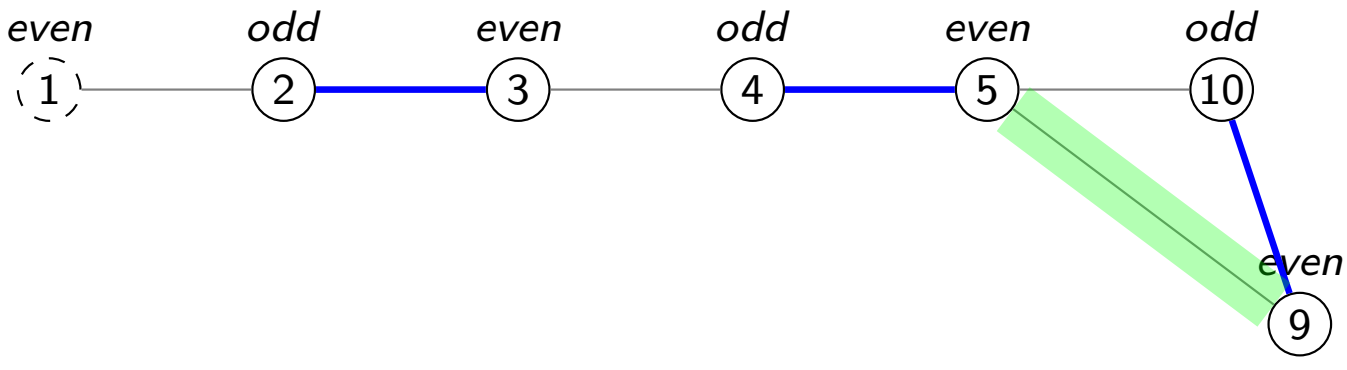


Examples

A graph $G = (V, E)$ and a matching M.

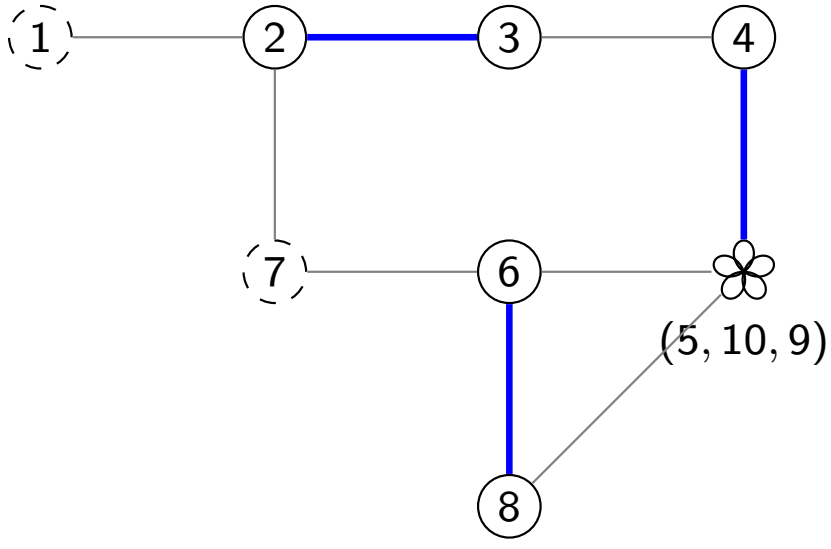


Encounter an edge connecting two even vertices. Find an odd cycle!

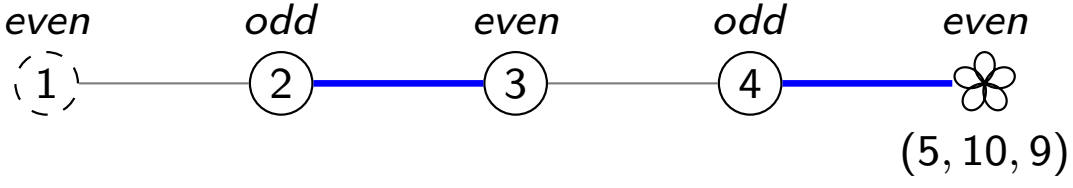


Examples

A graph $G = (V, E)$ and a matching M.

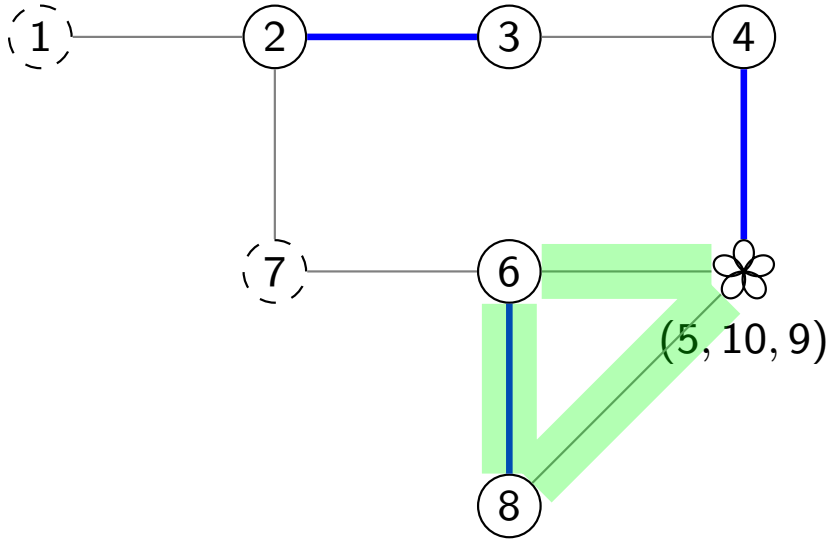


Shrink (5,10,9) into a single pseudo vertex.

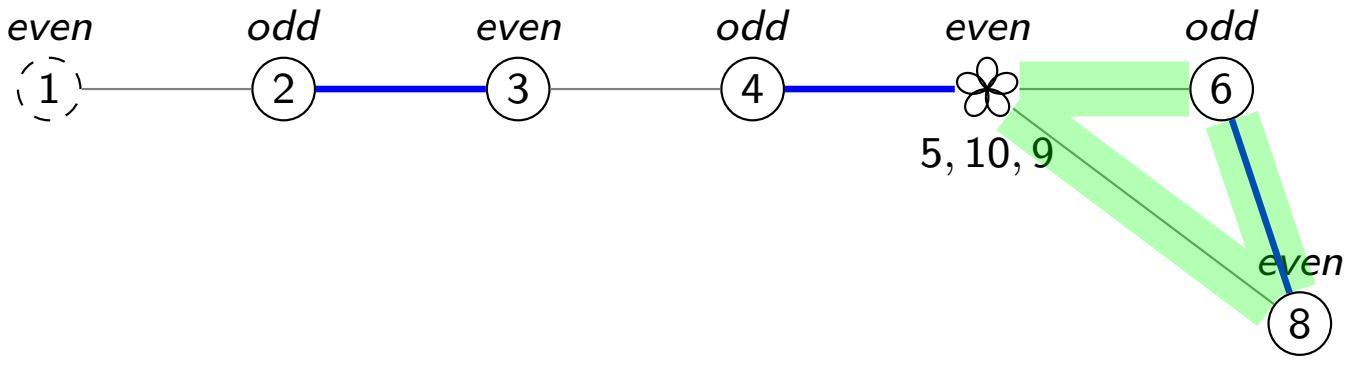


Examples

A graph $G = (V, E)$ and a matching M.

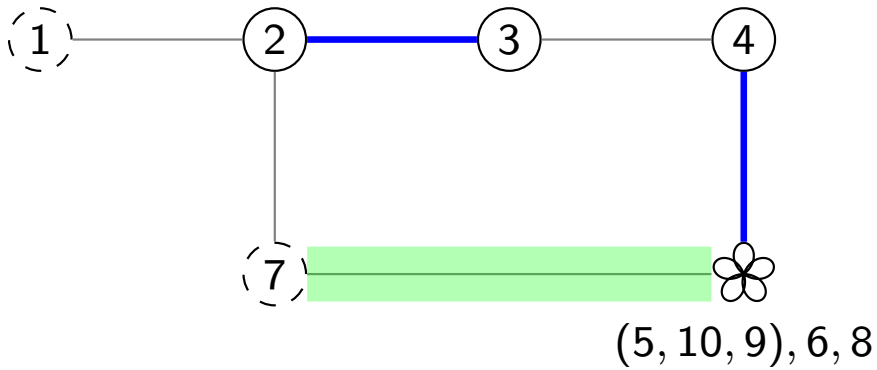


Continue the search, and encounter the odd cycle (5,10,9)-6-8.

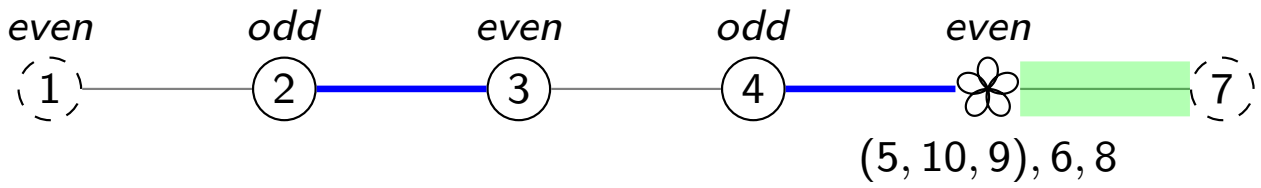


Examples

A graph $G = (V, E)$ and a matching M.

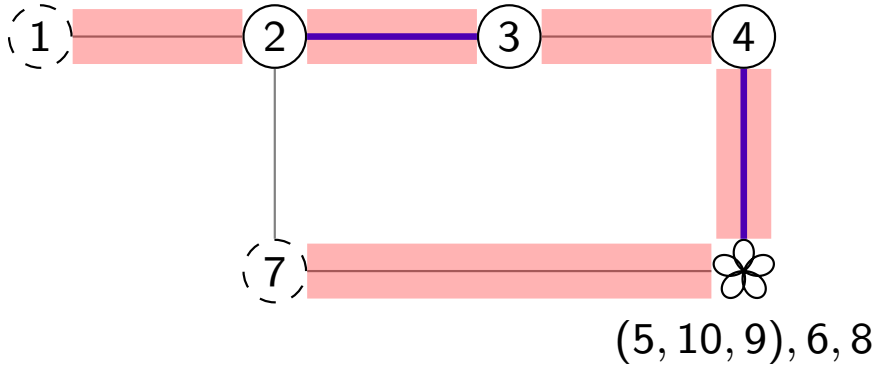


Shrinke the odd cycle and encounter an exposed vertex 7.

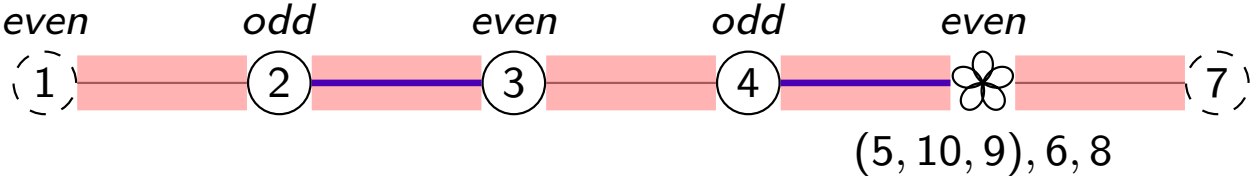


Examples

A graph $G = (V, E)$ and a matching M.

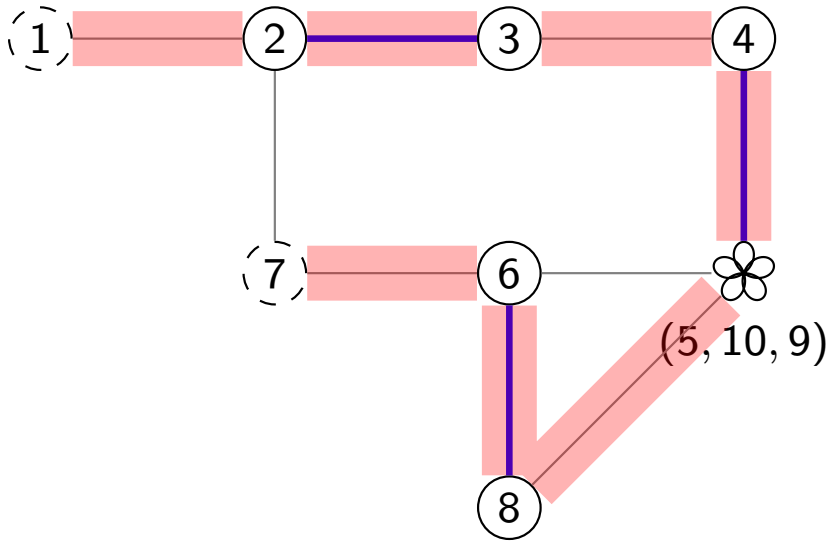


Find an augmenting path! Still need to find the augmenting path in the original graph...

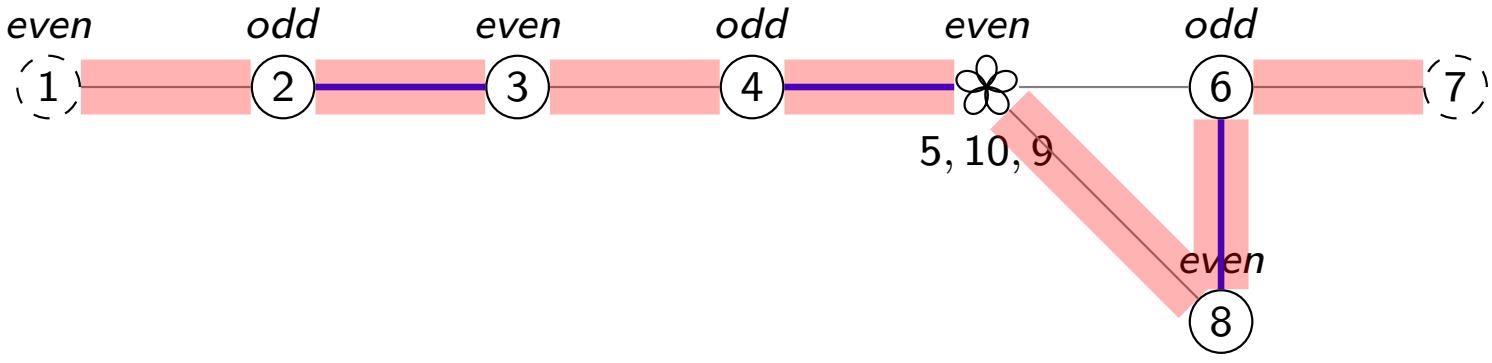


Examples

A graph $G = (V, E)$ and a matching M.

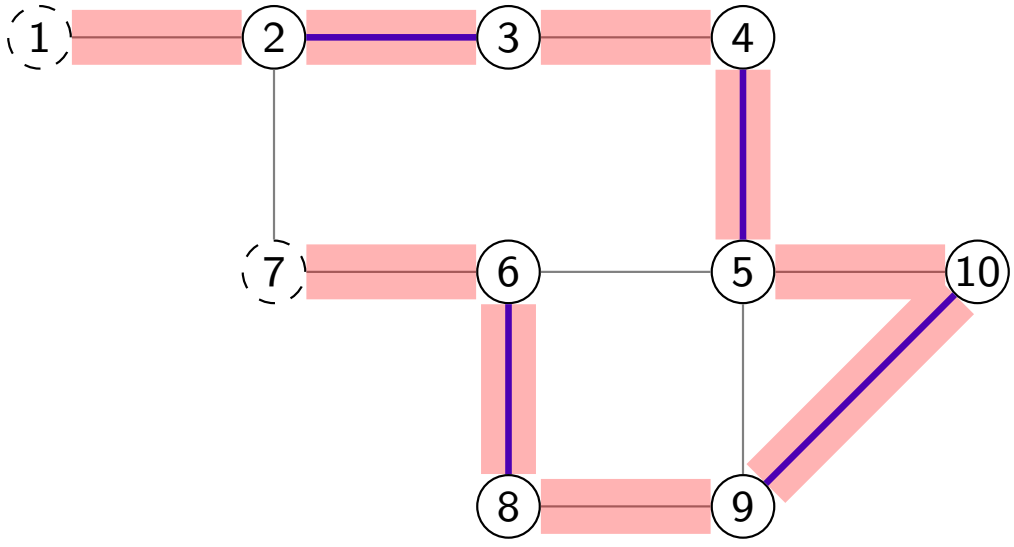


Expand the pseudo vertex to recover the augmenting path.
Since there are two ways to bypass the odd cycle, one is odd length, the other is even length, we can always choose one to ensure the path is still an augmenting path.

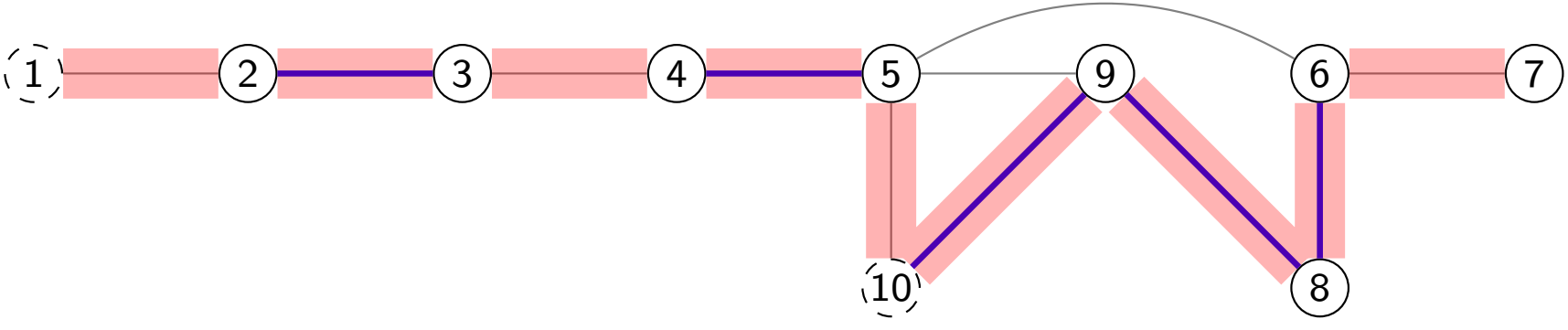


Examples

A graph $G = (V, E)$ and a matching M.

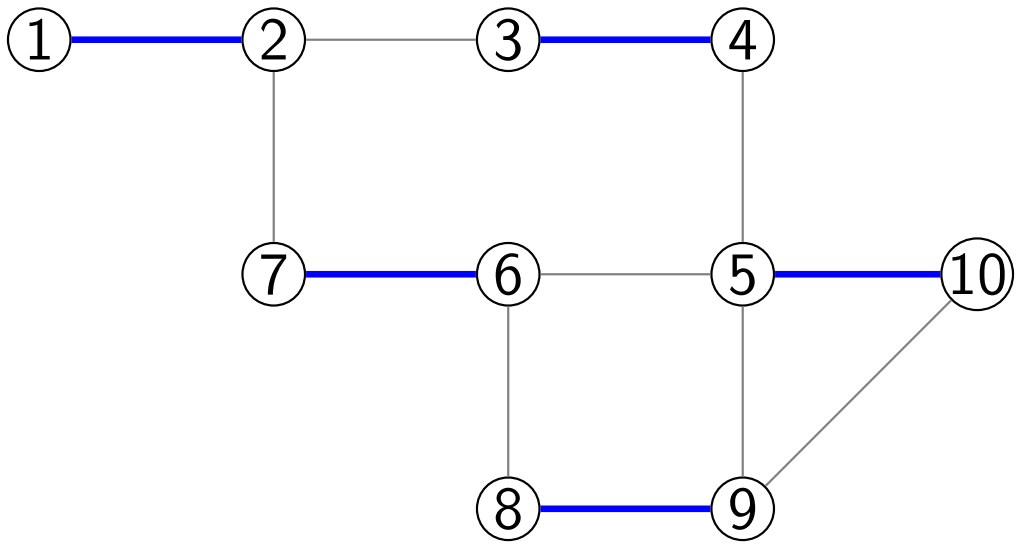


Expand the pseudo vertex to recover the augmenting path.



Examples

Applying $M = M \oplus P$ yeilds following enlarged matching in the original graph.



Summary

- An efficient algorithm has been found.
- The essence is to extend the search tree with odd nodes in the blossom. The other parts are the same with the algorithm on bipartite graph.

Matching-duality theorem

The maximum matching problem can also be solved as linear programming problem.

But, what is linear programming?

Linear programming

Standard linear programming model

maximize

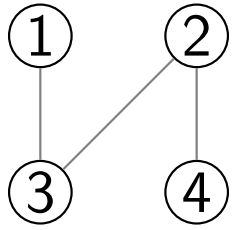
$$c^T x$$

subject to

$$Ax \leq 0$$

$$x \geq 0$$

Maximum matching as linear programming



Maximum matching

maximize

$$x_{13} + x_{23} + x_{24}$$

subject to

$$\begin{aligned}
x_{13} &\leq 1 \\
x_{23} + x_{24} &\leq 1 \\
x_{13} + x_{23} &\leq 1 \\
x_{24} &\leq 1
\end{aligned}$$

Matching-duality theorem

Linear programming duality theorem

If

$$x \geq 0, Ax \leq c$$
$$y \geq 0, A^T y \geq b$$

for given real vectors b and c and real matrix A, then for real vectors x and y,

$$max_x(b, x) = min_y(c, y)$$

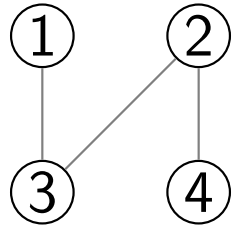
if such extrema exist.

Generally, if we can solve one, we can solve the other.

Matching-duality theorem

And the dual problem of max matching is...?

Matching-duality theorem on bipartite graph



Maximum matching

maximize

$$x_{13} + x_{23} + x_{24}$$

subject to

$$\begin{array}{rcl} x_{13} & \leq & 1 \\ x_{23} + x_{24} & \leq & 1 \\ x_{13} + x_{23} & \leq & 1 \\ x_{24} & \leq & 1 \end{array}$$

Minimum set cover

minimize

$$x_1 + x_2 + x_3 + x_4$$

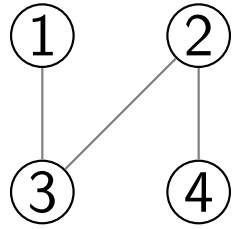
subject to

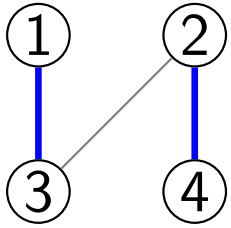
$$\begin{array}{rcl} x_1 + x_3 & \geq & 1 \\ x_2 + x_3 & \geq & 1 \\ x_2 + x_4 & \geq & 1 \end{array}$$

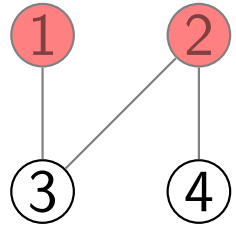
Matching-duality theorem on bipartite graph

König theorem

In a bipartite graph, the maximum size of a matching is equal to the minimum size of a vertex cover.







Matching-duality theorem

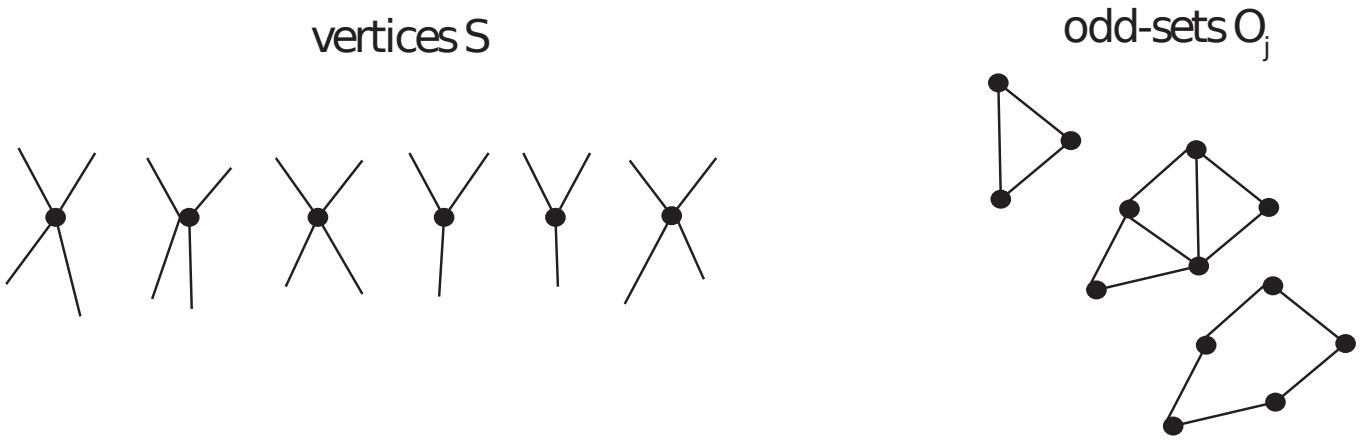
Can we extend it to general graph?

Matching-duality theorem

Odd set cover An odd set contains odd number of vertices. A edge is covered by a single vertex set S if it is adjacent to the vertex. If the size of the odd set is larger than 1, both end points of the edge must be in the set O.

The capacity of the odd-set cover is defined as: $|S| + \sum_{j=1}^{t} \frac{|O_j|-1}{2}$.

Note $\frac{|O|-1}{2}$ is the possible size of the maximum matching of the odd set O.



Matching-duality theorem

General König theorem

If M is a maximum matching and C is a minimum odd-set cover then

$$|M| = \mathit{capacity}(C)$$

- Proof:

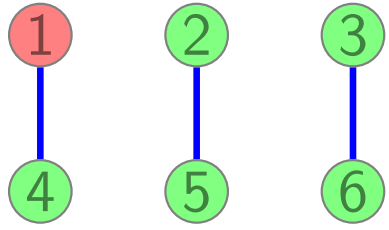
  Edges of M are covered by either S or O_j. Each vertex of S can cover at most one edge in the matching, and O_j can cover at most $\frac{|O|-1}{2}$ edges and leave one vertex exposed. Therefore, $|M| \leq \mathit{capacity}(C)$.

Matching-duality theorem

- Proof:

  Then we have only to prove the existence of an odd set cover and a matching for which the numbers are equal.

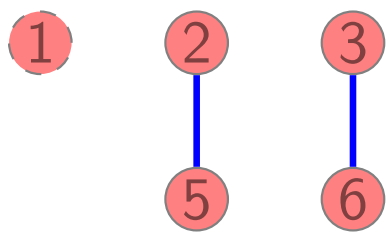
  For a pertect matching M with no exposed vertices, the odd set cover consists of two sets. One is a single vertex, one consists of other vertices.



Matching-duality theorem

- Proof:

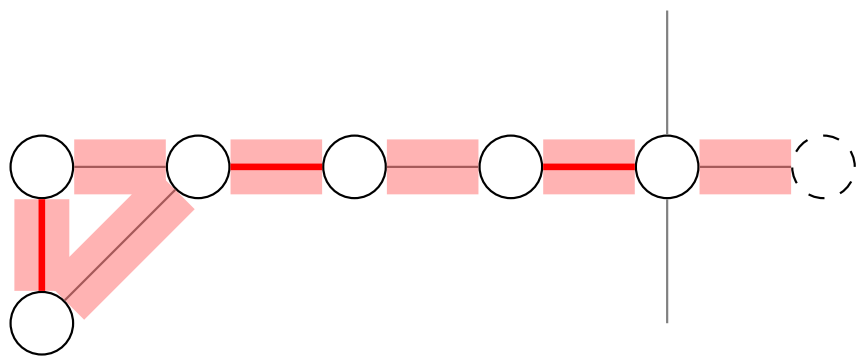
  For a graph which has a matching with one exposed vertices, the odd set cover consists of one set, that is the set of all the vertices.



Matching-duality theorem

- Proof:

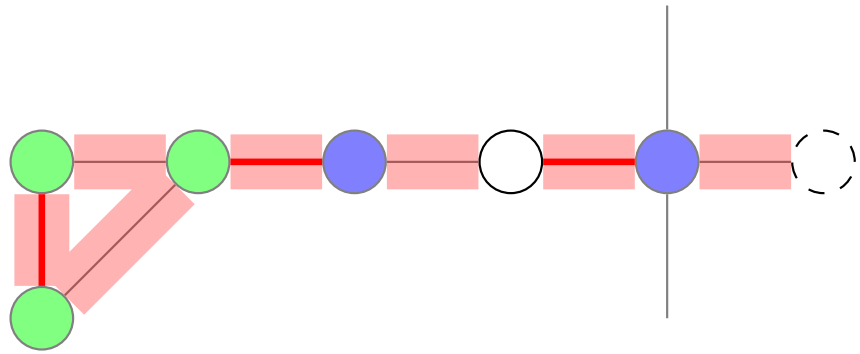
  For a graph with more than one exposed vertex, if we run the blossom algorithm from an exposed vertex, we get an alternating tree.



Matching-duality theorem

- Proof:

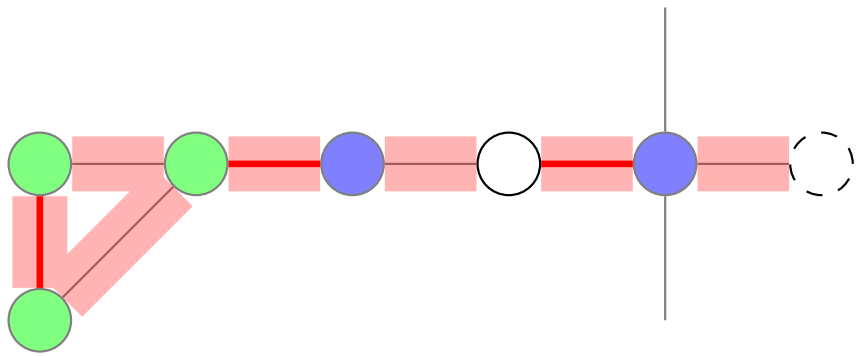
  The odd set cover for the alternating tree J consists of the sets of blossoms, and all the odd nodes as single vertex sets. And its capacity is equal to the size of the matching on J.



Matching-duality theorem

- Proof:

  For the other components of the graph $G - J$, the number of exposed vertex is reduced by one. If we assume $|M_{G-J}| = \mathit{capacity}(C_{G-J})$, we can prove the theorem by induction.



Thank you! && Questions?