

Combinatorial Optimization Course

Bipartite Matching Problem

Mohammad Hossein Bateni

February 9, 2005

Topics

- Maximum Matching

Topics

- Maximum Matching
 - ★ Direct Algorithm

Topics

- Maximum Matching
 - ★ Direct Algorithm
 - ★ Maximum Flow Application

Topics

- Maximum Matching
 - ★ Direct Algorithm
 - ★ Maximum Flow Application
 - ★ Vertex Cover, Edge Cover, Independent Set

Topics

- Maximum Matching
 - ★ Direct Algorithm
 - ★ Maximum Flow Application
 - ★ Vertex Cover, Edge Cover, Independent Set
- Maximum Weight Perfect Matching
 - ★ Algorithm
 - ★ Weighted Vertex Cover

Definitions

- Bipartite Graph

Definitions

- Bipartite Graph
- Matching

Definitions

- Bipartite Graph
- Matching
- Maximum Matching

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge
- Perfect Matching

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge
- Perfect Matching
- Maximum Weight Perfect Matching, ...

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge
- Perfect Matching
- Maximum Weight Perfect Matching, ...
- alternating path,

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge
- Perfect Matching
- Maximum Weight Perfect Matching, ...
- alternating path, augmenting path

Definitions

- Bipartite Graph
- Matching
- Maximum Matching *Maximum \neq Maximal*
- Matched Vertex and Matched Edge
- Perfect Matching
- Maximum Weight Perfect Matching, ...
- alternating path, augmenting path and symmetric difference

Applications

- **Job Assignment Problem**, we have a set of n jobs and n workers where each worker can do some of jobs. The objective is to accomplish the more jobs while no worker is assigned more than one job.

Applications

- **Job Assignment Problem**, we have a set of n jobs and n workers where each worker can do some of jobs. The objective is to accomplish the more jobs while no worker is assigned more than one job.
- **Factories Problem**, there are n farms and n processing plants. Sending the output of farm i to plant j has a revenue of w_{ij} . We want to maximize the profit.

Applications

- **Job Assignment Problem**, we have a set of n jobs and n workers where each worker can do some of jobs. The objective is to accomplish the more jobs while no worker is assigned more than one job.
- **Factories Problem**, there are n farms and n processing plants. Sending the output of farm i to plant j has a revenue of w_{ij} . We want to maximize the profit.
- **Government Problem** *We will see later*

Applications

- **Job Assignment Problem**, we have a set of n jobs and n workers where each worker can do some of jobs. The objective is to accomplish the more jobs while no worker is assigned more than one job.
- **Factories Problem**, there are n farms and n processing plants. Sending the output of farm i to plant j has a revenue of w_{ij} . We want to maximize the profit.
- Government Problem *We will see later*
- Map Labeling Problems
- Scheduling Problems

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation!

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation! For the other way,

- Consider a maximum Matching M'

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation! For the other way,

- Consider a maximum Matching M'
- and $H = M \Delta M'$

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation! For the other way,

- Consider a maximum Matching M'
- and $H = M \Delta M'$
- No Paths of odd length in H

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation! For the other way,

- Consider a maximum Matching M'
- and $H = M \Delta M'$
- No Paths of odd length in H
- A collection of Even Cycles and Even Paths

A Framework

THEOREM 1 (Berge) *A Matching M in a graph G is a maximum matching iff there is no augmenting path in G with respect to M .*

Proof. If there exists an augmenting path, so we get a better matching by augmentation! For the other way,

- Consider a maximum Matching M'
- and $H = M \Delta M'$
- No Paths of odd length in H
- A collection of Even Cycles and Even Paths
- Counting the Edges, we have $|M| = |M'|$

Now, we have the theorem proven. \square

A Framework (Cont'd)

- **General Framework.** Start with a matching $M = \emptyset$ and iteratively find augmenting paths, until no such path is found.

A Framework (Cont'd)

- **General Framework.** Start with a matching $M = \emptyset$ and iteratively find augmenting paths, until no such path is found.
- How to search?

A Framework (Cont'd)

- **General Framework.** Start with a matching $M = \emptyset$ and iteratively find augmenting paths, until no such path is found.
- How to search?
- We can grow alternating paths to find augmenting paths

A Framework (Cont'd)

- **General Framework.** Start with a matching $M = \emptyset$ and iteratively find augmenting paths, until no such path is found.
- How to search?
- We can grow alternating paths to find augmenting paths
- It is enough to start from free vertices of one part

Towards an Algorithm

- In Odd levels, we either find an augmenting path or we choose the current mate of the vertex

Towards an Algorithm

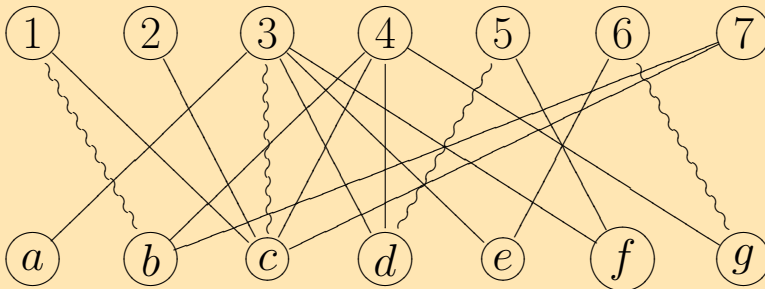
- In Odd levels, we either find an augmenting path or we choose the current mate of the vertex
- an **Outer Vertex** is a vertex in the same partition as start, i.e. even level

Towards an Algorithm

- In Odd levels, we either find an augmenting path or we choose the current mate of the vertex
- an **Outer Vertex** is a vertex in the same partition as start, i.e. even level
- We omit the odd levels and build an auxiliary digraph, in which we have edge (i, j) if we j can be next outer vertex after i

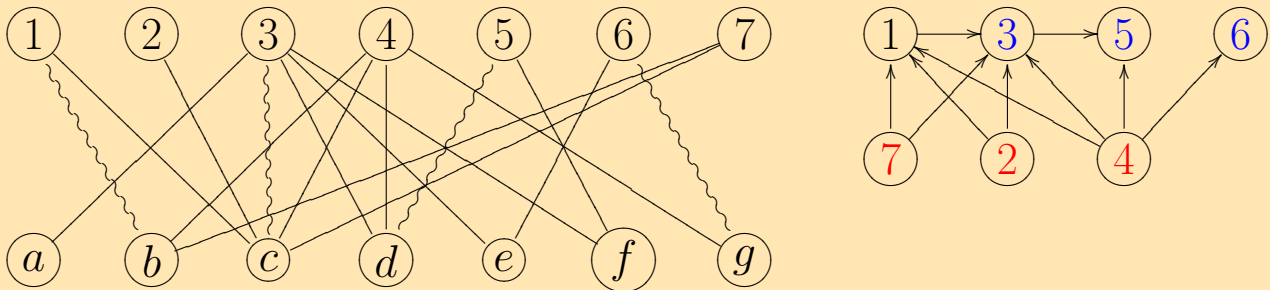
Towards an Algorithm

- In Odd levels, we either find an augmenting path or we choose the current mate of the vertex
- an **Outer Vertex** is a vertex in the same partition as start, i.e. even level
- We omit the odd levels and build an auxiliary digraph, in which we have edge (i, j) if we j can be next outer vertex after i
- And we use BFS to find a path between free vertices and a vertex with free neighbors



Towards an Algorithm

- In Odd levels, we either find an augmenting path or we choose the current mate of the vertex
- an **Outer Vertex** is a vertex in the same partition as start, i.e. even level
- We omit the odd levels and build an auxiliary digraph, in which we have edge (i, j) if we j can be next outer vertex after i
- And we use BFS to find a path between free vertices and a vertex with free neighbors



An Algorithm

1. $M = \emptyset$
2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment

An Algorithm

1. $M = \emptyset$
 2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment
- Step 1 runs at most $\frac{|M|}{2}$ times

An Algorithm

1. $M = \emptyset$
 2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment
- Step 1 runs at most $\frac{|M|}{2}$ times
 - Step 2(a) takes $O(V + E)$

An Algorithm

1. $M = \emptyset$
 2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment
- Step 1 runs at most $\frac{|M|}{2}$ times
 - Step 2(a) takes $O(V + E)$
 - Step 2(b) takes $O(V + E)$

An Algorithm

1. $M = \emptyset$
2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment

- Step 1 runs at most $\frac{|M|}{2}$ times
- Step 2(a) takes $O(V + E)$
- Step 2(b) takes $O(V + E)$
- Step 2(c) takes $O(V + E)$

An Algorithm

1. $M = \emptyset$
2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment

- Step 1 runs at most $\frac{|M|}{2}$ times
- Step 2(a) takes $O(V + E)$
- Step 2(b) takes $O(V + E)$
- Step 2(c) takes $O(V + E)$
- The algorithm runs in time $O(|V| \times |E|)$

An Algorithm

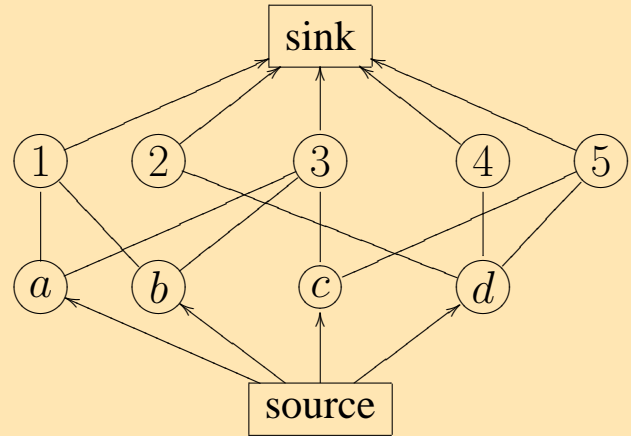
1. $M = \emptyset$
 2. **while** M is not maximum **do**
 - (a) Build Auxiliary Graph
 - (b) BFS
 - (c) **if** no path **then** M is maximum **else** augment
- Step 1 runs at most $\frac{|M|}{2}$ times
 - Step 2(a) takes $O(V + E)$
 - Step 2(b) takes $O(V + E)$
 - Step 2(c) takes $O(V + E)$
 - The algorithm runs in time $O(|V| \times |E|)$
 - Correctness due to **Berge's Theorem** and construction of the Auxiliary Graph

Network Flow Modeling

- How?

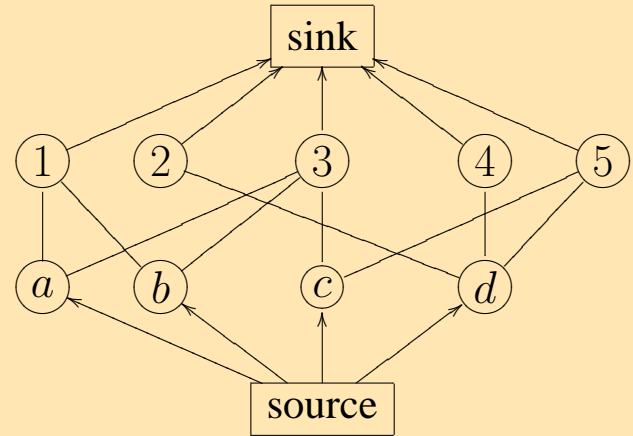
Network Flow Modeling

- How?



Network Flow Modeling

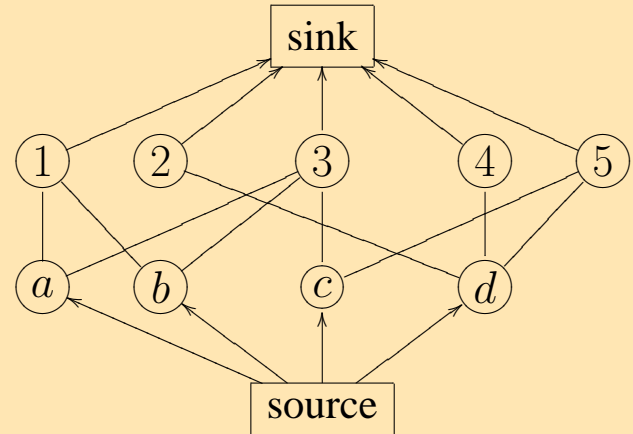
- How?



- Proof

Network Flow Modeling

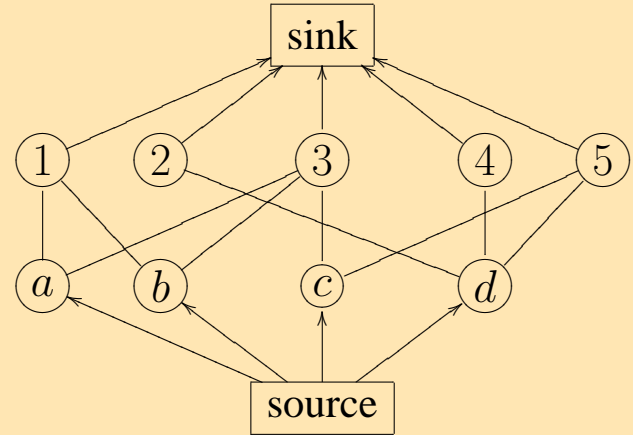
- How?



- Proof
- Why? We have solved the problem? Why?

Network Flow Modeling

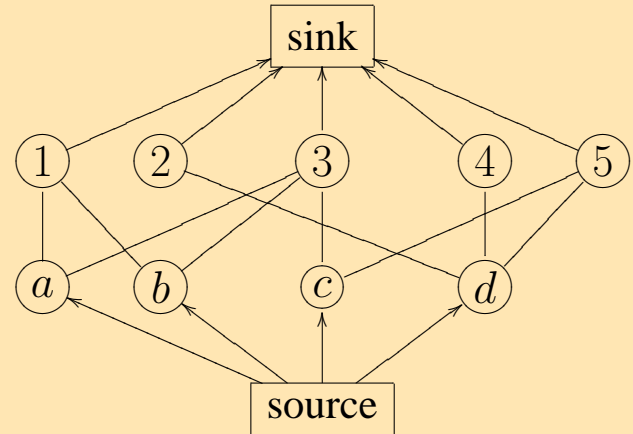
- How?



- Proof
- Why? We have solved the problem? Why?
 - ★ We can use any network flow algorithm to improve the efficiency of Matching Algorithm

Network Flow Modeling

- How?



- Proof
- Why? We have solved the problem? Why?
 - ★ We can use any network flow algorithm to improve the efficiency of Matching Algorithm
 - ★ We might understand some aspects of the problem through the similarity to a well-known problem

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm
- The algorithm runs in time $O(\sqrt{|V|} \cdot |E|)$ for simple networks

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm
- The algorithm runs in time $O(\sqrt{|V|} \cdot |E|)$ for simple networks
- So we have a faster algorithm

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm
- The algorithm runs in time $O(\sqrt{|V|} \cdot |E|)$ for simple networks
- So we have a faster algorithm
- Karp found a direct algorithm with this time bound

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm
- The algorithm runs in time $O(\sqrt{|V|} \cdot |E|)$ for simple networks
- So we have a faster algorithm
- Karp found a direct algorithm with this time bound
- He proves that, if we find a maximal set of disjoint shortest augmenting paths and augment them all; we will only need $O(\sqrt{|V|})$ iterations

A Better Algorithm

- A network in which all capacities are zero/one and every vertex has indegree or outdegree zero/one, is called **A Simple Network**
- There exists a network flow algorithm called Push-Relabel Algorithm
- The algorithm runs in time $O(\sqrt{|V|} \cdot |E|)$ for simple networks
- So we have a faster algorithm
- Karp found a direct algorithm with this time bound
- He proves that, if we find a maximal set of disjoint shortest augmenting paths and augment them all; we will only need $O(\sqrt{|V|})$ iterations
- And he gives an $O(V + E)$ algorithm to do so

Optimality

- How can we prove a matching to be optimal?

Optimality

- How can we prove a matching to be optimal?
- A proof should be small

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice
- A **Vertex Cover** is a subset of vertices which is incident to all edges

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice
- A **Vertex Cover** is a subset of vertices which is incident to all edges
- Minimum Vertex Cover Problem

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice
- A **Vertex Cover** is a subset of vertices which is incident to all edges
- Minimum Vertex Cover Problem
- The size of a vertex cover is at least the size of maximum matching

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice
- A **Vertex Cover** is a subset of vertices which is incident to all edges
- Minimum Vertex Cover Problem
- The size of a vertex cover is at least the size of maximum matching
- Having a matching and vertex cover of the same size proves that both are optimal

Optimality

- How can we prove a matching to be optimal?
- A proof should be small
- Dual optimization problem is a good choice
- A **Vertex Cover** is a subset of vertices which is incident to all edges
- Minimum Vertex Cover Problem
- The size of a vertex cover is at least the size of maximum matching
- Having a matching and vertex cover of the same size proves that both are optimal
- Can we always do so?

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths
- choose $C = \bar{Q} \cup T$ as the vertex cover where $\bar{Q} = X - Q$

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths
- choose $C = \bar{Q} \cup T$ as the vertex cover where $\bar{Q} = X - Q$
- an edge not incident to C has one endpoint in Q and one in T which contradicts definition of T

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths
- choose $C = \bar{Q} \cup T$ as the vertex cover where $\bar{Q} = X - Q$
- an edge not incident to C has one endpoint in Q and one in T which contradicts definition of T
- all the vertices of C are matched

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths
- choose $C = \bar{Q} \cup T$ as the vertex cover where $\bar{Q} = X - Q$
- an edge not incident to C has one endpoint in Q and one in T which contradicts definition of T
- all the vertices of C are matched
- and no edge from M has both endpoints present

Optimality (Cont'd)

THEOREM 2 (König Egerváry) *If G is a bipartite graph, then the size of a maximum matching in G equals the size of the minimum vertex cover in G .*

- we only need to prove that there always exists a vertex cover with the same size as the maximum matching
- Let P be the free vertices of the upper partition, i.e. X
- $Q \in X$ and $T \in Y$ be the vertices reachable from P by alternating paths
- choose $C = \bar{Q} \cup T$ as the vertex cover where $\bar{Q} = X - Q$
- an edge not incident to C has one endpoint in Q and one in T which contradicts definition of T
- all the vertices of C are matched
- and no edge from M has both endpoints present
- We have a vertex cover of size at most $|M|$

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent
- Complements of a vertex cover is an independent set and vice versa

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent
- Complements of a vertex cover is an independent set and vice versa
- From a Matching M , we can obtain an edge cover of size $|V| - |M|$

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent
- Complements of a vertex cover is an independent set and vice versa
- From a Matching M , we can obtain an edge cover of size $|V| - |M|$
- From an edge cover C , we can find a matching of size $|V| - |C|$ using the edges of C

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent
- Complements of a vertex cover is an independent set and vice versa
- From a Matching M , we can obtain an edge cover of size $|V| - |M|$
- From an edge cover C , we can find a matching of size $|V| - |C|$ using the edges of C

THEOREM 3 *Size of the maximum independent set in a bipartite graph is equal to the size of minimum edge cover if one exists.*

More Duality

- An **Edge Cover** is a subset of edges containing all the vertices
- Only for graphs with no isolated vertices
- An **Independent Set** is a subset of vertices no two of which are adjacent
- Complements of a vertex cover is an independent set and vice versa
- From a Matching M , we can obtain an edge cover of size $|V| - |M|$
- From an edge cover C , we can find a matching of size $|V| - |C|$ using the edges of C

THEOREM 3 *Size of the maximum independent set in a bipartite graph is equal to the size of minimum edge cover if one exists.*

- We have algorithms to solve both problems.

Two Problems

- The Sticks Problem.

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Minimum Edge Cover Problem in a Bipartite Graph

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Minimum Edge Cover Problem in a Bipartite Graph
- The Towers Problem

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Minimum Edge Cover Problem in a Bipartite Graph
- The Towers Problem
 - ★ Build a Graph Problem

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Minimum Edge Cover Problem in a Bipartite Graph
- The Towers Problem
 - ★ Build a Graph Problem
 - ★ The graph is bipartite

Two Problems

- The Sticks Problem.
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Minimum Edge Cover Problem in a Bipartite Graph
- The Towers Problem
 - ★ Build a Graph Problem
 - ★ The graph is bipartite
 - ★ Maximum Independent Problem in a Bipartite Graph

Any Questions?

THE END