A Brief Review of Graph Theory

Lecture 7

Undirected Graphs

- An <u>undirected</u> **graph** $G = (V, E)$ consists of :
  - A set V of **vertices** (or **nodes**).
  - A set E of **edges** (or **links**) denoting *unordered* vertex pairs.

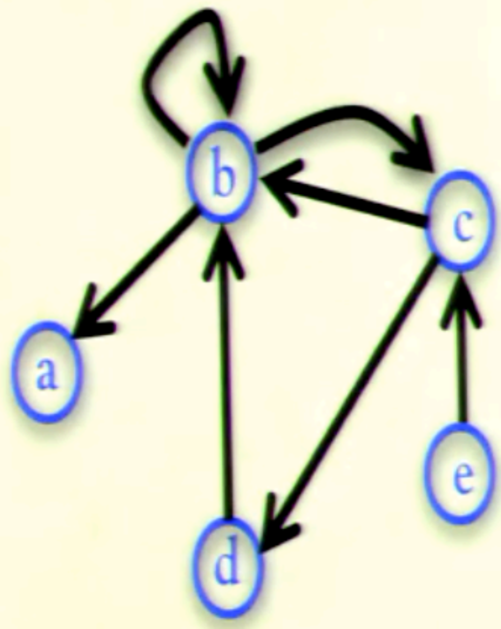
e.g.

- We set $n = |V|$ to be the *cardinality* of the vertex set.
- We set $m = |E|$ to be the *cardinality* of the edge set.

Directed Graphs

- A <u>directed</u> **graph** $G = (V, E)$ consists of:
  - A set V of **vertices.**
  - A set A of **arcs** (directed edges) denoting *ordered* vertex pairs.

e.g.



- We set $n = |V|$ to be the *cardinality* of the vertex set.
- We set $m = |A|$ to be the *cardinality* of the arc set.

Adjacency Matrix [Undirected Graphs]

- For an <u>undirected</u> graph, an **adjaceny matrix** M has the properties that:
  - There is a **row** for each *vertex*.
  - There is **column** for each *vertex*.
  - The *ij*-th **entry** of the matrix is defined by:

$$M_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

e.g.

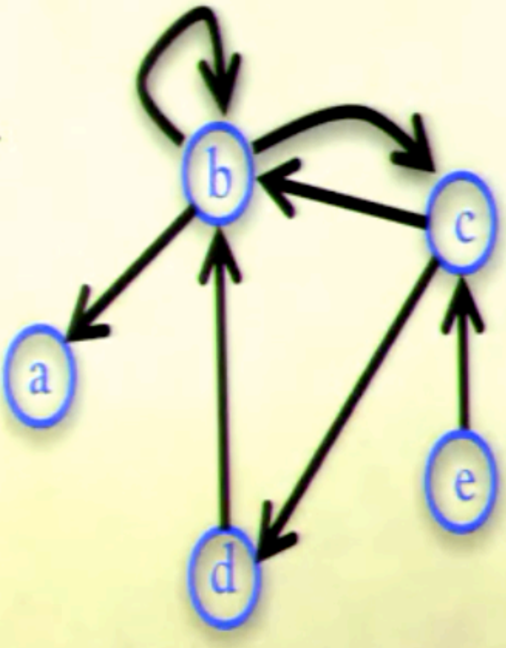
$$M = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Adjacency Matrix [Directed Graphs]

- For a <u>directed</u> graph, an **adjaceny matrix** M has the properties that:
  - There is a **row** for each *vertex*.
  - There is **column** for each *vertex*.
  - The *ij*-th **entry** of the matrix is defined by:

$$M_{ij} = \begin{cases} 1 & (i,j) \in A \\ 0 & (i,j) \notin A \end{cases}$$

e.g.



$$M = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Adjacency Lists [Undirected Graphs]

- An undirected graph can also be stored using **adjaceny lists.**
  - For each vertex *i*, we store a list of the **neighbours** of *i*.

The set of vertices that i shares an edge with.

Adjacency Lists [Undirected Graphs]

- An undirected graph can also be stored using **adjaceny lists.**
  - For each vertex *i*, we store a list of the **neighbours** of *i*.

e.g.

a : *b*

b : *a*, *b*, *c*, *d*

c : *b*, *d*, *e*

d : *b*, *c*

e : *c*

  - Equivalently, we store a list of the edges *incident* to each vertex.

Adjacency Lists [Directed Graphs]

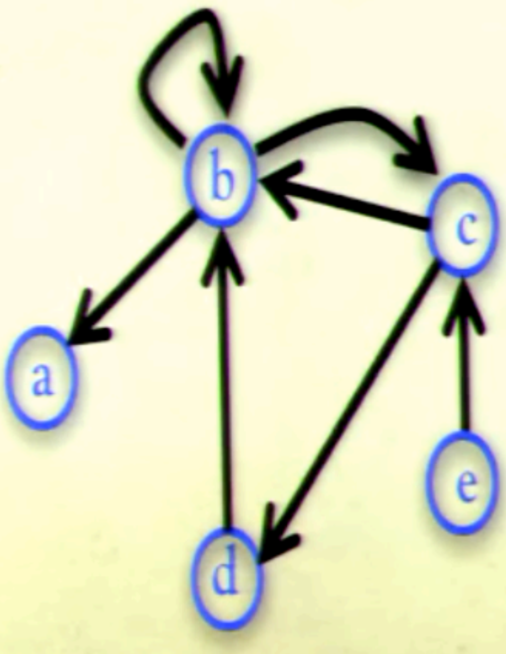
- An directed graph can also be stored using **adjaceny lists**.
  - For each vertex *i*, we store a list of the **in-neighbours** of *i*.
  - For each vertex *i*, we also store a list of the **out-neighbours** of *i*.

The set of vertices that i has arcs pointing to.

Adjacency Lists [Directed Graphs]

- An directed graph can also be stored using **adjaceny lists.**
  - For each vertex *i*, we store a list of the **in-neighbours** of *i*.
  - For each vertex *i*, we also store a list of the **out-neighbours** of *i*.

e.g.



a : b

$-$

b : b, c, d

b, c

c : b, e

b, d

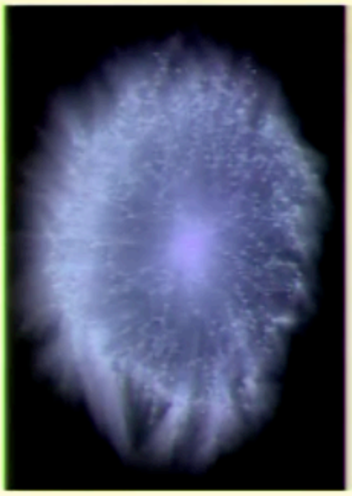
Adjacency Lists versus Adjacency Matrices

- The main difference is in the amount of **storage** required
  - An *adjacency matrix* requires storing $\Theta(n^2)$ numbers.
  - *Adjacency lists* requires storing $\Theta(m)$ numbers.
- In any graph $m = O(n^2)$ and often $m \ll n^2$.

$\Longrightarrow$ In **sparse graphs** it is much more preferable to use adjacency lists.

Graph Applications

- The number of practical applications for is extraordinarily large.
- Obviously there are useful in modelling networks.
  - **Transportation Networks**: e.g. Roads.
  - **Social Networks**: e.g. Facebook Graph.
  - **Information Networks**: e.g. the World Wide Web.
  - **Financial Networks**: e.g. Monetary Flows.

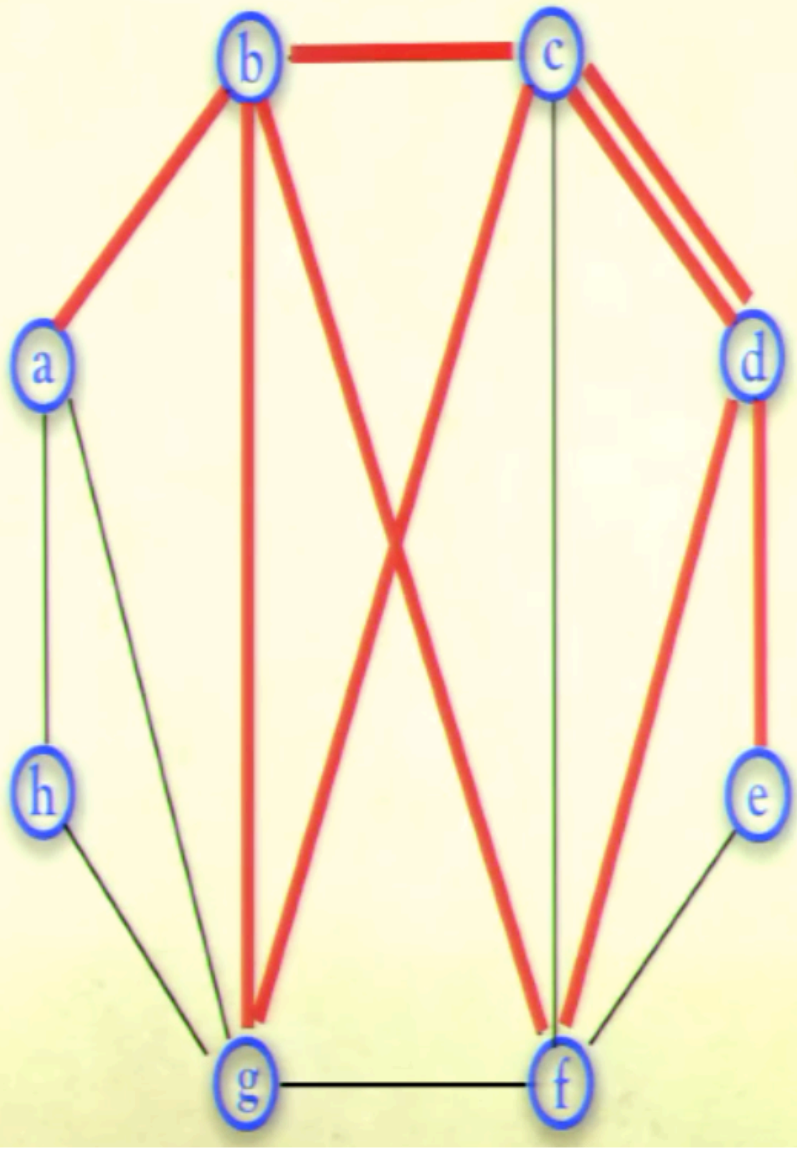


Graph Applications [cont.]

- But, their flexibility allows them to model a plethora of diverse applications:
  - **Hierarchy**: *data structures, linguistics.*
  - **Similarity**: *data clustering, biology.*
  - **Conflict**: *wavelength allocation, scheduling.*
  - **Priority**: *industrial planning, operations research.*
  - **Structure:** *chemistry, physics.*
  - **Time Relations:** *evolution, migration patterns.*

Walks

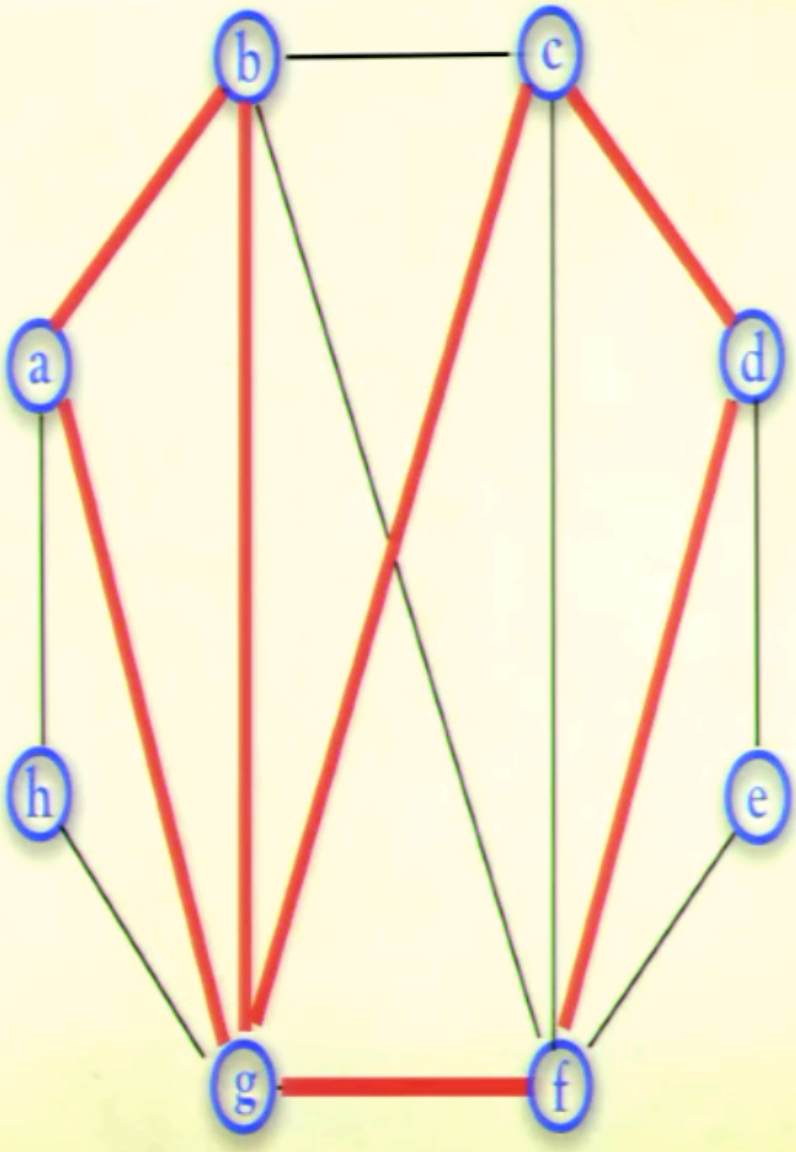
- A **walk** is list of vertices $\{v_0, v_1, v_2, \ldots, v_l\}$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < \ell$.



Circuits

A circuit is a *closed* walk.

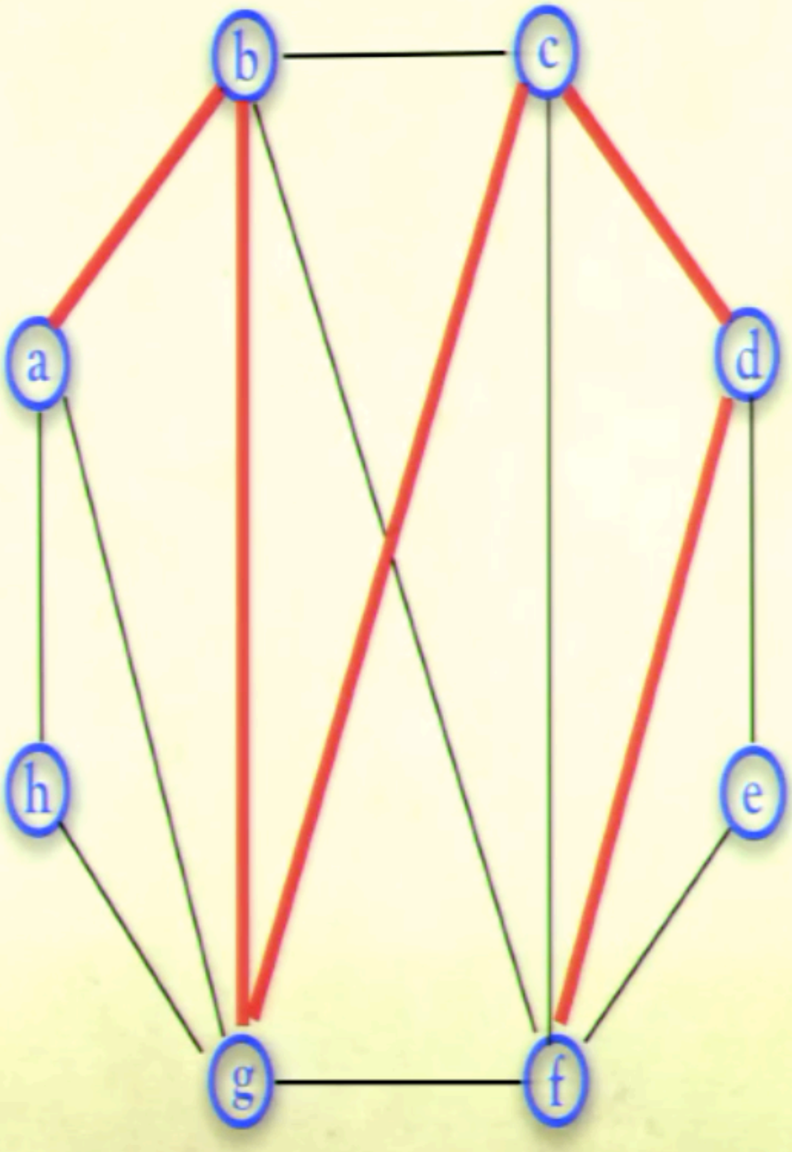
- A **circuit** is a walk $\{v_0, v_1, v_2, \ldots, v_l\}$ where $v_0 = v_\ell$.



- An **Eulerian Circuit** is a circuit that uses every edge *exactly once*.

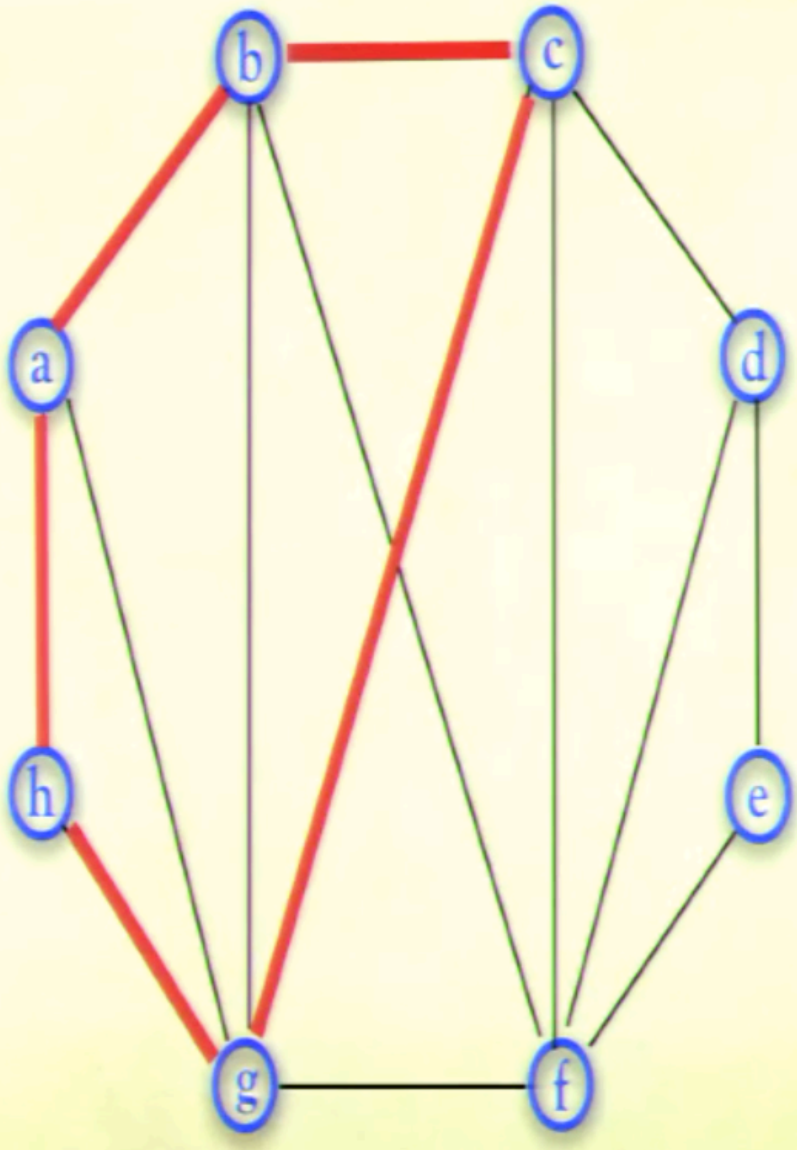
Paths

- A **path** is a walk where every vertex is distinct.



Cycles

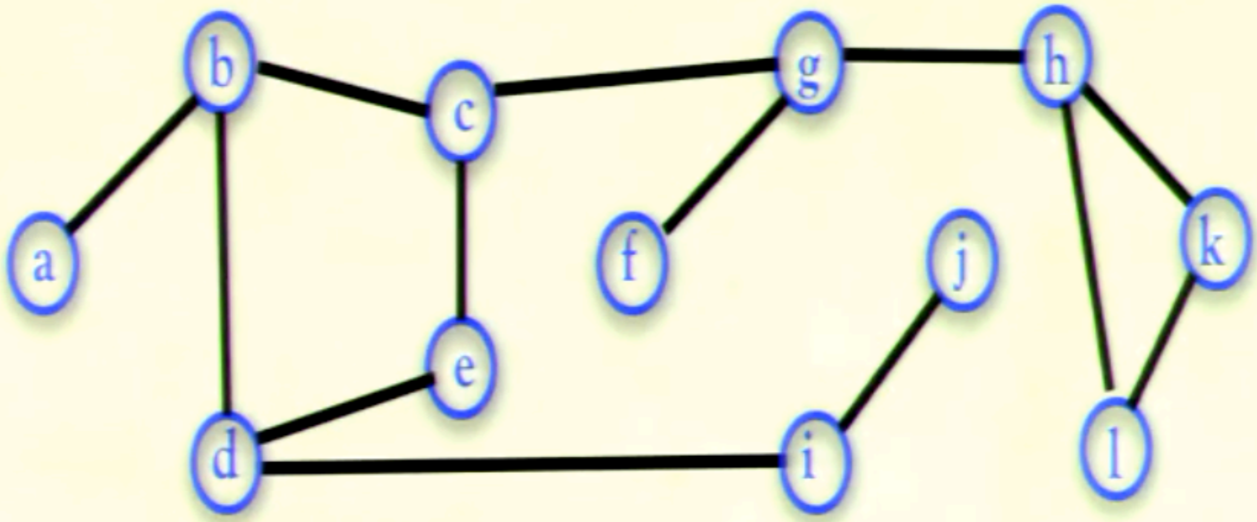
- A **cycle** is a walk $\{v_0, v_1, v_2, \ldots, v_l\}$ where every vertex is <u>distinct</u> except for the end-vertices $v_0 = v_\ell$.



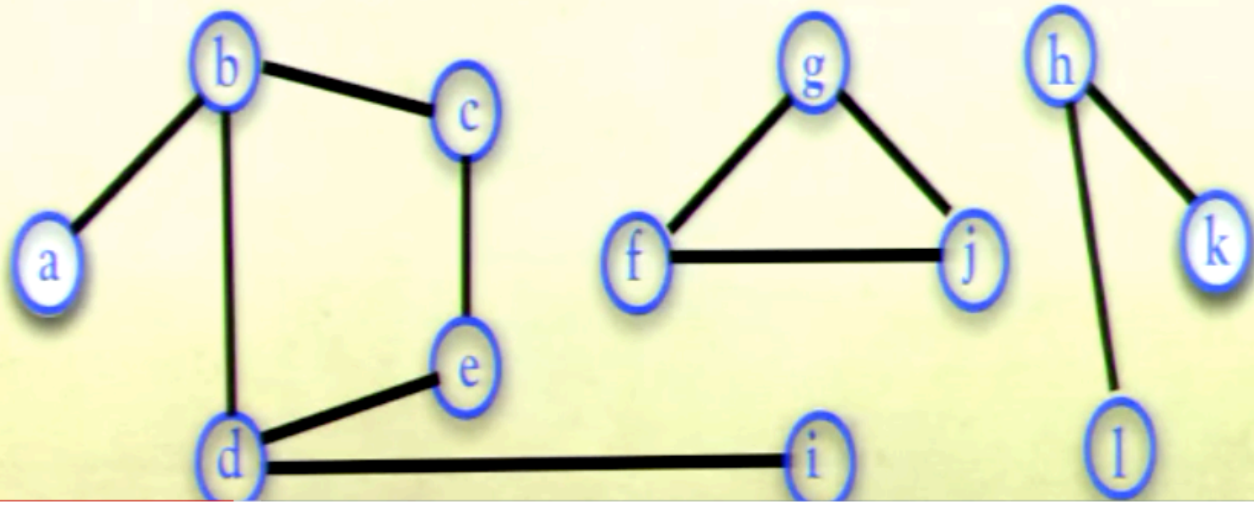
- An **Hamiltonian Cycle** is a cycle that uses every vertex *exactly once*.

Connected Graphs

- A graph is **connected** if for every pair of vertices $u, v \in V$ it is possible to walk from u to v.

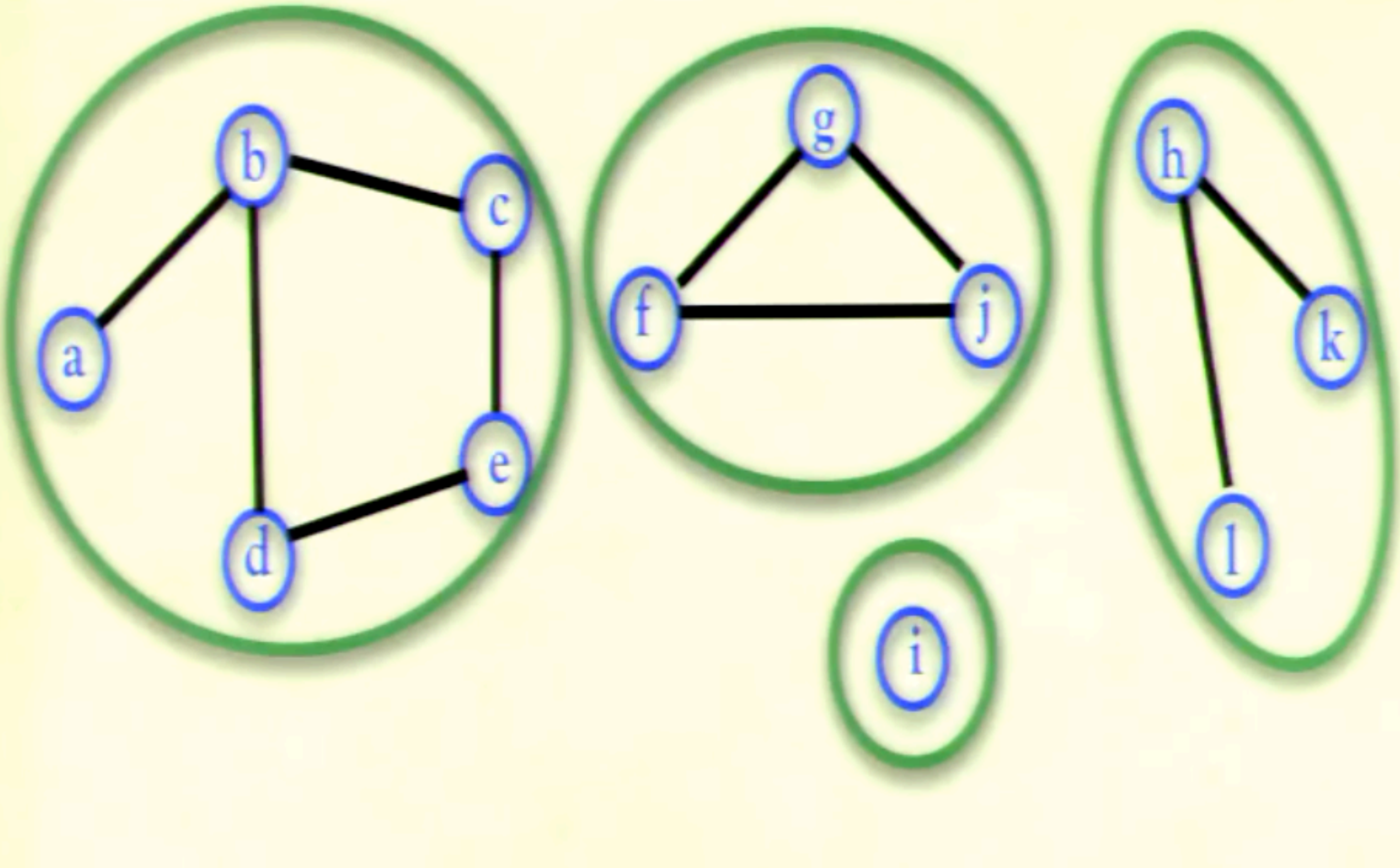


- A graph is **disconnected** if there exists a pair of vertices $u, v \in V$ for which there is no possible walk from u to v.



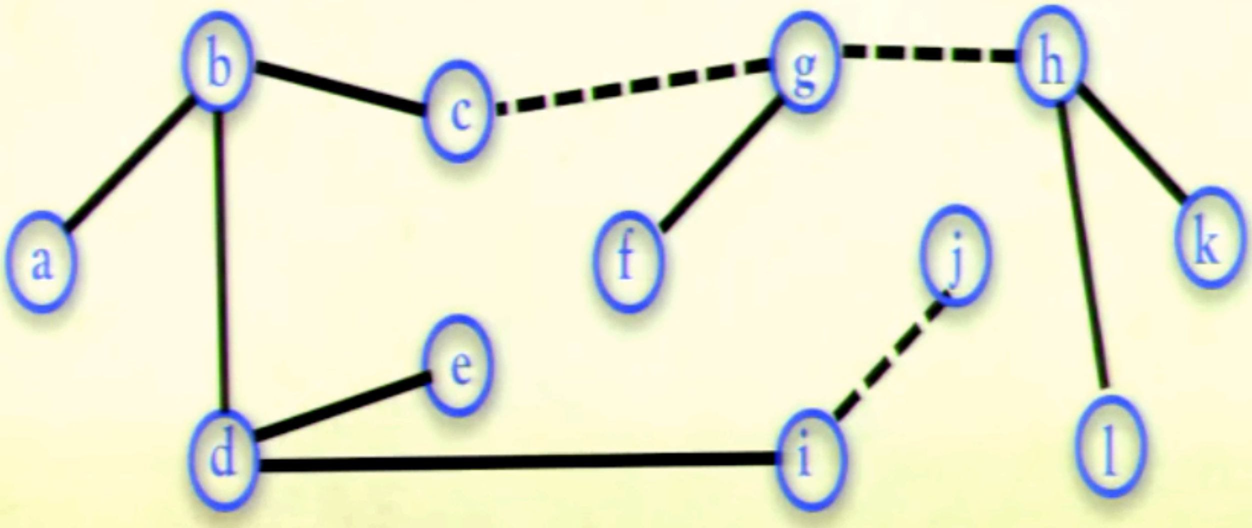
Graph Components

- A connected subgraphs are called the **components** of the graph.



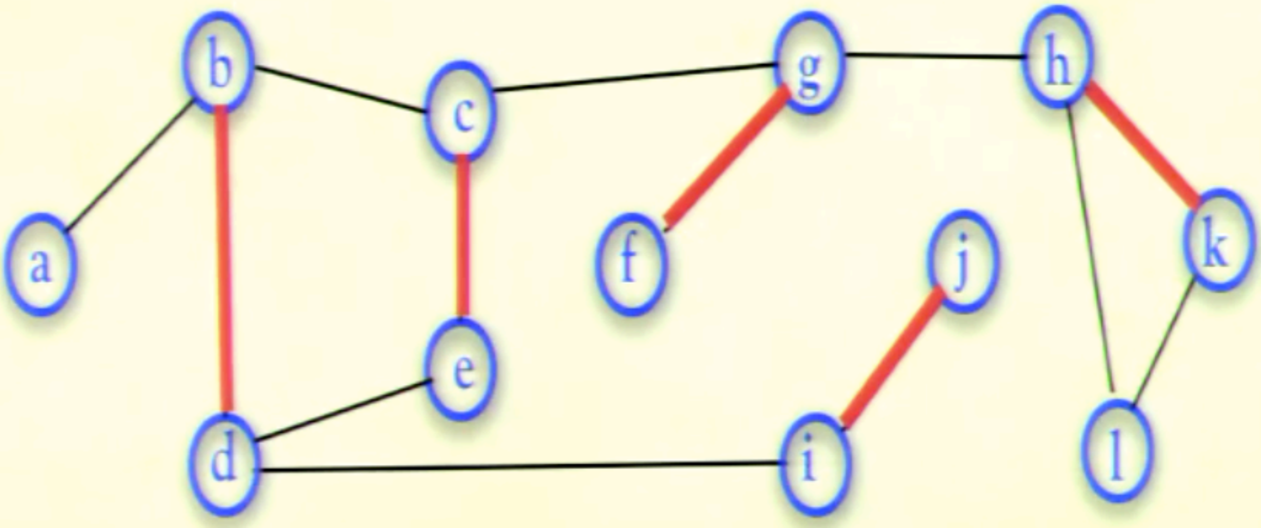
- Thus a *connected graph* has exactly <u>one</u> component.

- A **tree** is a connected component with <u>no</u> cycles.
- A **forest** is a graph whose components are all *trees*.

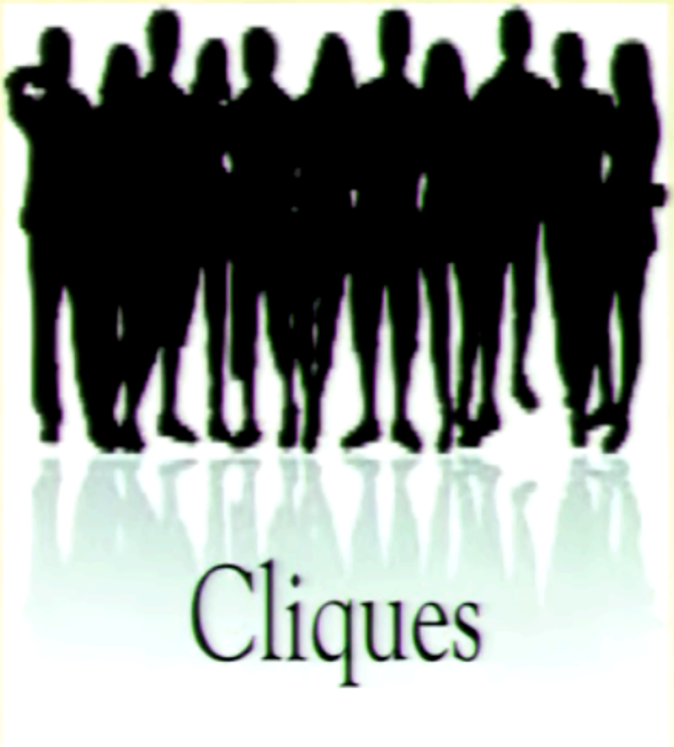


- A tree is **spanning** if it contains every vertex in the graph.

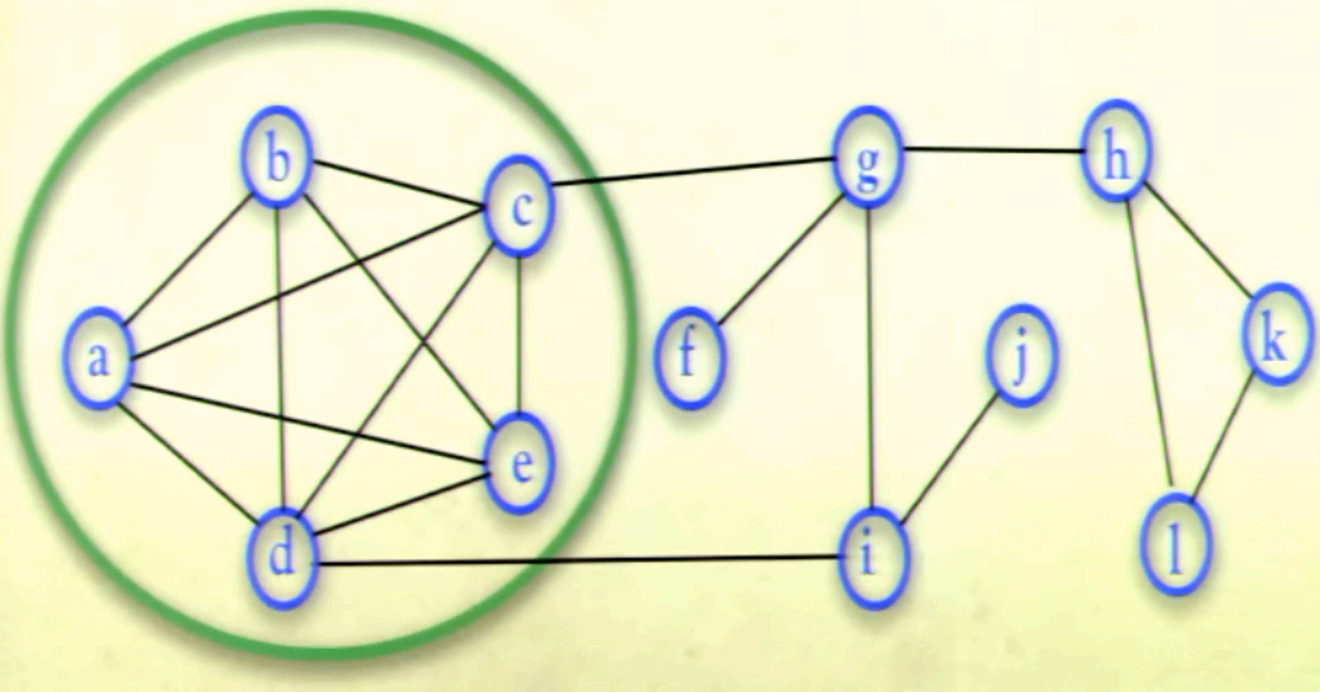
- A **matching** is a set of vertex-disjoint edges.



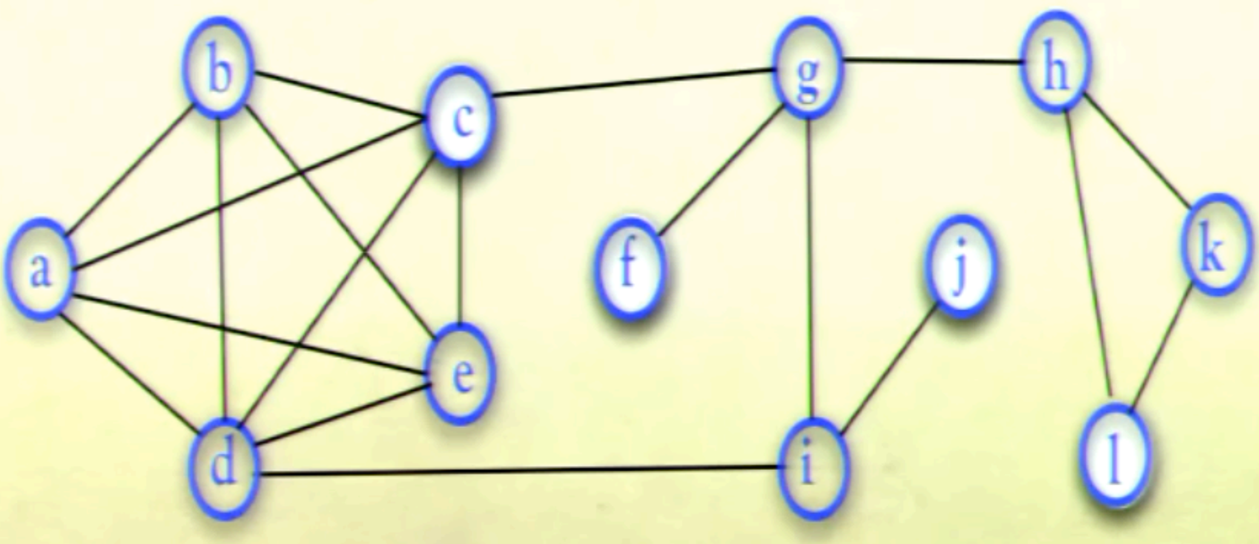
- Hence, each vertex is incident to *at most* one edge in the matching.
- A matching is **perfect** if every vertex is incident to an edge in the matching.



- A **clique** is a set of pairwise adjacent vertices.

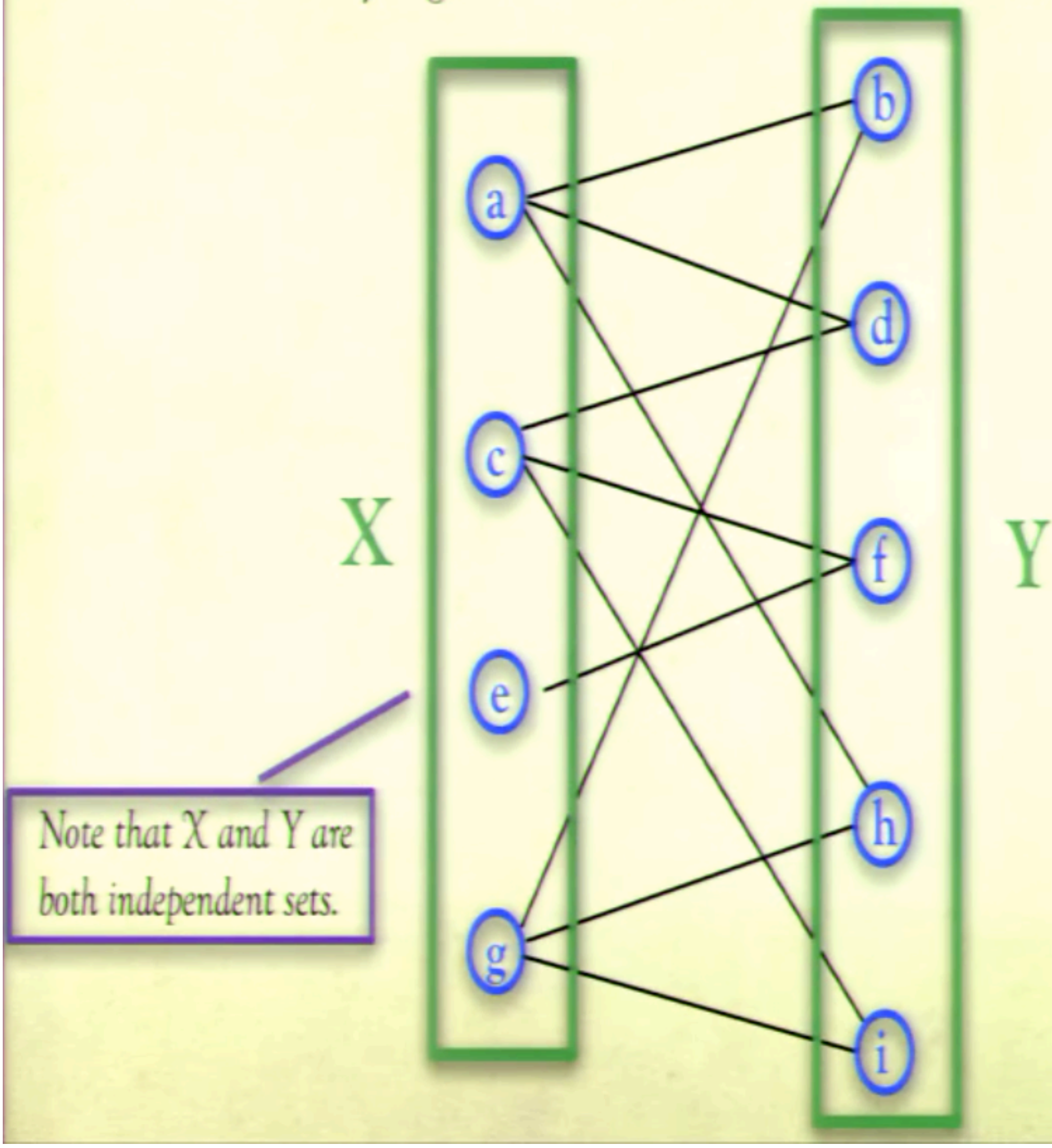


- An **independent set (stable set)** is a set of pairwise non-adjacent vertices.



Bipartite Graphs

- In a **bipartite graph** the vertex set can be partitioned as $V = X \cup Y$ such that every edge has one end-vertex in X and one end-vertex in Y.



The Handshaking Lemma

- Let $\Gamma(v) = \{u : (u, v) \in E\}$ be the set of *neighbours* of *v*.
- The **degree**, $deg(v)$, of a vertex *v* is the cardinality of $\Gamma(v)$.

The Handshaking Lemma. In an undirected graph, there are an even number of vertices with odd degree.

Proof.

- We have:

$$2 \cdot |E| = \sum_{v \in V} deg(v)$$

Double count the number of pairs (v,e) where e is an edge incident to v.

The Handshaking Lemma

- Let $\Gamma(v) = \{u : (u, v) \in E\}$ be the set of *neighbours* of *v*.
- The **degree**, $deg(v)$, of a vertex *v* is the cardinality of $\Gamma(v)$.

The Handshaking Lemma. In an undirected graph, there are an even number of vertices with odd degree.

Proof.

- We have:

$$
\begin{aligned}
2 \cdot |E| &= \sum_{v \in V} deg(v) \\
&= \sum_{v \in \mathcal{O}} deg(v) + \sum_{v \in \mathcal{E}} deg(v)
\end{aligned}
$$

$\mathcal{O}$ is set of vertices with odd degree, and
$\mathcal{E}$ is set of vertices with even degree.

The Handshaking Lemma

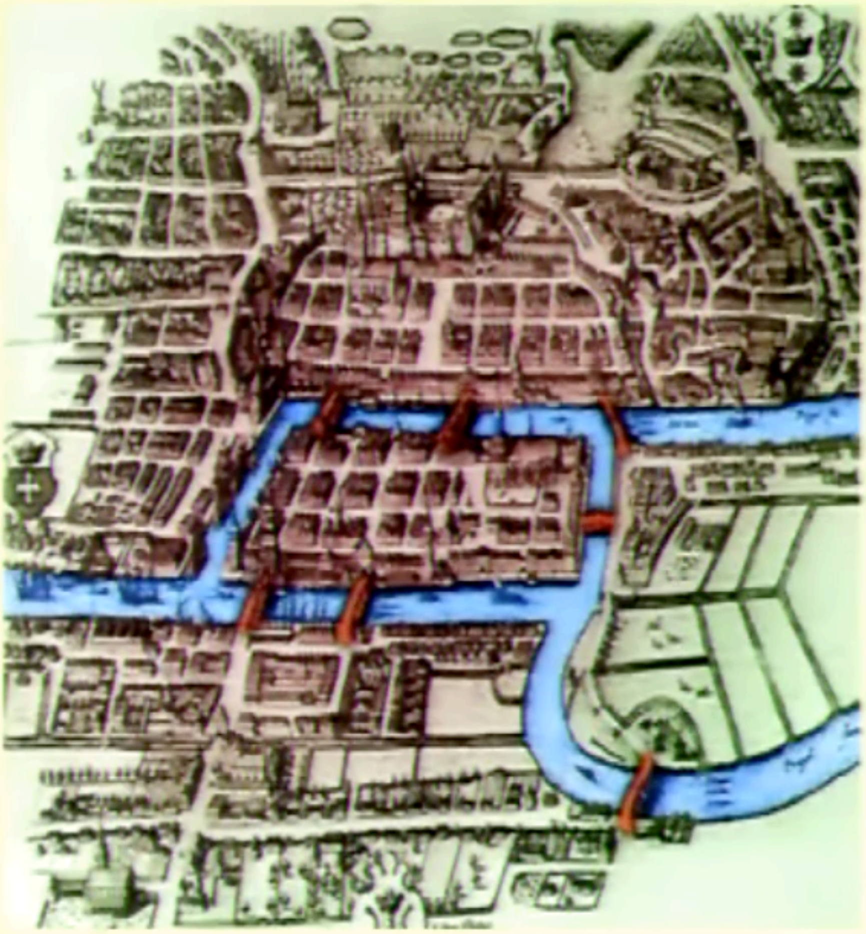
- Let $\Gamma(v) = \{u : (u, v) \in E\}$ be the set of *neighbours* of v.
- The **degree**, $deg(v)$, of a vertex v is the cardinality of $\Gamma(v)$.

The Handshaking Lemma. In an undirected graph, there are an even number of vertices with odd degree.

Proof.

- We have:

$$
\begin{aligned}
2 \cdot |E| &= \sum_{v \in V} deg(v) \\
&= \sum_{v \in \mathcal{O}} deg(v) + \sum_{v \in \mathcal{E}} deg(v) \\
\Rightarrow \quad \underbrace{\sum_{v \in \mathcal{O}} deg(v)}_{\text{Even}} &= \underbrace{2 \cdot |E|}_{\text{Even}} - \underbrace{\sum_{v \in \mathcal{E}} deg(v)}_{\text{Even}}
\end{aligned}
$$



Euler's Theorem

- The first result in Graph Theory is the following.

Theorem. [Euler 1736, Hierholzer 1873] An undirected graph contains an Euler Circuit *if and only if* every vertex has even degree.

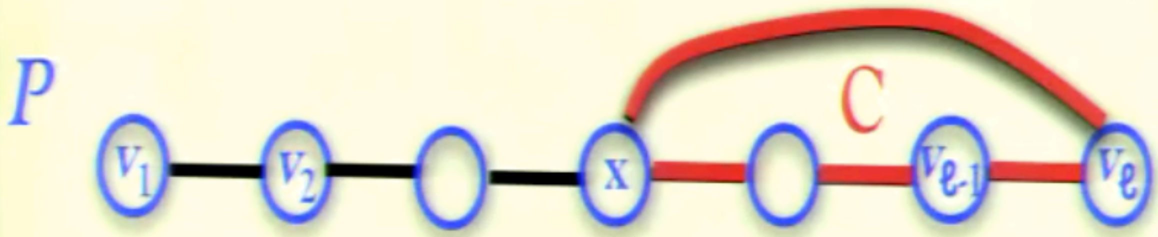
Leaves

- A vertex with degree *one* in a tree is called a **leaf**.

Lemma. A tree T with $n \geq 2$ vertices has at least one leaf vertex.

Proof.

- A tree is connected $\implies$ There are <u>no</u> vertices with degree 0 as $n \geq 2$.
- For a contradiction, assume that every vertex has degree *at least* 2.
- Take the **longest path** $P = \{v_1, v_2, \ldots, v_{\ell-1}, v_\ell\}$ in T.

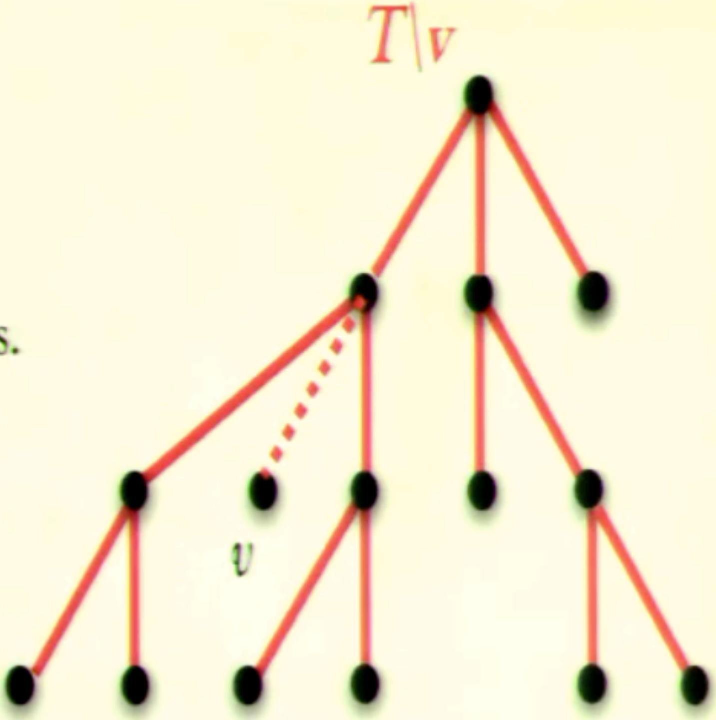


- But $\deg(v_\ell) \geq 2$ so v_ℓ has a neighbour $x \neq v_{\ell-1}$
- We must have $x = v_j$ for some $1 \leq j \leq \ell - 2$ otherwise $\{v_1, v, \ldots, v_\ell, x\}$ is a longer path than P.
- But then $C = \{x = v_j, v_{j+1}, \ldots, v_\ell, x\}$ is a cycle, contradiction. $\square$

The Number of Edges in a Tree

Theorem. A tree with n vertices has n-1 edges.

Proof.



- Let's prove this by induction.

Base Case:

- A tree on **one** vertex has **zero** edges.

Induction Hypothesis

- Assume that *any* tree on n-1 vertices has n-2 edges.

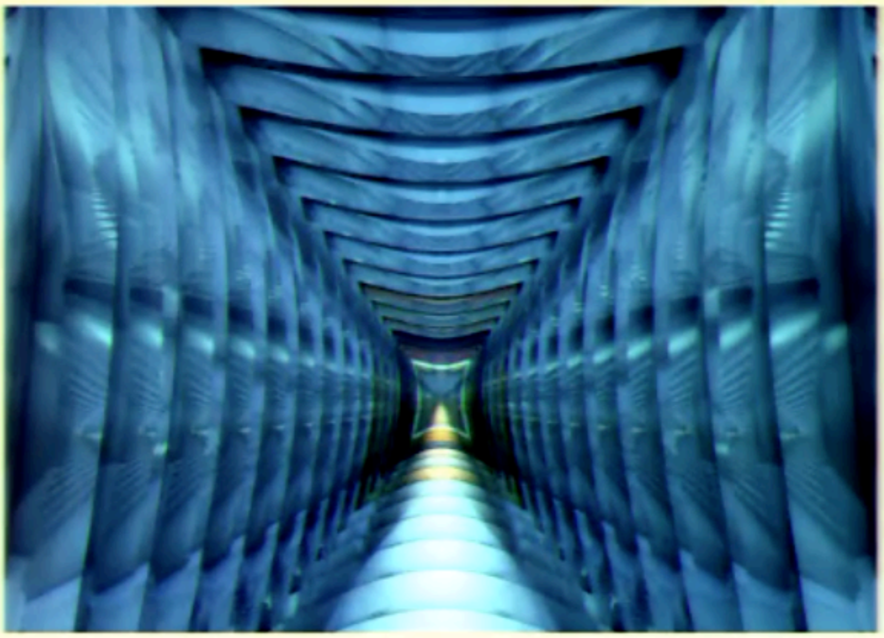
Induction Step

- Take a tree T with $n \geq 2$ vertices.
- By the previous lemma, this tree contains a **leaf** vertex v.

$\Longrightarrow$ $T \setminus \{v\}$ is a tree on n-1 vertices.

$\Longrightarrow$ By the induction hypothesis, $T \setminus \{v\}$ is a tree with n-2 edges.

$\Longrightarrow$ T is a tree with n-1 edges.



Hall's Theorem

- How do we know if a bipartite graph $G = (X \cup Y, E)$ contains a **perfect matching**?
- This is actually easy to test using *Hall's Condition*.

<u>Hall's Condition:</u> $\forall B \subseteq X, \quad |\Gamma(B)| \geq |B|$

Hall's Theorem. A bipartite graph, with $|X| = |Y|$, contains a perfect matching *if and only if* Hall's Condition is satisfied.*

Hall's Theorem

Hall's Theorem. A bipartite graph, with $|X| = |Y|$, contains a perfect matching *if and only if* $\forall B \subseteq X, \quad |\Gamma(B)| \geq |B|$

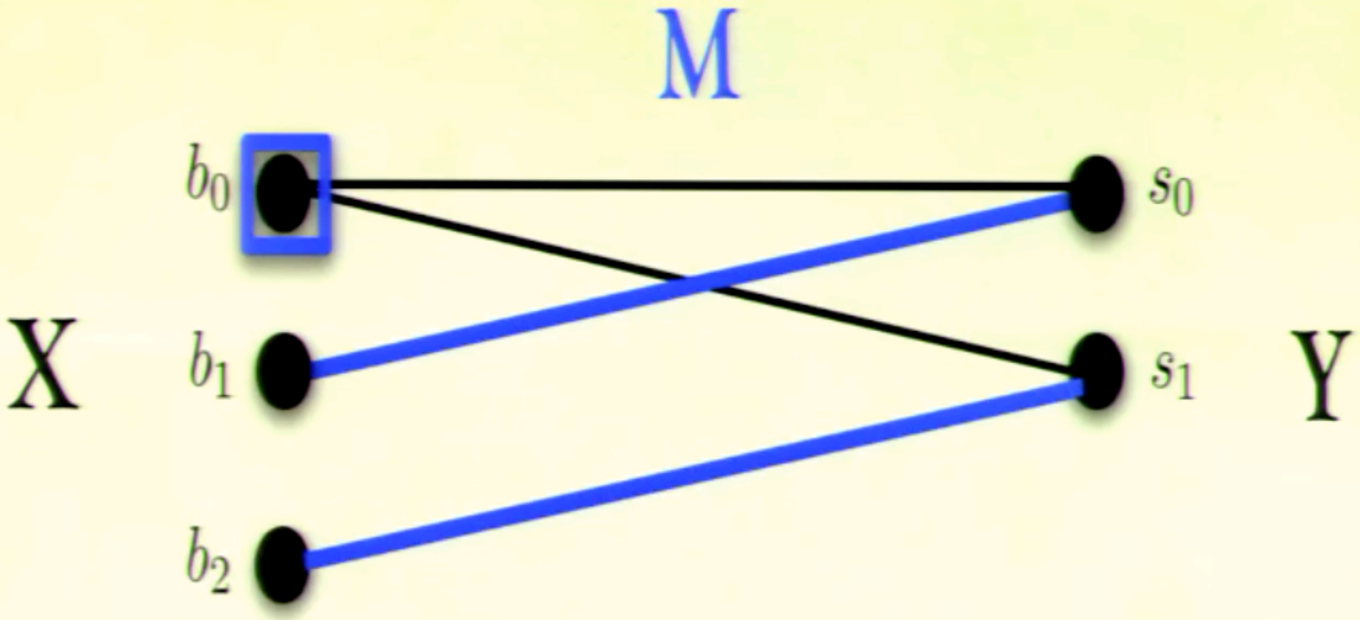
Proof.

$(\Rightarrow)$

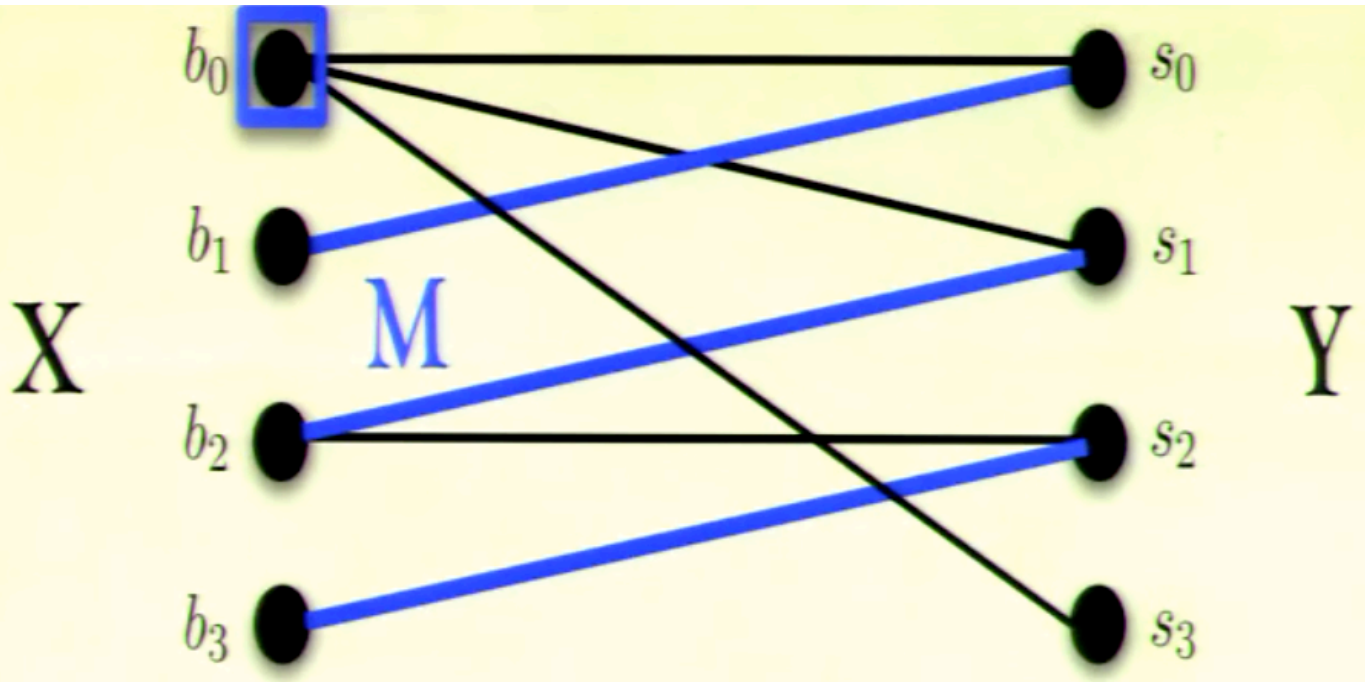
- If there is an set $B \subseteq X$ with $|\Gamma(B)| < |B|$ then the graph **cannot** have a perfect matching.

$(\Leftarrow)$

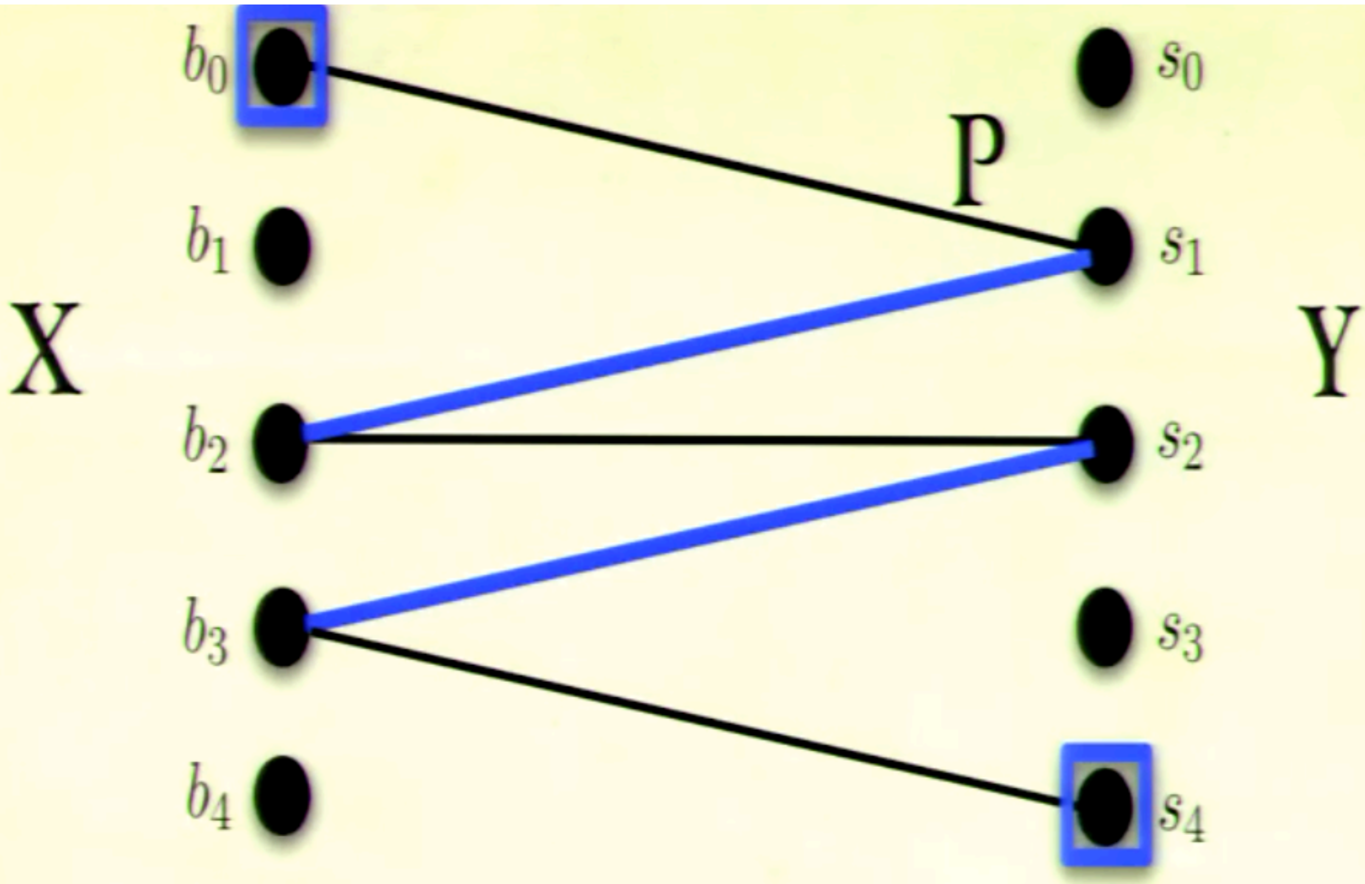
- Suppose Hall's Condition is satisfied: $\forall B \subseteq X, \quad |\Gamma(B)| \geq |B|$
- Take a **maximum cardinality** matching M in the graph.
- If M is *perfect* we are done.
- So we may assume M is <u>not</u> perfect and there is an **unmatched** vertex b_0 in X.



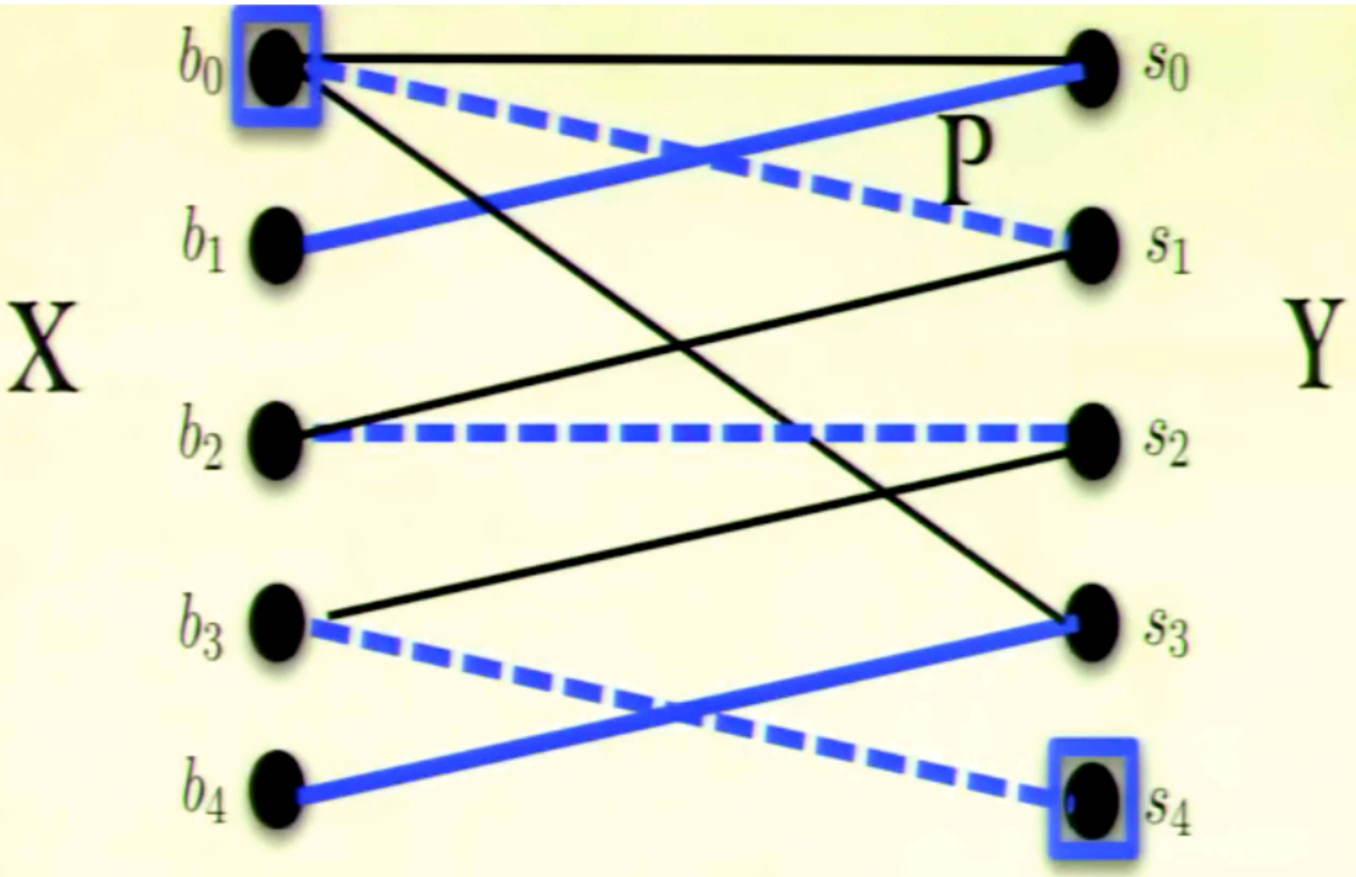
- As Hall's condition holds we have: $|\Gamma(\{b_0\})| \geq |\{b_0\}| = 1$
- So b_0 has at least one neighbour s_0
- Let s_0 be matched to b_1 in M.
- As Hall's condition holds we have: $|\Gamma(\{b_0, b_1\})| \geq |\{b_0, b_1\}| = 2$
- So either b_0 or b_1 has a neighbour $s_1 \neq s_0$
- Let s_1 be matched to b_2 in M.



- By Hall's condition: $|\Gamma(\{b_0, b_1, b_2\})| \geq |\{b_0, b_1, b_2\}| = 3$
- So $\{b_0, b_1, b_2\}$ have a neighbour $s_2 \neq \{s_0, s_1\}$
- Let s_2 be matched to b_3 in M.
- By Hall's condition: $|\Gamma(\{b_0, b_1, b_2, b_3\})| \geq |\{b_0, b_1, b_2, b_3\}| = 4$
- So $\{b_0, b_1, b_2, b_3\}$ have a neighbour $s_3 \neq \{s_0, s_1, s_2\}$
- Let s_3 be matched to b_4 in M, etc...



- The graph contains a *finite number of nodes* so this process must terminate.
- The process can only terminate if we reach an *unmatched* node $s_k \in S$
- Using the edges we have found we can trace back a **path** P from s_k to b_0 that *alternates* between using non-matching edges and matching edges.
- The path P is called an **alternating path.**



- **Swapping** the matching and non-matching edges gives *one extra matching edge*.
- Note this is still a valid matching. Firstly, the internal nodes of P are still incident to exactly one matching edge.
- Secondly, its end-nodes, s_k and b_0, were previously unmatched so are now incident to exactly one edge in the new matching.
- This contradicts the fact that M was a *maximum cardinality matching*. □