# Lecture 11 - Graphs Contd

April 9, 2019

# Recap

- Graphs – finite/infinite graphs, simple/multigraphs, directed/undirected graphs
- Neighbours of a vertex
- Basic but import results:
  - Handshaking theorem – the sum of the degrees of the vertices is twice the number of edges.
  - An undirected graph has an even numbers of vertices of odd degree.
  - In-degree of a vertex $v$ is denoted by $deg^{-}(v)$, Out-degree of a vertex $v$ is denoted by $deg^{+}(v)$.
  - Complete graphs, Cycles, Wheels
  - Bipartite graphs – 2-colourable.

# Matchings!

# Matchings

- A matching $M$ in a simple graph $G = (V, E)$ is a subset of the set $E$ of edges of the graph such that no two edges are incident with the same vertex.

# Matchings

- A matching $M$ in a simple graph $G = (V, E)$ is a subset of the set $E$ of edges of the graph such that no two edges are incident with the same vertex.
- A matching is a subset of edges such that if $\{s, t\}$ and $\{u, v\}$ are distinct edges of the matching, then $s, t, u$, and $v$ are distinct.
- A vertex that is the endpoint of an edge of a matching $M$ is said to be matched in $M$; otherwise it is said to be unmatched.
- A maximum matching is a matching with the largest number of edges.

# Complete Matchings

- We say that a matching $M$ in a bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ is a complete matching from $V_1$ to $V_2$ if every vertex in $V_1$ is the endpoint of an edge in the matching of equivalently if $|M| = |V_1|$.

# Complete Matchings

- We say that a matching $M$ in a bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ is a complete matching from $V_1$ to $V_2$ if every vertex in $V_1$ is the endpoint of an edge in the matching of equivalently if $|M| = |V_1|$.
- Perfect matching or 1-factor – A perfect matching is a matching involving all the vertices of a graph not necessarily bipartite graphs.

# Complete Matchings

- We say that a matching $M$ in a bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ is a complete matching from $V_1$ to $V_2$ if every vertex in $V_1$ is the endpoint of an edge in the matching of equivalently if $|M| = |V_1|$.
- Perfect matching or 1-factor – A perfect matching is a matching involving all the vertices of a graph not necessarily bipartite graphs.
- A perfect matching exhausts all of the vertices, so a bipartite graph that has a perfect matching must have the same number of vertices in each part – a complete matching from one part into the other.

# Complete Matchings

- We say that a matching $M$ in a bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ is a complete matching from $V_1$ to $V_2$ if every vertex in $V_1$ is the endpoint of an edge in the matching of equivalently if $|M| = |V_1|$.
- Perfect matching or 1-factor – A perfect matching is a matching involving all the vertices of a graph not necessarily bipartite graphs.
- A perfect matching exhausts all of the vertices, so a bipartite graph that has a perfect matching must have the same number of vertices in each part – a complete matching from one part into the other.
- A perfect matching is therefore a matching containing $n/2$ edges (the largest possible), meaning perfect matchings are only possible on graphs with an even number of vertices.

# Hall's Marriage Problem

**Theorem (Hall's Marriage Theorem (Philip Hall (1935)))**
*The bipartite graph* $G = (V, E)$ *with bipartition* $(V_1, V_2)$ *has a complete matching from* $V_1$ *to* $V_2$ *if and only if* $|N(A)| \geq |A|$ *for all subsets* $A$ *of* $V_1$*.*

Proof:

- 'Only if': Assume that there is a complete matching $M$ from $V_1$ to $V_2$.

**Theorem (Hall's Marriage Theorem (Philip Hall (1935)))**
*The bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ has a complete matching from $V_1$ to $V_2$ if and only if $|N(A)| \geq |A|$ for all subsets $A$ of $V_1$.*

Proof:

- 'Only if': Assume that there is a complete matching $M$ from $V_1$ to $V_2$.
- Let $A \subseteq V_1$ for every vertex $v \in A$, there is an $e \in M$ connecting $v$ to a vertex in $V_2$.

# Hall's Marriage Problem

**Theorem (Hall's Marriage Theorem (Philip Hall (1935)))**
*The bipartite graph $G = (V, E)$ with bipartition $(V_1, V_2)$ has a complete matching from $V_1$ to $V_2$ if and only if $|N(A)| \geq |A|$ for all subsets $A$ of $V_1$.*

Proof:

- 'Only if': Assume that there is a complete matching $M$ from $V_1$ to $V_2$.
- Let $A \subseteq V_1$ for every vertex $v \in A$, there is an $e \in M$ connecting $v$ to a vertex in $V_2$.
- This implies there are as at least as many vertices in $V_2$ that are neighbours of vertices in $V_1$ as there are vertices in $V_1$, therefore $|N(A)| \geq |A|$.

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.
- Strong Induction is what we use – When using (weak) induction, we assume that $P(k)$ is true to prove $P(k+1)$. In strong induction, we assume that all of $P(1), P(2), \ldots, P(k)$ are true to prove $P(k+1)$.

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.
- Strong Induction is what we use – When using (weak) induction, we assume that $P(k)$ is true to prove $P(k+1)$. In strong induction, we assume that all of $P(1), P(2), \ldots, P(k)$ are true to prove $P(k+1)$. Is one really stronger than the other?

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.
- Strong Induction is what we use – When using (weak) induction, we assume that $P(k)$ is true to prove $P(k+1)$. In strong induction, we assume that all of $P(1), P(2), \ldots, P(k)$ are true to prove $P(k+1)$. Is one really stronger than the other?
- Basis step: If $|V_1| = 1$, then $V_1$ contains a single vertex $v_0$.

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.
- Strong Induction is what we use – When using (weak) induction, we assume that $P(k)$ is true to prove $P(k+1)$. In strong induction, we assume that all of $P(1), P(2), \ldots, P(k)$ are true to prove $P(k+1)$. Is one really stronger than the other?
- Basis step: If $|V_1| = 1$, then $V_1$ contains a single vertex $v_0$.
- $|N(\{v_0\})| \geq |\{v_0\}| = 1 \Rightarrow$ there is $\geq 1$ edge connecting $v_0$ and $w_0 \in V_2$ – a complete matching from $V_1$ to $V_2$.

# Hall's Marriage Problem

- 'If': Assume that if $|N(A)| \geq |A|$ for all $A \subseteq V_1$ then there is a complete matching $M$ from $V_1$ to $V_2$.
- Strong Induction is what we use – When using (weak) induction, we assume that $P(k)$ is true to prove $P(k+1)$. In strong induction, we assume that all of $P(1), P(2), \ldots, P(k)$ are true to prove $P(k+1)$. Is one really stronger than the other?
- Basis step: If $|V_1| = 1$, then $V_1$ contains a single vertex $v_0$.
- $|N(\{v_0\})| \geq |\{v_0\}| = 1 \Rightarrow$ there is $\geq 1$ edge connecting $v_0$ and $w_0 \in V_2$ – a complete matching from $V_1$ to $V_2$.
- Inductive hypothesis :
  - Let $k \in \mathbb{Z}_{\geq 0}$, $G = (V, E)$, bipartite graph with bipartition $(V_1, V_2)$.
  - Let $|V_1| = j \leq k$ then there is a complete matching $M$ from $V_1$ to $V_2$ whenever $|N(A) \geq |A|, \forall A \subseteq V_1$.

# Hall's Marriage Problem

- Let $H = (W, F)$ is a bipartite graph with bipartition $(W_1, W_2)$ and $|W_1| = k + 1$. We need to consider only the following two cases.

# Hall's Marriage Problem

- Let $H = (W, F)$ is a bipartite graph with bipartition $(W_1, W_2)$ and $|W_1| = k + 1$. We need to consider only the following two cases.
- Case(i) : For all integers $j$, s.t. $1 \leq j \leq k$, the vertices in every set of $j$ elements from $W_1$ are adjacent to at least $j + 1$ elements of $W_2$.
- Case (ii) : For some $j$ with $1 \leq j \leq k$ there is a subset $W_1{}'$ of $j$ vertices such that there are exactly $j$ neighbours of these vertices in $W_2$.

Why not consider a subset of $W_1$ of $k + 1$ elements?
Why not consider a subset of $W_1$ of size $j$ with less than $j$ neighbours?

## Hall's Marriage Problem

Case (i): For all integers $j$ with $1 \leq j \leq k$, the vertices in every subset of $j$ elements from $W_1$ are adjacent to at least $j+1$ elements of $W_2$.

- Select a vertex $v \in W_1$ and an element $w \in N(\{v\})$ (Must exist as per our assumption that $|N(\{v\})| \geq |\{v\}| = 1$)

# Hall's Marriage Problem

Case (i): For all integers $j$ with $1 \le j \le k$, the vertices in every subset of $j$ elements from $W_1$ are adjacent to at least $j+1$ elements of $W_2$.

- Select a vertex $v \in W_1$ and an element $w \in N(\{v\})$ (Must exist as per our assumption that $|N(\{v\})| \geq |\{v\}| = 1$)
- We delete $v$ and $w$ and all edges incident to them from $H$.

# Hall's Marriage Problem

Case (i): For all integers $j$ with $1 \leq j \leq k$, the vertices in every subset of $j$ elements from $W_1$ are adjacent to at least $j + 1$ elements of $W_2$.

- Select a vertex $v \in W_1$ and an element $w \in N(\{v\})$ (Must exist as per our assumption that $|N(\{v\})| \geq |\{v\}| = 1$)
- We delete $v$ and $w$ and all edges incident to them from $H$.
- This produces a bipartite graph $H^{'}$ with bipartition $(W_1 \setminus \{v\}, W_2 \setminus \{w\})$.

## Hall's Marriage Problem

Case (i): For all integers $j$ with $1 \leq j \leq k$, the vertices in every subset of $j$ elements from $W_1$ are adjacent to at least $j+1$ elements of $W_2$.

- Select a vertex $v \in W_1$ and an element $w \in N(\{v\})$ (Must exist as per our assumption that $|N(\{v\})| \geq |\{v\}| = 1$)
- We delete $v$ and $w$ and all edges incident to them from $H$.
- This produces a bipartite graph $H^{'}$ with bipartition $(W_1 \setminus \{v\}, W_2 \setminus \{w\})$.
- $|W \setminus \{v\}| = k$ from inductive hypothesis we have a complete matching from $W_1 - \{v\}$ to $W_2 - ske\{w\}$.

## Hall's Marriage Problem

Case (i): For all integers $j$ with $1 \leq j \leq k$, the vertices in every subset of $j$ elements from $W_1$ are adjacent to at least $j+1$ elements of $W_2$.

- Select a vertex $v \in W_1$ and an element $w \in N(\{v\})$ (Must exist as per our assumption that $|N(\{v\})| \geq |\{v\}| = 1$)
- We delete $v$ and $w$ and all edges incident to them from $H$.
- This produces a bipartite graph $H^{'}$ with bipartition $(W_1 \setminus \{v\}, W_2 \setminus \{w\})$.
- $|W \setminus \{v\}| = k$ from inductive hypothesis we have a complete matching from $W_1 - \{v\}$ to $W_2 - ske\{w\}$.
- Adding the edge from $v$ to $w$ – a complete matching from $W_1$ to $W_2$.

# Hall's Marriage Problem

Case (ii): Suppose that for some $j$ with $1 \leq j \leq k$, there is a subset $W_1{}'$ of $j$ vertices such that there are exactly $j$ neighbors of these vertices in $W_2$.

- Let $W_2{}'$ be the set of these neighbors.

# Hall's Marriage Problem

Case (ii): Suppose that for some $j$ with $1 \leq j \leq k$, there is a subset $W_1{}'$ of $j$ vertices such that there are exactly $j$ neighbors of these vertices in $W_2$.

- Let $W_2{}'$ be the set of these neighbors.
- By inductive hypothesis – there is a complete matching from $W_1{}'$ to $W_2{}'$.

# Hall's Marriage Problem

Case (ii): Suppose that for some $j$ with $1 \leq j \leq k$, there is a subset $W_1^{'}$ of $j$ vertices such that there are exactly $j$ neighbors of these vertices in $W_2$.

- Let $W_2^{'}$ be the set of these neighbors.
- By inductive hypothesis – there is a complete matching from $W_1^{'}$ to $W_2^{'}$.
- Remove these $2j$ vertices from $W_1$ and $W_2$ and all incident edges – we get a bipartite graph $K$ with bipartition $(W_1 - W_1^{'}, W_2 - W_2^{'})$.

# Hall's Marriage Problem

- Claim: $K$ satisfies the condition $|N(A)| \geq |A|$ for all subsets $A$ of $W_1 - {W_1}'$.

# Hall's Marriage Problem

- Claim: $K$ satisfies the condition $|N(A)| \geq |A|$ for all subsets $A$ of $W_1 - {W_1}'$.
- If not,there would be a subset of size $t$ of $W_1 - {W_1}'$ where $1 \leq t \leq k + 1 - j$ with fewer than $t$ vertices of $W_2 - W_2'$ as neighbours.

# Hall's Marriage Problem

- Claim: $K$ satisfies the condition $|N(A)| \geq |A|$ for all subsets $A$ of $W_1 - {W_1}'$.
- If not,there would be a subset of size $t$ of $W_1 - {W_1}'$ where $1 \leq t \leq k + 1 - j$ with fewer than $t$ vertices of $W_2 - W_2'$ as neighbours.
- But then the set of $j + t$ vertices of $W_1$ with these $t$ vertices together with the $j$ vertices we removed from $W_1$ has fewer than $j + t$ neighbors in $W_2$, contradicting the hypothesis that $|N(A)| \geq |A|$ for all $A \subseteq W_1$.

# Hall's Marriage Problem

- Claim: $K$ satisfies the condition $|N(A)| \geq |A|$ for all subsets $A$ of $W_1 - {W_1}'$.
- If not,there would be a subset of size $t$ of $W_1 - {W_1}'$ where $1 \leq t \leq k + 1 - j$ with fewer than $t$ vertices of $W_2 - W_2'$ as neighbours.
- But then the set of $j + t$ vertices of $W_1$ with these $t$ vertices together with the $j$ vertices we removed from $W_1$ has fewer than $j + t$ neighbors in $W_2$, contradicting the hypothesis that $|N(A)| \geq |A|$ for all $A \subseteq W_1$.
- From inductive hypothesis, the graph $K$ has a complete matching. Combining this complete matching with the complete matching from ${W_1}'$ to ${W_2}'$ we get a complete matching from $W_1$ to $W_2$.

# Matchings

- Read up on a constructive proof for Hall's Marriage problem.

# Matchings

- Read up on a constructive proof for Hall's Marriage problem.
- A graph is said to be regular if every node has the same degree.

# Matchings

- Read up on a constructive proof for Hall's Marriage problem.
- A graph is said to be regular if every node has the same degree.
- Exercise - Every regular bipartite graph has a perfect matching.

# Stable Marriage Problem/ stable matching problem /SMP

- The problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences.

# Stable Marriage Problem/ stable matching problem /SMP

- The problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences.
- When is a matching stable?

# Stable Marriage Problem/ stable matching problem /SMP

- The problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences.
- When is a matching stable?
  - There is an element $A$ of the first set which prefers some element $B$ of the second set over the element it was matched .
  - $B$ also prefers $A$ over the element to which $B$ is already matched.

# Stable Marriage Problem/ stable matching problem /SMP

- The problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences.
- When is a matching stable?
  - There is an element $A$ of the first set which prefers some element $B$ of the second set over the element it was matched .
  - $B$ also prefers $A$ over the element to which $B$ is already matched.
- Given $n$ men and $n$ women, where each person has ranked all members of the opposite sex in order of preference, marry them s.t. there are no two people of opposite sex who would both rather have each other than their current partners – the set of marriages is deemed stable.

## Stable Marriage Problem/ stable matching problem /SMP

- The problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences.
- When is a matching stable?
  - There is an element $A$ of the first set which prefers some element $B$ of the second set over the element it was matched .
  - $B$ also prefers $A$ over the element to which $B$ is already matched.
- Given $n$ men and $n$ women, where each person has ranked all members of the opposite sex in order of preference, marry them s.t. there are no two people of opposite sex who would both rather have each other than their current partners – the set of marriages is deemed stable.
- The Gale-Shapley algorithm (or the Deferred Acceptance algorithm) – $O(n^2)$ algorithm, $n$ is number of men or women. Ex - Read up on this!

# Representing Graphs, Creating New Graphs from Old

## Representing Simple Graphs

- Adjacency lists – specify the vertices that are adjacent to each vertex of the graph.

# Representing Simple Graphs

- Adjacency lists – specify the vertices that are adjacent to each vertex of the graph.
- Example

## Representing Simple Graphs

- Adjacency lists – specify the vertices that are adjacent to each vertex of the graph.
- Example
- Adjacency matrix $A$ (or $A_G$) of $G = (V, E)$, w.r.t. the listing of the vertices $v_1, \ldots, v_n$, is the $n \times n$ zeroone matrix with 1 as its $(i, j)$th entry when $v_i$ and $v_j$ are adjacent, and 0 as its $(i, j)$th entry when they are not adjacent.
- Its adjacency matrix is $A = [a_{ij}]$, then

$$\begin{aligned} a_{ij} &= 1 \text{ if } \{v_i, v_j\} \text{is an edge of } G \\ &= 0 \text{ otherwise .} \end{aligned}$$

## Representing Simple Graphs

- Adjacency lists – specify the vertices that are adjacent to each vertex of the graph.
- Example
- Adjacency matrix $A$ (or $A_G$) of $G = (V, E)$, w.r.t. the listing of the vertices $v_1, \ldots, v_n$, is the $n \times n$ zeroone matrix with 1 as its $(i, j)$th entry when $v_i$ and $v_j$ are adjacent, and 0 as its $(i, j)$th entry when they are not adjacent.
- Its adjacency matrix is $A = [a_{ij}]$, then

$$\begin{aligned} a_{ij} &= 1 \text{ if } \{v_i, v_j\} \text{is an edge of } G \\ &= 0 \text{ otherwise } . \end{aligned}$$

- Examples –

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n!$ different adjacency matrices for a graph with $n$ vertices!

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n$! different adjacency matrices for a graph with $n$ vertices!
- For a simple graph it is symmetric, i.e. $a_{ij} = a_{ji}$.

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n!$ different adjacency matrices for a graph with $n$ vertices!
- For a simple graph it is symmetric, i.e. $a_{ij} = a_{ji}$.
- Since a simple graphs has no loops, each entry $a_{ii}$ is 0.

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n!$ different adjacency matrices for a graph with $n$ vertices!
- For a simple graph it is symmetric, i.e. $a_{ij} = a_{ji}$.
- Since a simple graphs has no loops, each entry $a_{ii}$ is 0.
- Adjacency matrices for undirected graphs multigraphs - a loop at $v_i$ is rep by a 1 at the $(i, i)$th position.

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n!$ different adjacency matrices for a graph with $n$ vertices!
- For a simple graph it is symmetric, i.e. $a_{ij} = a_{ji}$.
- Since a simple graphs has no loops, each entry $a_{ii}$ is 0.
- Adjacency matrices for undirected graphs multigraphs - a loop at $v_i$ is rep by a 1 at the $(i, i)$th position.
- For multiple edges connecting the same pair of vertices $v_i$ and $v_j$ , (or multiple loops), the adjacency matrix is no longer a zero-one matrix, the $(i, j)$th entry equals the number of edges that are associated to $\{v_i, v_j\}$.

# Adjacency matrices

- An adjacency matrix of a graph depends on the ordering chosen for the vertices – there may be as many as $n!$ different adjacency matrices for a graph with $n$ vertices!
- For a simple graph it is symmetric, i.e. $a_{ij} = a_{ji}$.
- Since a simple graphs has no loops, each entry $a_{ii}$ is 0.
- Adjacency matrices for undirected graphs multigraphs - a loop at $v_i$ is rep by a 1 at the $(i, i)$th position.
- For multiple edges connecting the same pair of vertices $v_i$ and $v_j$ , (or multiple loops), the adjacency matrix is no longer a zero-one matrix, the $(i, j)$th entry equals the number of edges that are associated to $\{v_i, v_j\}$.
- All undirected graphs, including multigraphs and pseudographs, have symmetric adjacency matrices.

# Adjacency matrices for directed graphs

- The matrix for a directed graph $G = (V, E)$ has a 1 in its $(i, j)$th position if there is an edge from $v_i$ to $v_j$, where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph.

# Adjacency matrices for directed graphs

- The matrix for a directed graph $G = (V, E)$ has a 1 in its $(i, j)$th position if there is an edge from $v_i$ to $v_j$, where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph.
- If $A = [a_{ij}]$ is the adjacency matrix for the directed graph with respect to this listing of the vertices, then

$$\begin{aligned} a_{ij} &= 1 \text{ if } (v_i, v_j) \text{is an edge of } G \\ &= 0 \text{ otherwise .} \end{aligned}$$

# Adjacency matrices for directed graphs

- The matrix for a directed graph $G = (V, E)$ has a 1 in its $(i, j)$th position if there is an edge from $v_i$ to $v_j$, where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph.
- If $A = [a_{ij}]$ is the adjacency matrix for the directed graph with respect to this listing of the vertices, then

$$\begin{aligned} a_{ij} &= 1 \text{ if } (v_i, v_j) \text{is an edge of } G \\ &= 0 \text{ otherwise }. \end{aligned}$$

- The adjacency matrix for a directed graph does not have to be symmetric.

# Adjacency matrices for directed graphs

- The matrix for a directed graph $G = (V, E)$ has a 1 in its $(i, j)$th position if there is an edge from $v_i$ to $v_j$, where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph.
- If $A = [a_{ij}]$ is the adjacency matrix for the directed graph with respect to this listing of the vertices, then

$$\begin{aligned} a_{ij} &= 1 \text{ if } (v_i, v_j) \text{is an edge of } G \\ &= 0 \text{ otherwise .} \end{aligned}$$

- The adjacency matrix for a directed graph does not have to be symmetric.
- Adjacency matrices can also be used to represent directed multigraphs – again not a zero-one matrix then!

## Trade-Offs between Adjacency matrices and Adjacency lists

- For a simple graph contains relatively few edges, that is, sparse , adjacency lists are better.

# Trade-Offs between Adjacency matrices and Adjacency lists

- For a simple graph contains relatively few edges, that is, sparse , adjacency lists are better.
- For eg, if each vertex has degree not exceeding $c$, a constant much smaller than $n$, then each adjacency list contains $c$ or fewer vertices. Hence, there are no more than $cn$ items in all these adjacency lists.

## Trade-Offs between Adjacency matrices and Adjacency lists

- For a simple graph contains relatively few edges, that is, sparse , adjacency lists are better.
- For eg, if each vertex has degree not exceeding $c$, a constant much smaller than $n$, then each adjacency list contains $c$ or fewer vertices. Hence, there are no more than $cn$ items in all these adjacency lists.
- But the adjacency matrix for the graph has $n^2$ entries.

# Trade-Offs between Adjacency matrices and Adjacency lists

- For a simple graph contains relatively few edges, that is, sparse , adjacency lists are better.
- For eg, if each vertex has degree not exceeding $c$, a constant much smaller than $n$, then each adjacency list contains $c$ or fewer vertices. Hence, there are no more than $cn$ items in all these adjacency lists.
- But the adjacency matrix for the graph has $n^2$ entries.
- But it is a sparse matrix for which there are special techniques for representing and computing.

## Trade-Offs between Adjacency matrices and Adjacency lists

- For a dense simple graph, (it contains more than half of all possible edges) then we prefer an adjacency matrix over an adjacency list.

# Trade-Offs between Adjacency matrices and Adjacency lists

- For a dense simple graph, (it contains more than half of all possible edges) then we prefer an adjacency matrix over an adjacency list. Consider the complexity of determining whether the possible edge $\{v_i, v_j\}$ is present.

# Trade-Offs between Adjacency matrices and Adjacency lists

- For a dense simple graph, (it contains more than half of all possible edges) then we prefer an adjacency matrix over an adjacency list. Consider the complexity of determining whether the possible edge $\{v_i, v_j\}$ is present.
- In an adjacency matrix, its just a look up of $(i, j)$th entry in the matrix and one comparison with zero or one.

# Trade-Offs between Adjacency matrices and Adjacency lists

- For a dense simple graph, (it contains more than half of all possible edges) then we prefer an adjacency matrix over an adjacency list. Consider the complexity of determining whether the possible edge $\{v_i, v_j\}$ is present.
- In an adjacency matrix, its just a look up of $(i, j)$th entry in the matrix and one comparison with zero or one.
- In case of adjacency lists, we need to search the list of vertices adjacent to either $v_i$ or $v_j$ to determine whether this edge is present. This can require $\Theta(|V|)$ comparisons when many edges are present.

# Incidence Matrix

- Another common way to represent graphs is to use incidence matrices.
- Let $G = (V, E)$ be an undirected graph.
- Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of $G$.

# Incidence Matrix

- Another common way to represent graphs is to use incidence matrices.
- Let $G = (V, E)$ be an undirected graph.
- Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of $G$.
- Then the incidence matrix w.r.t. this ordering of $V$ and $E$ is the $n \times m$ matrix $M = [m_{ij}]$, where

$$\begin{aligned} m_{ij} &= 1 \text{ when edge } e_j \text{ is incident with } v_i \\ &= 0 \text{ otherwise.} \end{aligned}$$

# Incidence Matrix

- Another common way to represent graphs is to use incidence matrices.
- Let $G = (V, E)$ be an undirected graph.
- Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of $G$.
- Then the incidence matrix w.r.t. this ordering of $V$ and $E$ is the $n \times m$ matrix $M = [m_{ij}]$, where

$$\begin{aligned} m_{ij} &= 1 \text{ when edge } e_j \text{ is incident with } v_i \\ &= 0 \text{ otherwise.} \end{aligned}$$

- Example –

# Subgraphs

Sometimes we work with only part of the graph. The smaller graph is called a subgraph .

**Definition**

A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$, where $W \subseteq V$ and $F \subseteq E$. A subgraph $H$ of $G$ is a proper subgraph of $G$ if $H \neq G$.

# Subgraphs

Sometimes we work with only part of the graph. The smaller graph is called a subgraph .

**Definition**

A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$, where $W \subseteq V$ and $F \subseteq E$. A subgraph $H$ of $G$ is a proper subgraph of $G$ if $H \neq G$.

**Definition**

Let $G = (V, E)$ be a simple graph. The subgraph induced by a subset $W$ of the vertex set $V$ is the graph $(W, F)$, where the edge set $F$ contains an edge in $E$ if and only if both endpoints of this edge are in $W$.

# Subgraphs

Sometimes we work with only part of the graph. The smaller graph is called a subgraph .

**Definition**

A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$, where $W \subseteq V$ and $F \subseteq E$. A subgraph $H$ of $G$ is a proper subgraph of $G$ if $H \neq G$.

**Definition**

Let $G = (V, E)$ be a simple graph. The subgraph induced by a subset $W$ of the vertex set $V$ is the graph $(W, F)$, where the edge set $F$ contains an edge in $E$ if and only if both endpoints of this edge are in $W$.

Every subgraph of a bipartite graph is —.

# Subgraphs

Sometimes we work with only part of the graph. The smaller graph is called a subgraph .

**Definition**

A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$, where $W \subseteq V$ and $F \subseteq E$. A subgraph $H$ of $G$ is a proper subgraph of $G$ if $H \neq G$.

**Definition**

Let $G = (V, E)$ be a simple graph. The subgraph induced by a subset $W$ of the vertex set $V$ is the graph $(W, F)$, where the edge set $F$ contains an edge in $E$ if and only if both endpoints of this edge are in $W$.

Every subgraph of a bipartite graph is ——.

Exercise: **Every simple graph has a bipartite subgraph with at least $|E|/2$ edges.**

## Edge Contractions

- Some cases when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph – we perform an edge contraction .

# Edge Contractions

- Some cases when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph – we perform an edge contraction.
- We remove an edge $e$ with endpoints $u$ and $v$ and merge $u$ and $w$ into a new single vertex $w$, and for each edge with $u$ or $v$ as an endpoint replaces the edge with one with $w$ as endpoint in place of $u$ or $v$ and with the same second endpoint.

# Edge Contractions

- Some cases when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph – we perform an edge contraction .
- We remove an edge $e$ with endpoints $u$ and $v$ and merge $u$ and $w$ into a new single vertex $w$, and for each edge with $u$ or $v$ as an endpoint replaces the edge with one with $w$ as endpoint in place of $u$ or $v$ and with the same second endpoint.
- For when we have a contraction of the edge $e$ with endpoints $u$ and $v$ what happens to $G = (V, E)$?

# Edge Contractions

- Some cases when we remove an edge from a graph, we do not want to retain the endpoints of this edge as separate vertices in the resulting subgraph – we perform an edge contraction .
- We remove an edge $e$ with endpoints $u$ and $v$ and merge $u$ and $w$ into a new single vertex $w$, and for each edge with $u$ or $v$ as an endpoint replaces the edge with one with $w$ as endpoint in place of $u$ or $v$ and with the same second endpoint.
- For when we have a contraction of the edge $e$ with endpoints $u$ and $v$ what happens to $G = (V, E)$?
- We have a new graph $G^{'} = (V^{'}, E^{'})$ ( not a subgraph of $G$), where $V^{'} = V \setminus \{u, v\} \cup \{w\}$ and $E^{'}$ contains the edges in $E$ which do not have either $u$ or $v$ as endpoints and an edge connecting $w$ to every neighbor of either $u$ or $v$.

# Removing Vertices from a Graph & Union of Graphs

- When we remove a vertex $v$ and all edges incident to it from $G = (V, E)$, we produce a subgraph, denoted by $G - v$.
- $G - v = (V - v, E^{'})$, where $E^{'}$ is the set of edges of $G$ not incident to $v$.

# Removing Vertices from a Graph & Union of Graphs

- When we remove a vertex $v$ and all edges incident to it from $G = (V, E)$, we produce a subgraph, denoted by $G - v$.
- $G - v = (V - v, E^{'})$, where $E^{'}$ is the set of edges of $G$ not incident to $v$.
- The union of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with the vertext set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of $G_1$ and $G_2$ is denoted by $G_1 \cup G_2$.

# Graph Isomorphism

# Isomorphism of Graphs

- We often need to know whether if two graphs are essentially the same graph/structure if we ignore the identities of their vertices.
- It has applications in chemistry, for example where the same formula may represent different compounds.

# Isomorphism of Graphs

- We often need to know whether if two graphs are essentially the same graph/structure if we ignore the identities of their vertices.
- It has applications in chemistry, for example where the same formula may represent different compounds.

**Definition**

The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a one- to-one and onto function f from $V_1$ to $V_2$ with the property that $a$ and $b$ are adjacent in $G_1$ if and only if $f(a)$ and $f(b)$ are adjacent in $G_2$, for all $a$ and $b$ in $V_1$. Such a function $f$ is called an isomorphism.

# Isomorphism of Graphs

- We often need to know whether if two graphs are essentially the same graph/structure if we ignore the identities of their vertices.
- It has applications in chemistry, for example where the same formula may represent different compounds.

**Definition**

The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a one- to-one and onto function f from $V_1$ to $V_2$ with the property that $a$ and $b$ are adjacent in $G_1$ if and only if $f(a)$ and $f(b)$ are adjacent in $G_2$, for all $a$ and $b$ in $V_1$. Such a function $f$ is called an isomorphism.

Example:

# Determining if two graphs are isomorphic

- Not that easy to determine if two simple graphs are isomorphic.
- There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with $n$ vertices.

## Determining if two graphs are isomorphic

- Not that easy to determine if two simple graphs are isomorphic.
- There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with $n$ vertices. If $n$ is large, testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical!
- Sometimes it is not hard – we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism.

## Determining if two graphs are isomorphic

- Not that easy to determine if two simple graphs are isomorphic.
- There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with $n$ vertices. If $n$ is large, testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical!
- Sometimes it is not hard – we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism.
- Graph Invariant – a property preserved by isomorphism of graphs.

## Determining if two graphs are isomorphic

- Not that easy to determine if two simple graphs are isomorphic.
- There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with $n$ vertices. If $n$ is large, testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical!
- Sometimes it is not hard – we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism.
- Graph Invariant – a property preserved by isomorphism of graphs.
- Examples –

# Determining if two graphs are isomorphic

- Not that easy to determine if two simple graphs are isomorphic.
- There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with $n$ vertices. If $n$ is large, testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical!
- Sometimes it is not hard – we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism.
- Graph Invariant – a property preserved by isomorphism of graphs.
- Examples –
  - Must have the same number of vertices.
  - Must have the same number of edges.
  - Must have the same degrees of the vertices.

# Determining if two graphs are isomorphic

- The adjacency matrix of $G$ is the same as the adjacency matrix of $H$ , when rows and columns are labeled to correspond to the images under a bijective/isomorphism $f$ of the vertices in $G$.

## Determining if two graphs are isomorphic

- The adjacency matrix of $G$ is the same as the adjacency matrix of $H$ , when rows and columns are labeled to correspond to the images under a bijective/isomorphism $f$ of the vertices in $G$.
- Consider the example:

# Determining if two graphs are isomorphic

- The adjacency matrix of $G$ is the same as the adjacency matrix of $H$ , when rows and columns are labeled to correspond to the images under a bijective/isomorphism $f$ of the vertices in $G$.
- Consider the example:
- Note that if $f$ turned out not to be an isomorphism, we would not have established that $G$ and $H$ are not isomorphic, because another correspondence of the vertices in $G$ and $H$ may be an isomorphism.

## Determining if two graphs are isomorphic

- The adjacency matrix of $G$ is the same as the adjacency matrix of $H$ , when rows and columns are labeled to correspond to the images under a bijective/isomorphism $f$ of the vertices in $G$.
- Consider the example:
- Note that if $f$ turned out not to be an isomorphism, we would not have established that $G$ and $H$ are not isomorphic, because another correspondence of the vertices in $G$ and $H$ may be an isomorphism.
- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).

# Determining if two graphs are isomorphic

- The adjacency matrix of $G$ is the same as the adjacency matrix of $H$ , when rows and columns are labeled to correspond to the images under a bijective/isomorphism $f$ of the vertices in $G$.
- Consider the example:
- Note that if $f$ turned out not to be an isomorphism, we would not have established that $G$ and $H$ are not isomorphic, because another correspondence of the vertices in $G$ and $H$ may be an isomorphism.
- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).
- Linear average-case time complexity algorithms are known that solve this problem.

# Connectivity

# Moving from $a$ to $b$ in a graph

Path – a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. It visits the vertices along this path, i.e. the endpoints of these edges.

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an undirected graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that $e_i$ has for $i = 1, \ldots, n$, the endpoints $x_{i-1}$ and $x_i$ .

# Moving from $a$ to $b$ in a graph

Path – a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. It visits the vertices along this path, i.e. the endpoints of these edges.

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an undirected graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that $e_i$ has for $i = 1, \ldots, n$, the endpoints $x_{i-1}$ and $x_i$ .

- For simple graph, a path is given by its vertex sequence as it uniquely determines the path.

# Moving from $a$ to $b$ in a graph

Path – a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. It visits the vertices along this path, i.e. the endpoints of these edges.

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an undirected graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that $e_i$ has for $i = 1, \ldots, n$, the endpoints $x_{i-1}$ and $x_i$ .

- For simple graph, a path is given by its vertex sequence as it uniquely determines the path.
- The path is a circuit if it begins and ends at the same vertex, that is, if $u = v$, and has length greater than zero.

# Moving from $a$ to $b$ in a graph

Path – a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. It visits the vertices along this path, i.e. the endpoints of these edges.

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an undirected graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that $e_i$ has for $i = 1, \ldots, n$, the endpoints $x_{i-1}$ and $x_i$ .

- For simple graph, a path is given by its vertex sequence as it uniquely determines the path.
- The path is a circuit if it begins and ends at the same vertex, that is, if $u = v$, and has length greater than zero.
- A path or circuit is simple if it does not contain the same edge more than once.

# Moving from $a$ to $b$ in a (directed )graph

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an directed graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ such that $e_i$ is associated with $(x_0, x_1)$, $e_2$ with $(x_1, x_2)$ etc, where $x_0 = u$ and $x_n = v$. For a simple directed graph, we denote the path as a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices.

# Moving from $a$ to $b$ in a (directed )graph

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an directed graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ such that $e_i$ is associated with $(x_0, x_1)$, $e_2$ with $(x_1, x_2)$ etc, where $x_0 = u$ and $x_n = v$. For a simple directed graph, we denote the path as a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices.

- A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle.

# Moving from $a$ to $b$ in a (directed )graph

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an directed graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ such that $e_i$ is associated with $(x_0, x_1)$, $e_2$ with $(x_1, x_2)$ etc, where $x_0 = u$ and $x_n = v$. For a simple directed graph, we denote the path as a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices.

- A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle.
- A path or circuit is called simple if it does not contain the same edge more than once.
- Walk is also used instead of path, closed walk for circuit and trail for simple path.

# Moving from $a$ to $b$ in a (directed )graph

**Definition**

Let $n \in \mathbb{Z}_{\geq 0}$ and $G$ an directed graph. A path of length $n$ from $u$ to $v$ in $G$ is a sequence of $n$ edges $e_1, \ldots, e_n$ of $G$ such that $e_i$ is associated with $(x_0, x_1)$, $e_2$ with $(x_1, x_2)$ etc, where $x_0 = u$ and $x_n = v$. For a simple directed graph, we denote the path as a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices.

- A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle.
- A path or circuit is called simple if it does not contain the same edge more than once.
- Walk is also used instead of path, closed walk for circuit and trail for simple path.
- Important examples of paths – Erdos number, the length of the shortest path between a person and Paul Erdos.

# Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.

# Path and Cycles

- A path $P_n$ of length $n - 1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.

## Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.

# Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.

## Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
    - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
    - Therefore to reach where you started you need odd steps.

# Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
  - Therefore to reach where you started you need odd steps.
  - Conversely, every cycle of $G$ is even. For a connected component, fix a vertex in it : $v_0$.

# Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
  - Therefore to reach where you started you need odd steps.
  - Conversely, every cycle of $G$ is even. For a connected component, fix a vertex in it : $v_0$.
  - For each vertex $v$ in the same component color it red : if its shortest path from $v_0$ is even, else blue.

# Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
  - Therefore to reach where you started you need odd steps.
  - Conversely, every cycle of $G$ is even. For a connected component, fix a vertex in it : $v_0$.
  - For each vertex $v$ in the same component color it red : if its shortest path from $v_0$ is even, else blue.
  - Do the same for all components.

# Path and Cycles

- A path $P_n$ of length $n - 1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
  - Therefore to reach where you started you need odd steps.
  - Conversely, every cycle of $G$ is even. For a connected component, fix a vertex in it : $v_0$.
  - For each vertex $v$ in the same component color it red : if its shortest path from $v_0$ is even, else blue.
  - Do the same for all components.
  - Verify : $G$ has an edge with same colours only if it had an odd cycle.

## Path and Cycles

- A path $P_n$ of length $n-1$ is bipartite.
- A cycle $C_n$ of length $n$ is bipartite iff $n$ is even.
- A graph is bipartite iff it contains no odd cycles.
  - Proof: If bipartite with $(V_1, V_2)$ as bipartition, then every step of a path takes you from $V_1$ to $V_2$ or vice-versa.
  - Therefore to reach where you started you need odd steps.
  - Conversely, every cycle of $G$ is even. For a connected component, fix a vertex in it : $v_0$.
  - For each vertex $v$ in the same component color it red : if its shortest path from $v_0$ is even, else blue.
  - Do the same for all components.
  - Verify : $G$ has an edge with same colours only if it had an odd cycle.
- Collect the characterizations of bipartite graphs!

# Connectivity

- Interested in questions related to : is there a path between two points in a computer network so as to sent messages through intermediate computers.

# Connectivity

- Interested in questions related to : is there a path between two points in a computer network so as to sent messages through intermediate computers.
- In an undirected graph, connected if there is a path between every pair of distinct vertices of the graph.

**Theorem**

*There is a simple path between every pair of distinct vertices of a connected undirected graph.*

- Let $u$ and $v$ be two distinct vertices of the connected undirected graph $G = (V, E)$.

# Connectivity

- Interested in questions related to : is there a path between two points in a computer network so as to sent messages through intermediate computers.
- In an undirected graph, connected if there is a path between every pair of distinct vertices of the graph.

**Theorem**

*There is a simple path between every pair of distinct vertices of a connected undirected graph.*

- Let $u$ and $v$ be two distinct vertices of the connected undirected graph $G = (V, E)$.
- $G$ is connected,there is atleast one path between $u$ and $v$.

# Connectivity

- Interested in questions related to : is there a path between two points in a computer network so as to sent messages through intermediate computers.
- In an undirected graph, connected if there is a path between every pair of distinct vertices of the graph.

**Theorem**

*There is a simple path between every pair of distinct vertices of a connected undirected graph.*

- Let $u$ and $v$ be two distinct vertices of the connected undirected graph $G = (V, E)$.
- $G$ is connected,there is atleast one path between $u$ and $v$.
- Let $x_0 = u$,$x_1, \ldots, x_n = v$ be the vertex sequence of a path of least length.

# Connectivity

- Claim : This path of least length is simple. If not, then $x_i = x_j$ for some $i$ and $j$ with $0 \leq i \lneq j$.

# Connectivity

- Claim : This path of least length is simple. If not, then $x_i = x_j$ for some $i$ and $j$ with $0 \leq i \lneq j$.
- But then there is a path from $u$ to $v$ of shorter length – $x_0, x_1, ..., x_{i1}, x_j, \ldots, x_n$.

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.
- i.e. a maximal connected subgraph of $G$.

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.
- i.e. a maximal connected subgraph of $G$.
- A graph $G$ that is not connected has two or more connected components that are disjoint and have $G$ as their union.

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.
- i.e. a maximal connected subgraph of $G$.
- A graph $G$ that is not connected has two or more connected components that are disjoint and have $G$ as their union.
- Example:

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.
- i.e. a maximal connected subgraph of $G$.
- A graph $G$ that is not connected has two or more connected components that are disjoint and have $G$ as their union.
- Example:
- Another way to look at it:
    - Let $\sim$ be the relation on $V$ s.t $v \sim w$ iff $v$ and $w$ are connected by a path. Then $\sim$ is an equivalence relation on $V$. (Prove!)

# Connected Components

- A connected component of a graph $G$ is a connected subgraph of $G$ that is not a proper subgraph of another connected subgraph of $G$.
- i.e. a maximal connected subgraph of $G$.
- A graph $G$ that is not connected has two or more connected components that are disjoint and have $G$ as their union.
- Example:
- Another way to look at it:
  - Let $\sim$ be the relation on $V$ s.t $v \sim w$ iff $v$ and $w$ are connected by a path. Then $\sim$ is an equivalence relation on $V$. (Prove!)
  - Let $V^{'}$ be an equivalence class of the relation $\sim$ on $V$. The subgraph induced by $V^{'}$ is called a connected component of the graph.

# Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)

# Vertex Connectivity

- Vertex Cut : A subset $V'$ of $V$ in $G$ if $G - V'$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.

# Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.
- For a complete graph, removing any subset of its vertices and all incident edges still leaves a complete graph.

# Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.
- For a complete graph, removing any subset of its vertices and all incident edges still leaves a complete graph.
- We set $\kappa(K_n) = n - 1$, the number of vertices removed to give a graph of one vertex.

# Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.
- For a complete graph, removing any subset of its vertices and all incident edges still leaves a complete graph.
- We set $\kappa(K_n) = n - 1$, the number of vertices removed to give a graph of one vertex.
- We have $0 \leq \kappa(G) \leq n - 1$ if $G$ has $n$ vertices, $\kappa(G) = 0$ iff $G$ is disconnected or $G = K_1$ and $n - 1$ iff $G$ is complete (Prove!).

# Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.
- For a complete graph, removing any subset of its vertices and all incident edges still leaves a complete graph.
- We set $\kappa(K_n) = n - 1$, the number of vertices removed to give a graph of one vertex.
- We have $0 \leq \kappa(G) \leq n - 1$ if $G$ has $n$ vertices, $\kappa(G) = 0$ iff $G$ is disconnected or $G = K_1$ and $n - 1$ iff $G$ is complete (Prove!).
- The larger $\kappa(G)$ the more connected $G$ is.

## Vertex Connectivity

- Vertex Cut : A subset $V^{'}$ of $V$ in $G$ if $G - V^{'}$ is disconnected. (defined for noncomplete graphs!)
- Vertex connectivity, $\kappa(G)$ : minimum number of vertices in a vertex cut.
- For a complete graph, removing any subset of its vertices and all incident edges still leaves a complete graph.
- We set $\kappa(K_n) = n - 1$, the number of vertices removed to give a graph of one vertex.
- We have $0 \leq \kappa(G) \leq n - 1$ if $G$ has $n$ vertices, $\kappa(G) = 0$ iff $G$ is disconnected or $G = K_1$ and $n - 1$ iff $G$ is complete (Prove!).
- The larger $\kappa(G)$ the more connected $G$ is.
- We say that the graph is $k$-connected if $\kappa(G) \geq k$.

# Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.

# Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.
- Edge connectivity, $\lambda(G)$ – the minimum number of edges in an edge cut of $G$.

# Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.
- Edge connectivity, $\lambda(G)$ – the minimum number of edges in an edge cut of $G$.
- It is zero if not connected. (Or if $G$ contains a single vertex).

## Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.
- Edge connectivity, $\lambda(G)$ – the minimum number of edges in an edge cut of $G$.
- It is zero if not connected. (Or if $G$ contains a single vertex).
- Ex: $0 \leq \lambda(G) \leq n - 1$ and $\lambda(G) = n - 1$ iff $G = K_n$

## Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.
- Edge connectivity, $\lambda(G)$ – the minimum number of edges in an edge cut of $G$.
- It is zero if not connected. (Or if $G$ contains a single vertex).
- Ex: $0 \leq \lambda(G) \leq n-1$ and $\lambda(G) = n-1$ iff $G = K_n$
- Ex: When $G = (V, E)$ is a noncomplete connected graph with at least three vertices, $\kappa(G) \leq \min_{v \in V} \deg(v)$ and $\lambda(G) \leq \min_{v \in V} \deg(v)$

# Edge Connectivity

- Edge Cut – The minimum number of edges that we can remove to disconnect it, maybe its not one but a set of edges we need to disconnect $G$.
- Edge connectivity, $\lambda(G)$ – the minimum number of edges in an edge cut of $G$.
- It is zero if not connected. (Or if $G$ contains a single vertex).
- Ex: $0 \leq \lambda(G) \leq n-1$ and $\lambda(G) = n-1$ iff $G = K_n$
- Ex: When $G = (V, E)$ is a noncomplete connected graph with at least three vertices, $\kappa(G) \leq \min_{v \in V} \deg(v)$ and $\lambda(G) \leq \min_{v \in V} \deg(v)$
- For all graphs $\kappa(G) \leq \lambda(G) \leq \min_{v \in V} \deg(v)$