

# CS473-Algorithms I

## Lecture 13-A

### Graphs

# Graphs

---

A **directed graph** (or **digraph**)  $G$  is a pair  $(V, E)$ , where

$V$  is a finite set, and

$E$  is a binary relation on  $V$

The set  $V$ : Vertex set of  $G$

The set  $E$ : Edge set of  $G$

Note that, **self-loops** -edges from a vertex to itself- are possible

In an **undirected graph**  $G=(V, E)$

- the edge set  $E$  consists of **unordered pairs of vertices** rather than ordered pairs, that is,  $(u, v)$  &  $(v, u)$  denote the same edge
- **self-loops are forbidden**, so every edge consists of two distinct vertices

# Graphs

---

Many definitions for directed and undirected graphs are the same although certain terms have slightly different meanings

If  $(u, v) \in E$  in a **directed graph**  $G=(V, E)$ , we say that  $(u, v)$  is **incident from** or **leaves** vertex  $u$  and is **incident to** or **enters** vertex  $v$

If  $(u, v) \in E$  in an **undirected graph**  $G=(V, E)$ , we say that  $(u, v)$  is **incident on** vertices  $u$  and  $v$

If  $(u, v)$  is an edge in a graph  $G=(V, E)$ , we say that vertex  $v$  is **adjacent to** vertex  $u$

When the graph is **undirected**, the **adjacency relation** is **symmetric**

When the graph is **directed** the **adjacency relation** is **not necessarily symmetric**  
if  $v$  is adjacent to  $u$ , we sometimes write  $u \rightarrow v$

# Graphs

---

The **degree** of a vertex in an **undirected graph** is the number of edges **incident on it**

In a **directed graph**,

**out-degree** of a vertex: number of edges **leaving it**

**in-degree** of a vertex : number of edges **entering it**

**degree** of a vertex : its **in-degree** + its **out-degree**

A **path** of **length**  $k$  from a vertex  $u$  to a vertex  $u'$  in a graph

$G=(V, E)$  is a **sequence**  $\langle v_0, v_1, v_2, \dots, v_k \rangle$  of vertices such that

$v_0=u, v_k=u'$  and  $(v_{i-1}, v_i) \in E$ , for  $i=1, 2, \dots, k$

The **length** of a **path** is the **number of edges** in the path

# Graphs

---

If there is a path  $p$  from  $u$  to  $u'$ , we say that

$u'$  is **reachable** from  $u$  via  $p$ :  $u \xrightarrow{p} u'$

A **path is simple** if all vertices in the path are **distinct**

A **subpath** of path  $p = \langle v_0, v_1, v_2, \dots, v_k \rangle$  is a **contiguous subsequence** of its vertices

That is, for any  $0 \leq i \leq j \leq k$ , the subsequence of vertices

$\langle v_i, v_{i+1}, \dots, v_j \rangle$  is a **subpath** of  $p$

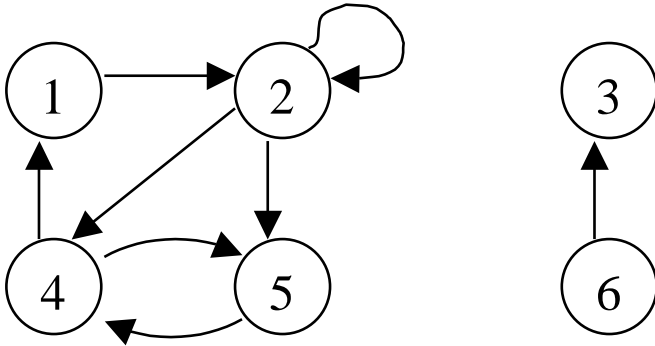
In a **directed graph**, a path  $\langle v_0, v_1, \dots, v_k \rangle$  forms a **cycle** if  $v_0 = v_k$  and the path contains at least one edge

The **cycle** is **simple** if, in addition,  $v_0, v_1, \dots, v_k$  are **distinct**

A **self-loop** is a **cycle** of **length 1**

# Graphs

Two paths  $\langle v_0, v_1, v_2, \dots, v_k \rangle$  &  $\langle v_0', v_1', v_2', \dots, v_k' \rangle$  form the **same cycle** if there is an integer  $j$  such that  $v_i' = v_{(i+j) \bmod k}$  for  $i = 0, 1, \dots, k-1$



The path  $p_1 = \langle 1, 2, 4, 1 \rangle$  forms the same cycles as the paths  $p_2 = \langle 2, 4, 1, 2 \rangle$  and  $p_3 = \langle 4, 1, 2, 4 \rangle$

A **directed graph** with **no self-loops** is **simple**

In an **undirected graph** a path  $\langle v_0, v_1, \dots, v_k \rangle$  forms a **cycle** if  $v_0 = v_k$  and  $v_1, v_2, \dots, v_k$  are **distinct**

A **graph** with **no cycles** is **acyclic**

# Graphs

---

An **undirected graph** is **connected**  
if every pair of vertices is **connected** by a **path**

The **connected components** of a graph are the  
**equivalence classes** of vertices under the  
“**is reachable from**” relation

An **undirected graph** is **connected** if it has exactly **one component**,  
i.e., if every vertex is reachable from every other vertex

A **directed graph** is **strongly-connected**  
if every two vertices are reachable from each other

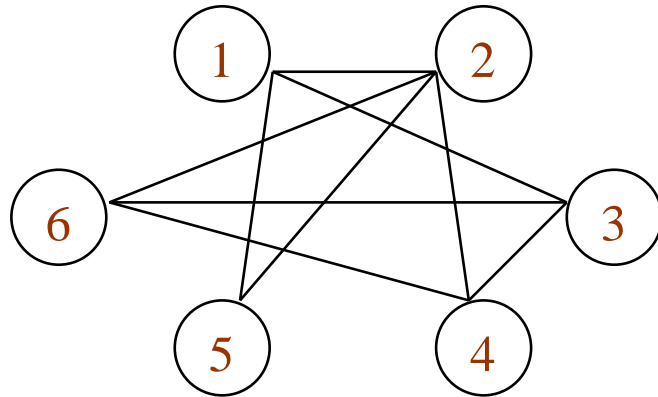
The **strongly-connected components** of a digraph are the  
**equivalence classes** of vertices under the  
“**are mutually reachable**” relation

A directed graph is **strongly-connected**  
if it has **only one strongly-connected component**

# Graphs

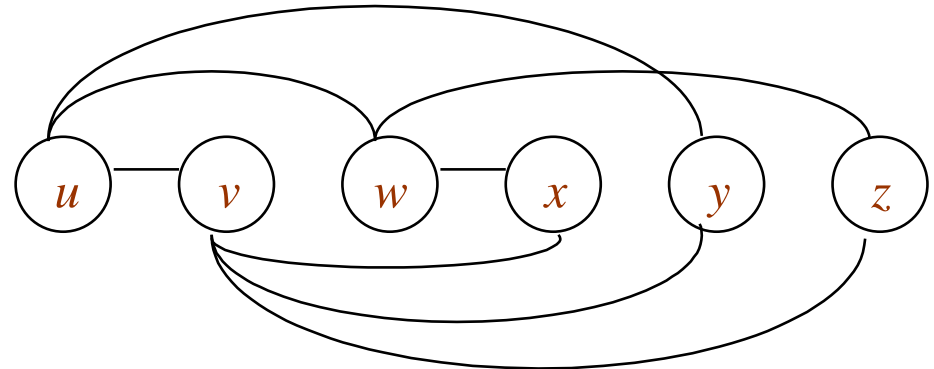
Two graphs  $G=(V, E)$  and  $G'=(V', E')$  are **isomorphic** if there exists a bijection  $f: V \rightarrow V'$  such that  $(u, v) \in E$  iff  $(f(u), f(v)) \in E'$

That is, we can relabel the vertices of  $G$  to be vertices of  $G'$  maintaining the corresponding edges in  $G$  and  $G'$



$G=(V, E)$

$V=\{1,2,3,4,5,6\}$



$G'=(V', E')$

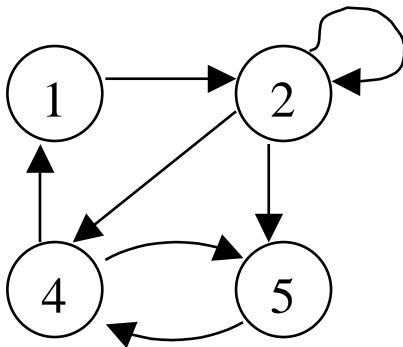
$V'=\{u,v,w,x,y,z\}$

Map from  $V \rightarrow V'$ :  $f(1)=u, f(2)=v, f(3)=w, f(4)=x, f(5)=y, f(6)=z$

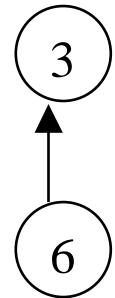
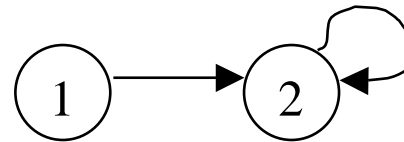
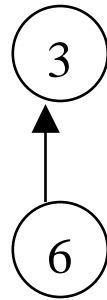
# Graphs

A graph  $G'=(V', E')$  is a **subgraph** of  $G=(V, E)$  if  
 $V' \subseteq V$  and  $E' \subseteq E$

Given a set  $V' \subseteq V$ , the subgraph of  $G$  **induced** by  $V'$  is the graph  
 $G'=(V', E')$  where  $E'=\{(u,v) \in E: u,v \in V'\}$



$G=(V, E)$



$G'=(V', E')$ , the subgraph of  $G$   
induced by the vertex set  
 $V'=\{1,2,3,6\}$

# Graphs

---

Given an **undirected** graph  $G=(V, E)$ , the **directed version** of  $G$  is the directed graph  $G'=(V', E')$ , where  
 $(u,v) \in E'$  and  $(v,u) \in E' \Leftrightarrow (u,v) \in E$

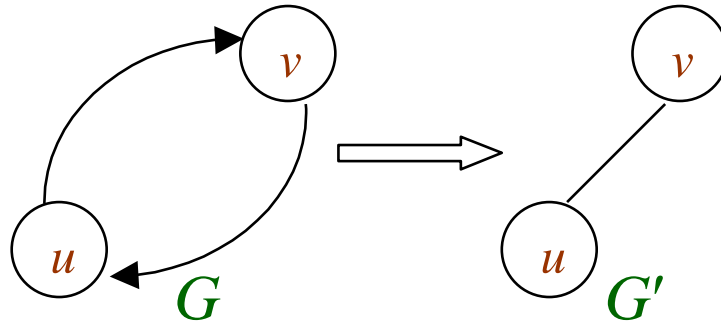
That is, each undirected edge  $(u,v)$  in  $G$  is replaced in  $G'$  by two directed edges  $(u,v)$  and  $(v,u)$

Given a **directed** graph  $G=(V, E)$ , the **undirected version** of  $G$  is the undirected graph  $G'=(V', E')$ , where  
 $(u,v) \in E' \Leftrightarrow u \neq v$  and  $(u,v) \in E$

That is the undirected version contains the edges of  $G$   
“**with their directions removed**” and with **self-loops eliminated**

# Graphs

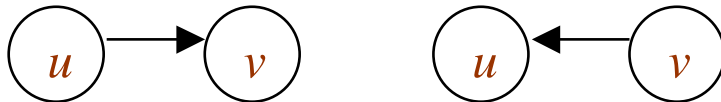
Note:



i.e.,  $(u,v)$  and  $(v,u)$  in  $G$  are replaced in  $G'$  by the same edge  $(u,v)$

In a **directed graph**  $G=(V, E)$ , a **neighbor** of a vertex  $u$  is any vertex that is **adjacent** to  $u$  in the **undirected version** of  $G$

That, is  $v$  is a neighbor of  $u$  **iff** either  $(u,v) \in E$  or  $(v,u) \in E$



$v$  is a neighbor of  $u$  in both cases

In an **undirected graph**,  $u$  and  $v$  are **neighbors** if they are **adjacent**

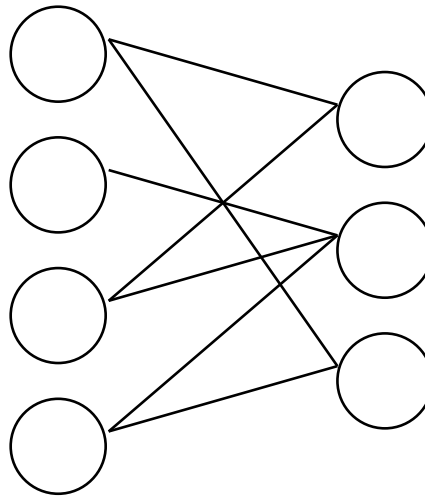
# Graphs

---

Several kinds of graphs are given special names

**Complete graph:** undirected graph in which every pair of vertices is **adjacent**

**Bipartite graph:** undirected graph  $G=(V, E)$  in which  $V$  can be **partitioned** into two disjoint sets  $V_1$  and  $V_2$  such that  $(u,v) \in E$  implies either  $u \in V_1$  and  $v \in V_2$  or  $u \in V_2$  and  $v \in V_1$



# Graphs

---

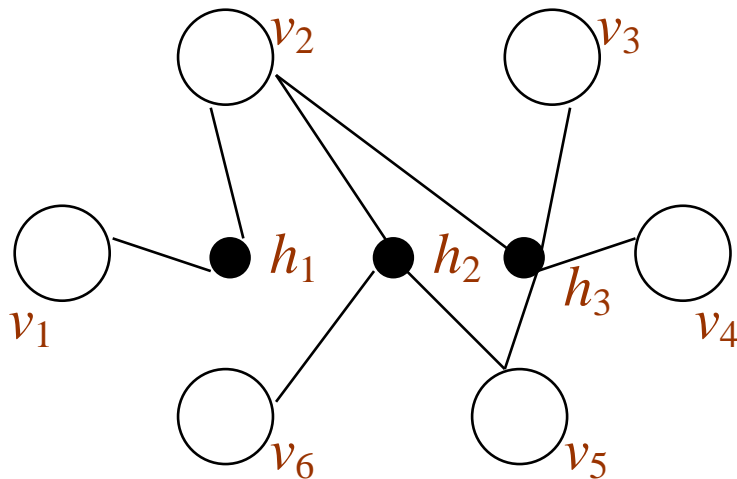
Forest: acyclic, undirected graph

Tree: connected, acyclic, undirected graph

Dag: directed acyclic graph

Multigraph: undirected graph with multiple edges between vertices and self-loops

Hypergraph: like an undirected graph, but each hyperedge, rather than connecting two vertices, connects an arbitrary subset of vertices



$$h_1 = (v_1, v_2)$$

$$h_2 = (v_2, v_5, v_6)$$

$$h_3 = (v_2, v_3, v_4, v_5)$$

# Free Trees

---

- A **free tree** is a **connected, acyclic, undirected** graph
- We often omit the adjective “free” when we say that a graph is a tree
- If an undirected graph is acyclic but possibly disconnected it is a **forest**

# Theorem (Properties of Free Trees)

---

The following are equivalent for an undirected graph  $G=(V,E)$

1.  $G$  is a free tree
2. Any two vertices in  $G$  are connected by a unique simple-path
3.  $G$  is connected, but if any edge is removed from  $E$  the resulting graph is disconnected
4.  $G$  is connected, and  $|E| = |V| - 1$
5.  $G$  is acyclic, and  $|E| = |V| - 1$
6.  $G$  is acyclic, but if any edge is added to  $E$ , the resulting graph contains a cycle

# Properties of Free Trees (1 $\Rightarrow$ 2)

---

- (1) **G** is a free tree
- (2) Any two vertices in **G** are connected by a unique simple-path

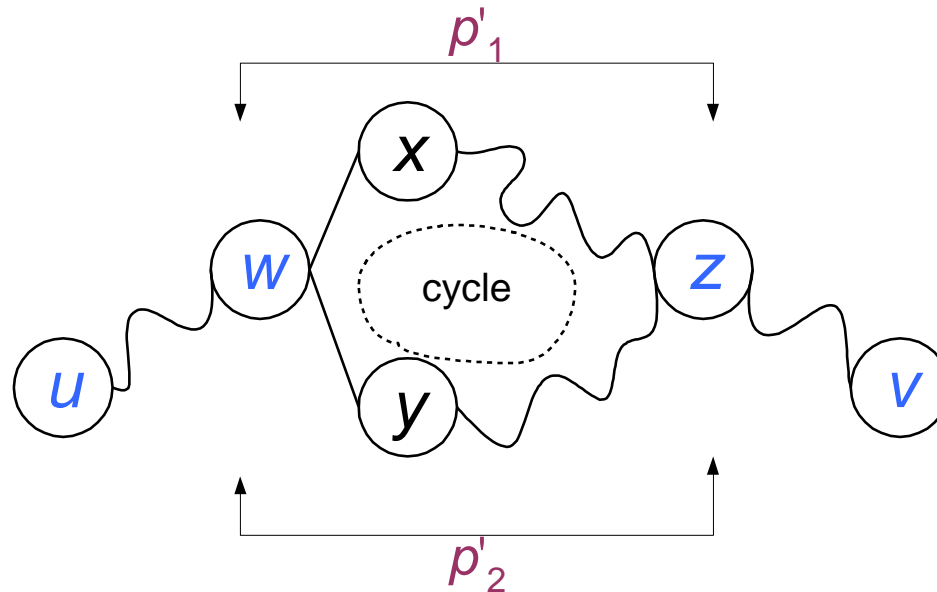
# Properties of Free Trees ( $1 \Rightarrow 2$ )

---

Since a tree is connected, any two vertices in  $G$  are connected by a simple path

- Let two vertices  $u, v \in V$  are connected by two simple paths  $p_1$  and  $p_2$
- Let  $w$  and  $z$  be the first vertices at which  $p_1$  and  $p_2$  diverge and re-converge
- Let  $p'_1$  be the subpath of  $p_1$  from  $w$  to  $z$
- Let  $p'_2$  be the subpath of  $p_2$  from  $w$  to  $z$
- $p'_1$  and  $p'_2$  share no vertices except their end points
- The path  $p'_1 \parallel p'_2$  is a cycle (contradiction)

# Properties of Free Trees ( $1 \Rightarrow 2$ )



- $p'_1$  and  $p'_2$  share no vertices except their end points
- $p'_1 \parallel p'_2$  is a cycle (contradiction)
- Thus, if  $G$  is a tree, there can be at most one path between two vertices

# Properties of Free Trees (2 $\Rightarrow$ 3)

---

- (2) Any two vertices in **G** are connected by a unique simple-path
- (3) **G** is connected, but if any edge is removed from **E** the resulting graph is disconnected

# Properties of Free Trees (2 $\Rightarrow$ 3)

---

If any two vertices in  $G$  are connected by a unique simple path, then  $G$  is connected

- Let  $(u,v)$  be any edge in  $E$ . This edge is a path from  $u$  to  $v$ . So it must be the unique path from  $u$  to  $v$
- Thus, if we remove  $(u,v)$  from  $G$ , there is no path from  $u$  to  $v$
- Hence, its removal disconnects  $G$

# Properties of Free Trees ( $3 \Rightarrow 4$ )

---

Before proving  $3 \Rightarrow 4$  consider the following

**Lemma:** any connected, undirected graph  $G=(V,E)$  satisfies  $|E| \geq |V|-1$

**Proof:** Consider a graph  $G'$  with  $|V|$  vertices and no edges. Thus initially there are  $|C|=|V|$  connected components

- Each isolated vertex is a connected component

Consider an edge  $(u,v)$  and let  $C_u$  and  $C_v$  denote the connected-components of  $u$  and  $v$

# Properties of Free Trees (Lemma)

---

If  $C_u \neq C_v$  then  $(u,v)$  connects  $C_u$  and  $C_v$  into a connected component  $C_{uv}$

Otherwise  $(u,v)$  adds an extra edge to the connected component  $C_u = C_v$

Hence, each edge added to the graph reduces the number of connected components by at most 1

Thus, at least  $|V|-1$  edges are required to reduce the number of components to 1 **Q.E.D**

# Properties of Free Trees (3 $\Rightarrow$ 4)

---

- (3) **G** is connected, but if any edge is removed from **E** the resulting graph is disconnected
- (4) **G** is connected, and  $|\mathbf{E}| = |\mathbf{V}| - 1$

# Properties of Free Trees (3 $\Rightarrow$ 4)

---

By assuming (3), the graph  $G$  is connected

We need to show both  $|E| \geq |V|-1$  and  $|E| \leq |V|-1$   
in order to show that  $|E| = |V|-1$

$|E| \geq |V|-1$ : valid due previous lemma

$|E| \leq |V|-1$ : (proof by induction)

Basis: a connected graph with  $n = 1$  or  $n = 2$   
vertices has  $n-1$  edges

IH: suppose that all graphs  $G' = (V', E')$   
satisfying (3) also satisfy  $|E'| \leq |V'|-1$

# Properties of Free Trees (3 $\Rightarrow$ 4)

---

Consider  $G=(V,E)$  that satisfies (3) with  $|V| = n \geq 3$

Removing an arbitrary edge  $(u,v)$  from  $G$  separates the graph into 2 connected graphs  $G_u=(V_u,E_u)$  and  $G_v=(V_v,E_v)$  such that  $V = V_u \cup V_v$  and  $E = E_u \cup E_v$

Hence, connected graphs  $G_u$  and  $G_v$  both satisfy (3) else  $G$  would not satisfy (3)

Note that  $|V_u|$  and  $|V_v| < n$  since  $|V_u| + |V_v| = n$

Hence,  $|E_u| \leq |V_u|-1$  and  $|E_v| \leq |V_v|-1$  (by IH)

Thus,  $|E| = |E_u| + |E_v| + 1 \leq (|V_u|-1) + (|V_v|-1) + 1$

$\Rightarrow |E| \leq |V|-1$

**Q.E.D**

# Properties of Free Trees (4 $\Rightarrow$ 5)

---

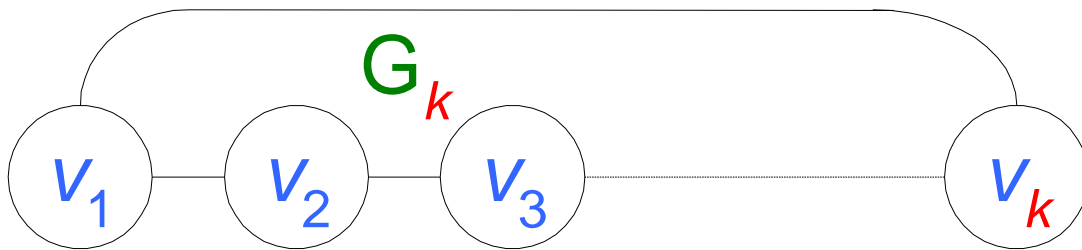
(4) **G** is connected, and  $|E| = |V| - 1$

(5) **G** is acyclic, and  $|E| = |V| - 1$

# Properties of Free Trees ( $4 \Rightarrow 5$ )

Suppose that  $G$  is connected, and  $|E| = |V| - 1$ , we must show that  $G$  is acyclic

- Suppose  $G$  has a cycle containing  $k$  vertices  $v_1, v_2, \dots, v_k$
- Let  $G_k = (V_k, E_k)$  be subgraph of  $G$  consisting of the cycle



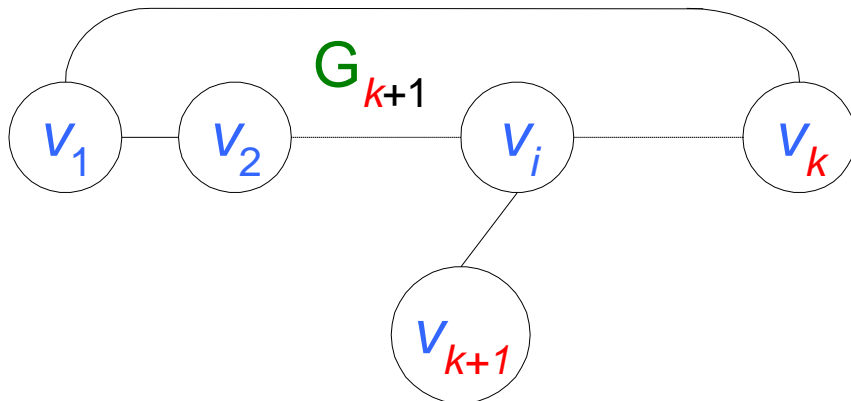
Note:  $|V_k| = |E_k| = k$

If  $k < |V|$ , there must be a vertex  $v_{k+1} \in V - V_k$  that is adjacent to some vertex  $v_i \in V_k$ , since  $G$  is connected

# Properties of Free Trees ( $4 \Rightarrow 5$ )

Define  $G_{k+1} = (V_{k+1}, E_{k+1})$  to be subgraph of  $G$  with

$$V_{k+1} = V_k \cup v_{k+1} \text{ and } E_{k+1} = E_k \cup (v_{k+1}, v_j)$$



Note:  $|V_{k+1}| = |E_{k+1}|$

If  $k+1 < |V|$ , we can similarly define  $G_{k+2} = (V_{k+2}, E_{k+2})$  to be the subgraph of  $G$  with  $V_{k+2} = V_{k+1} \cup v_{k+2}$  and  $E_{k+2} = E_{k+1} \cup (v_{k+2}, v_j)$  for some  $v_j \in V_{k+1}$  where  $|V_{k+2}| = |E_{k+2}|$

# Properties of Free Trees (4 $\Rightarrow$ 5)

---

We can continue defining  $G_{k+m}$  with  $|V_{k+m}| = |E_{k+m}|$  until we obtain  $G_n = (V_n, E_n)$  where

$$n = |V| \text{ and } |V_n| = |V| \text{ and } |V_n| = |E_n| = |V|$$

- Since  $G_n$  is a subgraph of  $G$ , we have

$E_n \subseteq E \Rightarrow |E| \geq |E_n| = |V|$  which contradicts the assumption  $|E| = |V| - 1$

Hence  $G$  is acyclic

Q.E.D

# Properties of Free Trees (5 $\Rightarrow$ 6)

---

- (5) **G** is acyclic, and  $|\mathbf{E}| = |\mathbf{V}| - 1$
- (6) **G** is acyclic, but if any edge is added to **E**, the resulting graph contains a cycle

# Properties of Free Trees (5 $\Rightarrow$ 6)

---

Suppose that  $G$  is acyclic and  $|E| = |V| - 1$

- Let  $k$  be the number of connected components of  $G$

$G_1=(V_1,E_1)$ ,  $G_2=(V_2,E_2), \dots, G_k=(V_k,E_k)$  such that

$$\bigcup_{i=1}^k V_i = V; \quad V_i \cap V_j = \emptyset; \quad 1 \leq i, j \leq k \text{ and } i \neq j$$

$$\bigcup_{i=1}^k E_i = E; \quad E_i \cap E_j = \emptyset; \quad 1 \leq i, j \leq k \text{ and } i \neq j$$

Each connected component  $G_i$  is a tree by definition

# Properties of Free Trees (5 $\Rightarrow$ 6)

---

Since (1 $\Rightarrow$ 5) each component  $G_i$  satisfies

$$|E_i| = |V_i| - 1 \quad \text{for } i = 1, 2, \dots, k$$

• Thus

$$\sum_{i=1}^k |E_i| = \sum_{i=1}^k |V_i| - \sum_{i=1}^k 1$$

$$|E| = |V| - k$$

• Therefore, we must have  $k = 1$

# Properties of Free Trees (5 $\Rightarrow$ 6)

---

That is (5)  $\Rightarrow$   $G$  is connected  $\Rightarrow$   $G$  is a tree

Since (1 $\Rightarrow$ 2)

any two vertices in  $G$  are connected by a unique simple path

Thus,

adding any edge to  $G$  creates a cycle

# Properties of Free Trees (6 $\Rightarrow$ 1)

---

- (6) **G** is acyclic, but if any edge is added to **E**, the resulting graph contains a cycle
- (1) **G** is a free tree

# Properties of Free Trees (6 $\Rightarrow$ 1)

---

Suppose that  $G$  is acyclic but if any edge is added to  $E$  a cycle is created

We must show that  $G$  is connected due to the definition

Let  $u$  and  $v$  be two arbitrary vertices in  $G$

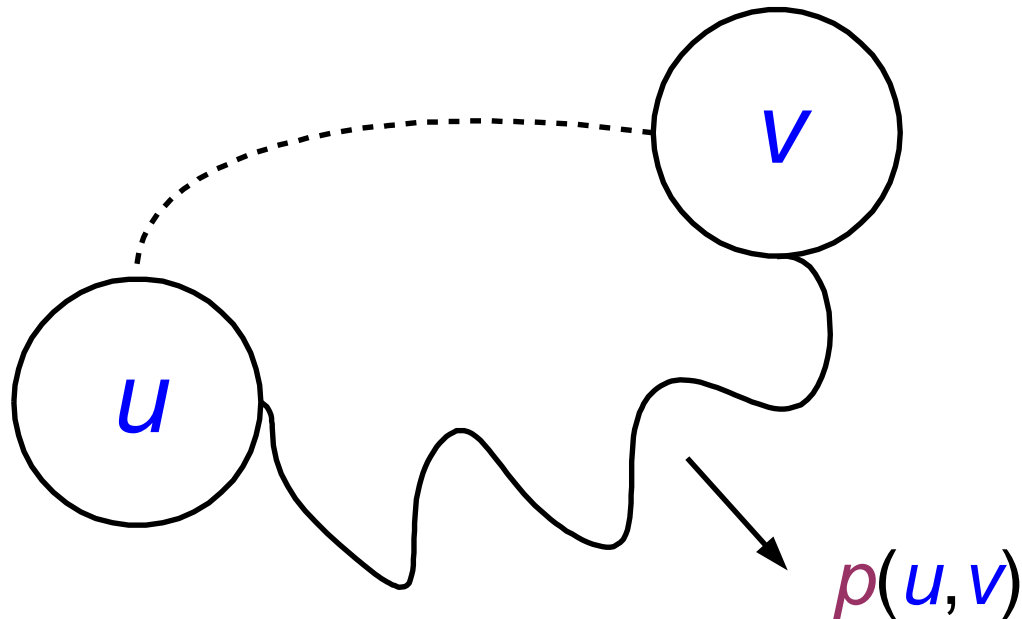
If  $u$  and  $v$  are not already adjacent

adding the edge  $(u,v)$  creates a cycle in which all edges but  $(u,v)$  belong to  $G$

# Properties of Free Trees ( $6 \Rightarrow 1$ )

---

Thus there is a path from  $u$  to  $v$ , and since  $u$  and  $v$  are chosen arbitrarily  $G$  is connected



# Representations of Graphs

---

- The standard two ways to represent a graph  $G=(V,E)$ 
  - As a collection of **adjacency-lists**
  - As an **adjacency-matrix**
- **Adjacency-list** representation is usually preferred
  - Provides a compact way to represent sparse graphs
    - Those graphs for which  $|E| \ll |V|^2$

# Representations of Graphs

---

- **Adjacency-matrix** representation may be preferred
  - for dense graphs for which  $|E|$  is close to  $|V|^2$
  - when we need to be able to tell quickly if there is an edge connecting two given vertices

# Adjacency-List Representation

---

- An array  $\text{Adj}$  of  $|V|$  lists, one for each vertex  $u \in V$
- For each  $u \in V$  the adjacency-list  $\text{Adj}[u]$  contains (pointers to) all vertices  $v$  such that  $(u, v) \in E$
- That is,  $\text{Adj}[u]$  consists of all vertices adjacent to  $u$  in  $G$
- The vertices in each adjacency-list are stored in an arbitrary order

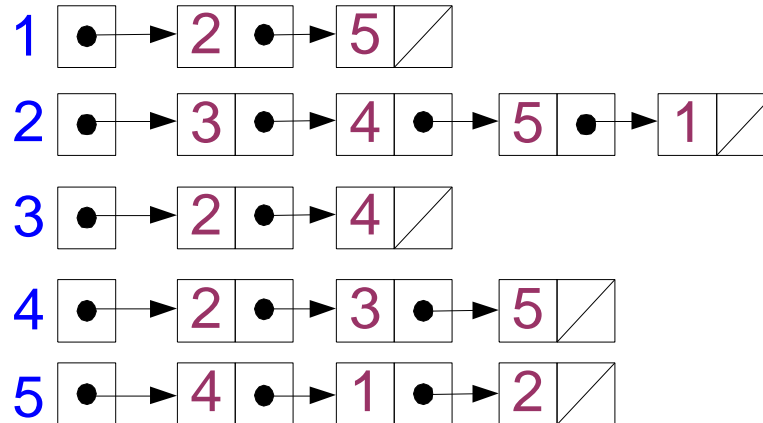
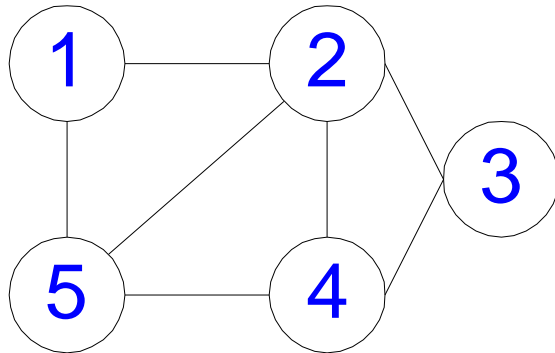
# Adjacency-List Representation

---

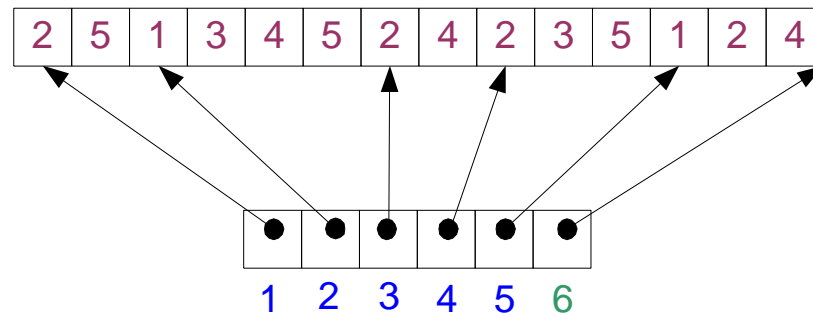
- If  $G$  is a directed graph
  - The sum of the lengths of the adjacency lists =  $|E|$
- If  $G$  is an undirected graph
  - The sum of the lengths of the adjacency lists =  $2|E|$   
since an edge  $(u,v)$  appears in both  $Adj[u]$  and  $Adj[v]$

# Representations of Graphs

## Undirected Graphs

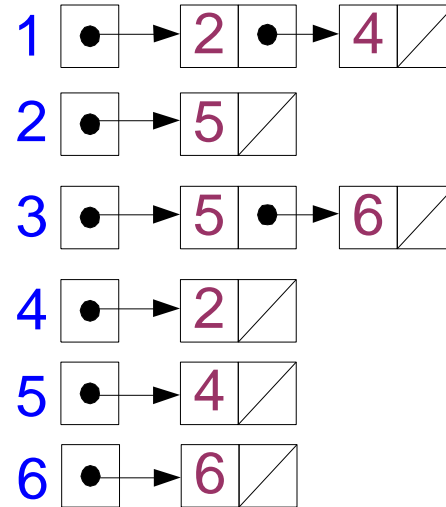
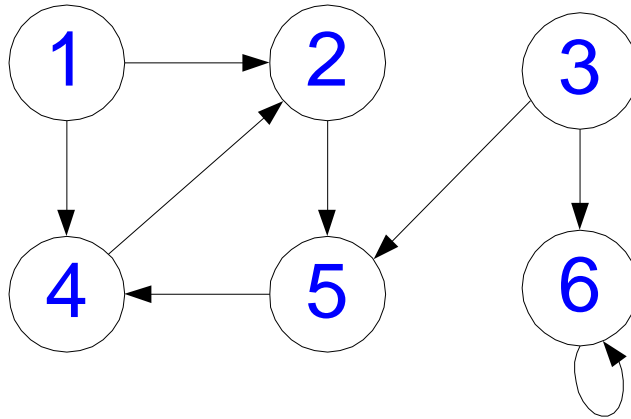


1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

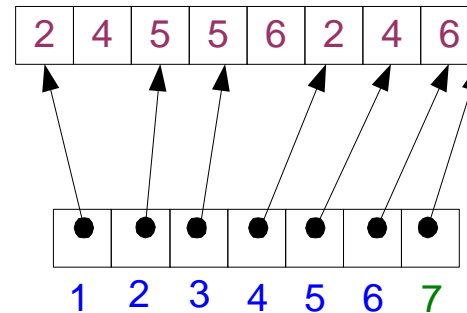


# Representations of Graphs

## Directed Graphs



1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1



# Adjacency List Representation (continued)

---

Adjacency list representation has the desirable property  
it requires  $O(\max(V, E)) = O(V+E)$  memory  
for both undirected and directed graphs

Adjacency lists can be adopted to represent **weighted graphs**  
each edge has an associated **weight** typically given by a  
**weight function**  $w: E \rightarrow R$

The weight  $w(u, v)$  of an edge  $(u, v) \in E$  is simply stored with  
vertex  $v$  in  $\text{Adj}[u]$  or with  
vertex  $u$  in  $\text{Adj}[v]$  or **both**

# Adjacency List Representation (continued)

---

A **potential disadvantage** of adjacency list representation  
there is no quicker way to determine if a given edge  $(u, v)$  is  
present in  $G$  than to search  $v$  in  $\text{Adj}[u]$  or  $u$  in  $\text{Adj}[v]$

This disadvantage can be remedied by an **adjacency matrix**  
representation at the cost of using asymptotically more  
memory

# Adjacency Matrix Representation

---

Assume that, the vertices of  $G=(V, E)$  are numbered as  $1,2,\dots,|V|$

Adjacency matrix rep. consists of a  $|V|\times|V|$  matrix  $A=(a_{ij}) \ni$

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Requires  $\Theta(V^2)$  memory independent of the number of edges  $|E|$

We define the transpose of a matrix  $A=(a_{ij})$  to be the matrix

$$A^T = (a_{ij})^T \text{ given by } a_{ij}^T = a_{ji}$$

Since in an undirected graph,  $(u,v)$  and  $(v,u)$  represent the same edge  $A = A^T$  for an undirected graph

That is, adjacency matrix of an undirected graph is symmetric

Hence, in some applications, only upper triangular part is stored

# Adjacency Matrix Representation

---

Adjacency matrix representation can also be used for weighted graphs

$$a_{ij} = \begin{cases} w(i, j) & \text{if } (i, j) \in E \\ \text{NIL or } 0 \text{ or } \infty & \text{otherwise} \end{cases}$$

Adjacency matrix may also be preferable for reasonably small graphs

Moreover, if the graph is unweighted rather than using one word of memory for each matrix entry adjacency matrix representation uses one bit per entry