Advanced Algorithms Analysis and Design

By

Nazir Ahmad Zafar

Lecture No 27

Huffman Coding Problem

and

Graph Theory

Lemma 2: Optimal Substructure Property

- Let C be a given alphabet with frequency $f[c]$ defined for each character $c \in C$.
- Let x, y (characters) $\in C$ with minimum frequency.
- Let C' be alphabet C with characters x, y removed, new character z added, so that $C' = C - \{x, y\} \cup \{z\}$;
- Define f for C' as for C, except that $f[z] = f[x] + f[y]$.
- Let T' be any tree representing an optimal prefix code for the alphabet C'.
- Then tree T, obtained from T' by replacing leaf node for z with an internal node having x, y as children, represents an optimal prefix code for alphabet C.

Proof

- *Since* $C' = C - \{x, y\} \cup \{z\}$; where $f(z) = f(x) + f(y)$,
- We are give that T' is an optimal tree for C'.
- Let T be tree obtained from T' by making x, y children of z.
- We prove following relationship in $B(T)$ and $B(T')$:

$$B(T) = B(T') + f(x) + f(y)$$

$$\begin{aligned}
B(T) &= \sum_{c \in C} f(c) d_T(c) \\
&= \sum_{c \in C \setminus \{x,y\}} f(c) d_T(c) + f(x) d_T(x) + f(y) d_T(y) \\
&= \sum_{c \in C' \setminus \{z\}} f(c) d_{T'}(c) + (f(x) + f(y))(d_{T'}(z) + 1) \\
&= \sum_{c \in C' \setminus \{z\}} f(c) d_{T'}(c) + f(z) d_{T'}(z) + f(x) + f(y) \\
&= \sum_{c \in C'} f(c) d_{T'}(c) + f(x) + f(y) \\
&= B(T') + f(x) + f(y)
\end{aligned}$$

Contd..

If T' is optimal for C', then T is optimal for C?

- Assume on contrary that there exists a better tree T'' for C, such that B(T'') < $B(T)$
- Assume without loss of generality T'' has siblings x and y.
- Let tree T''' be a tree T'' with the common parent of x and y replaced by vertex z with frequency $f[z] = f[x] + f[y]$. Then

$$B(T''') = B(T'') - f(x) - f(y)$$
$$< B(T) - f(x) - f(y) \qquad \text{Since, B(T'')} < B(T)$$
$$= B(T').$$

- This contradicts the optimality of $B(T')$.
- Hence, T must be optimal for C.

Road Trip Problem

Road Trip Problem

Problem Statement

- You purchase a new car. On your semester break, you decide to take a road trip from Peshawar to Karachi.
- Your car has a tank of some capacity such that only a distance k km can be traveled before refilling the tank.
- Suppose there are filling stations at distances of

  $$d_0 < d_1 < d_2 < \ldots < d_n$$

  where d_n is the total distance of your trip.
- Your goal is to find the smallest number of stops required i.e. shortest subsequence of $<d_0 \cdots d_n>$, given that you start at d_0 and end at d_n.

Road Trip Problem

INPUT:

- The max distance k, along with the distances: $d_0, d_1, \ldots, d_n$.

GOAL:

- To find a smallest sub sequence of $d_0, \ldots, d_n$ so that you can start from d_0 and end at d_n.

Note

- Greedy approach is considered each d_i in order.
- We stop to refuel at d_i only if the tank will finish before we get to d_{i+1}.

Road Trip Problem

Greedy algorithm

1. **for** i = 1 **to** n **do**
2. **if** $d_i - d_{i-1} > k$ **then** "do not use this car"
3. $S = d_0$
4. last = d_0 (the distance of the last item in S)
5. $d_{n+1} = \infty$ (forces d_n to be in S)
6. **for** i = 1 **to** n **do**
7. **if** d_{i+1} - last > k **then**
8. $S := S \cup d_i$
9. last := d_i

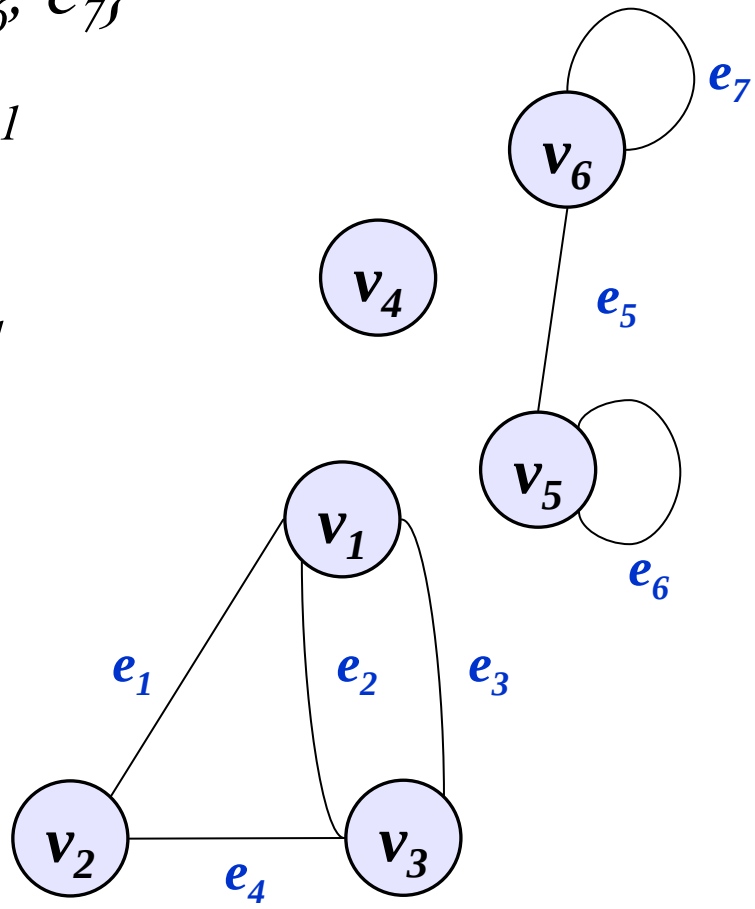
Graph Theoretic Concepts

Definitions

- **Graph** *G* consists of two finite sets *V(G)* and *E(G)*
- **Endpoints** a set of one or two vertices of an edge
- **Loop** an edge with just one endpoint
- **Edge-endpoint function:** End-Point-Function: E→ Set of V
- **Parallel edges** two distinct edges with same endpoints
- **Adjacent vertices** vertices connected by an edge
- **Vertex adjacent to itself** vertex endpoint of a loop
- **Adjacent edges** two edges incident on the same endpoint
- **Isolated** a vertex on which no edge is incident
- **Empty** a graph with no vertices, otherwise **nonempty**.

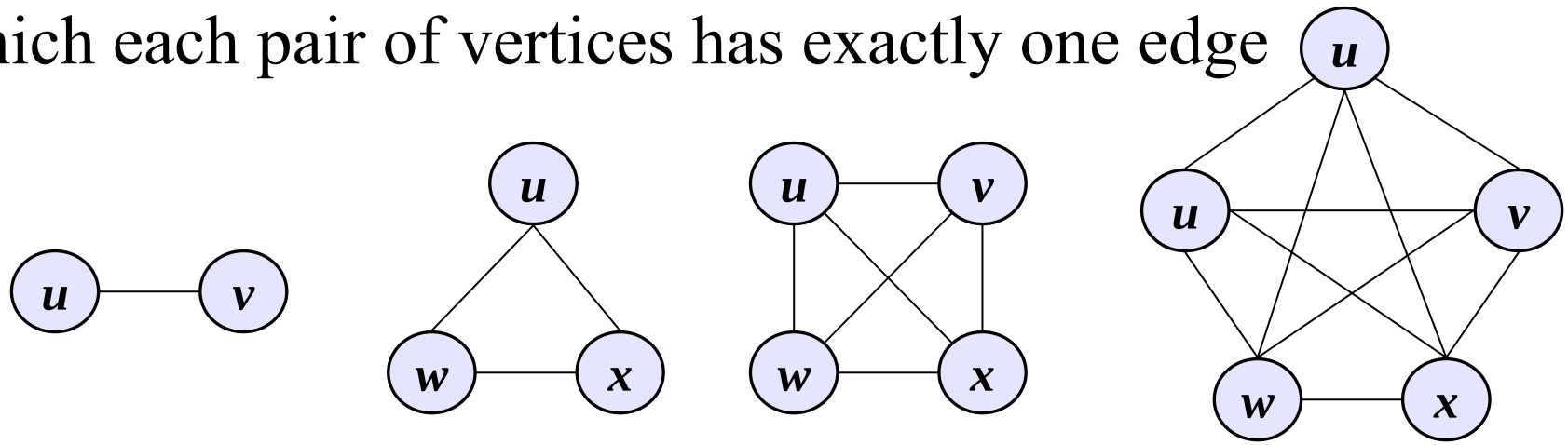
Examples

1. Vertex set = $\{v_1, v_2, v_3, v_4, v_5, v_6\}$
2. Edge set = $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$
3. e_1, e_2, and e_3 are incident on v_1
4. v_2 and v_3 are adjacent to v_1
5. e_2, e_3 and e_4 are adjacent to e_1
6. e_6 and e_7 are loops
7. e_2 and e_3 are parallel
8. v_5 and v_6 are adjacent to themselves
9. v_4 is an isolated vertex
10. $Endpoint(e_5) = (v_5, v_6)$



Directed, Simple and Complete Graph

- **Directed graph (digraph)** in which each edge is associated with an **ordered pairs** of vertices
- **Simple graph** does not have any loop or parallel edge
- **Subgraph** *H* is subgraph of *G* if and only if, every vertex in *H* is also vertex in *G*, and every edge in *H* has the same endpoints as in *G*.
- **Complete graph on *n* vertices**, (K_n) is a simple graph in which each pair of vertices has exactly one edge



Complete Graph

Example 1:

Find a recursive formula to compute number of edges in a **complete graph on *n* vertices**.

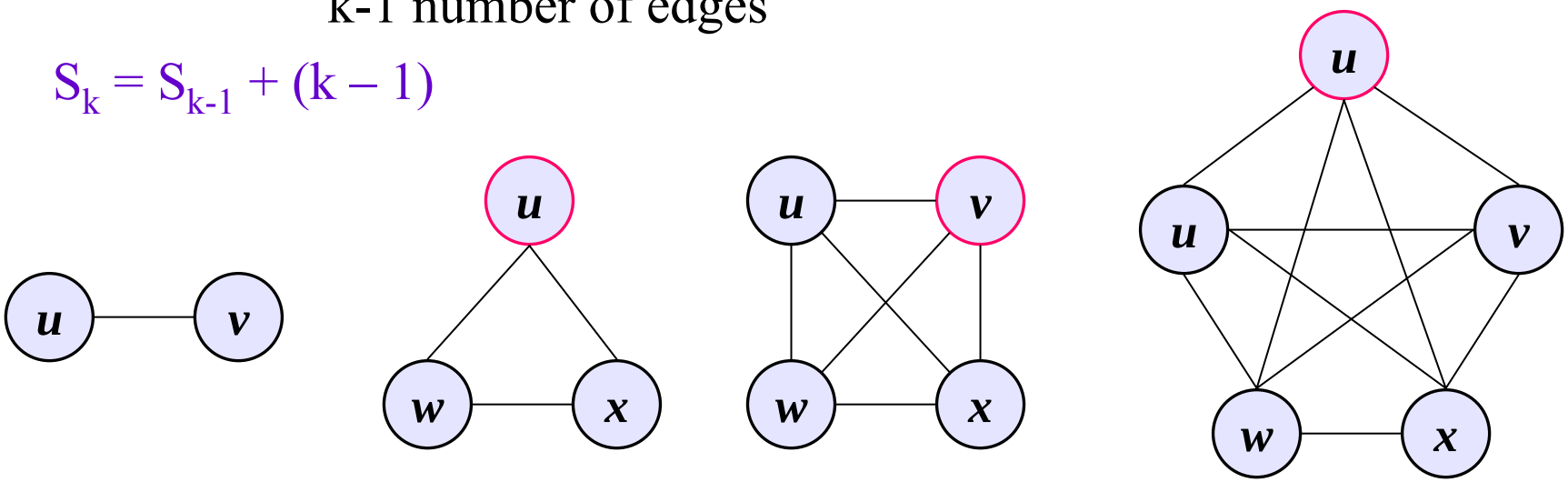
Solution:

Let S_k = total number of edges of a complete graph with k vertices

S_k = total number of edges of a complete graph with k - 1 vertices + total number of edges connected k^{th} node

= total number of edges of a complete graph with k - 1 vertices + k-1 number of edges

$S_k = S_{k-1} + (k - 1)$



Complete Graph

Example 2:

Find an **explicit** formula to compute number of edges in **complete graph on *n* vertices**.

Solution:

Since, $S_k = S_{k-1} + (k - 1) \Rightarrow S_{k-1} = S_{k-2} + (k - 2)$ and so on

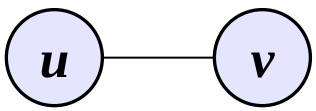
By back substitution

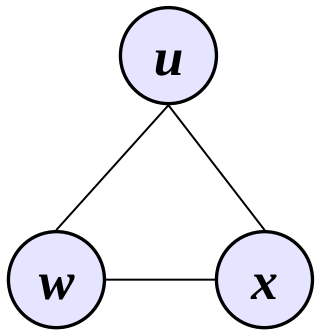
$S_k = S_{k-2} + (k - 2) + (k - 1) = S_{k-3} + (k - 3) + (k - 2) + (k - 1)$

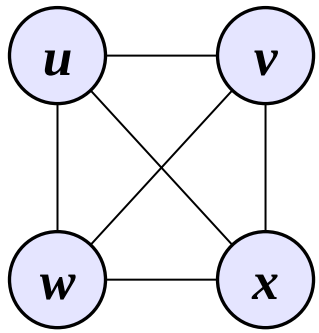
$S_k = S_1 + 1 + 2 + \ldots + (k - 2) + (k - 1) = (k-1)k/2$

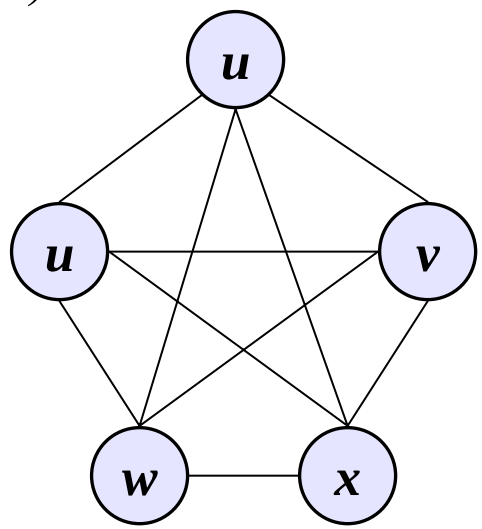
$S_k = (k-1)k/2$

$S_n = (n-1)n/2$







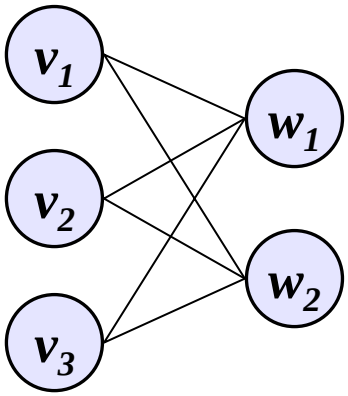


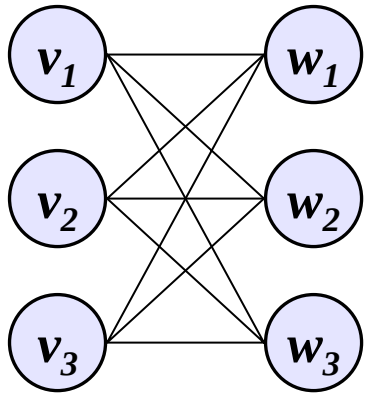
Complete Bipartite Graph

- A **complete bipartite** graph on (m, n) vertices, denoted $\boldsymbol{K_{m,\,n}}$, is a simple graph with vertices $v_1, v_2, \ldots, v_n$ and $w_1, w_2, \ldots, w_n$ that satisfies the following properties:

  for all $i, k = 1, 2, \ldots, m$ and for all $j, l = 1, 2, \ldots, n$

  1. there is an edge from each vertex v_i to each vertex w_j ;
  2. there is not an edge from any vertex v_i to any other vertex v_k ;
  3. there is not an edge from any vertex w_j to any other vertex w_l





Degree of Graph

Definition degree of v is number of edges incident on v

Theorem:

In a graph G, sum of degrees of all vertices equals twice the number of edges of G. Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$ where n is a positive integer, then

Total degree of $G = \deg(v_1) + deg(v_2) + \ldots + deg(v_n)$

$= 2$. (the number of edges of G)

Proof: easy

Note: Total degree of a graph, G, is even.

Degree of Graph

Proposition

In a graph there are even number of vertices of odd degree.

Solution:

Let us suppose that: V = all vertices of a graph

V1 = $\{v_1, v_2, \ldots, v_k\}$ = set of all vertices of odd degree

V2 = $\{w_1, w_2, \ldots, w_m\}$ = set of all vertices of even degree

Now we have to prove that k is even?

On contrary, suppose that k is odd

Degree (graph) = $\deg(v_1) + \deg(v_2) + , \ldots, \deg(v_k) + \deg(V2)$

= odd + even = odd, contradiction.

Hence k must be even.

That is there even number of vertices of odd degree.

Walk of Graph

- **Walk from *v* to *w*** is a finite alternating sequence of adjacent vertices and edges of G. It has the form

$$v = v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n = w,$$

for all $i = 1, 2, \ldots n$, v_{i-1} and v_i are endpoints of e_i.

- **Trivial walk from *v* to *v*** consists of single vertex v.
- **Closed walk** starts and ends at the same vertex
- **Path** a walk that does not contain a repeated edge.

$$v = v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n = w\ ,$$

where all the e_i are distinct (that is, $e_i \neq e_k$ for any $i \neq k$).

- **Simple path** a path that does not contain a repeated vertex. Thus a simple path is a walk of the form

$$v = v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n = w\ ,$$

all e_i and v_j are distinct ($v_i \neq v_j$, $e_i \neq e_j$ for any $i \neq j$).

Circuit of Graph

- A **circuit** is a closed walk that does not contain a repeated edge. Thus a circuit is a walk of the form

$$v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n$$

where $v_0 = v_n$ and all the e_i are distinct.

- A **simple circuit** is a circuit that does not have any other repeated vertex except the first and last. Thus a simple circuit is walk of the form

$$v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n$$

where all the e_i are distinct and all the v_j are distinct except that $v_0 = v_n$

Euler Circuit

- An **Euler circuit** for G is a circuit that contains every vertex and every edge of G. That is, an Euler circuit is a sequence of adjacent vertices and edges in G that starts and ends at the same vertex, uses every vertex of G at least once, and every edge exactly once.

Theorem

If a graph has an Euler circuit, then every vertex of the graph has even degree.

Theorem

- If every vertex of a nonempty graph has even degree and if the graph is connected, then the graph has an Euler circuit.

Theorem

- A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has even degree.

Euler Path

- Let G be a graph and let v and w be two vertices of G. An **Euler path** from v to w is a sequence of adjacent edges and vertices that starts at v, ends at w, passes through every vertex of G at least once, and traverses every edge of G exactly once.

Corollary

- Let G be a graph and let v and w be two vertices of G. There is a an **Euler path** from v to w if, and only if, G is connected, v and w have odd degree, and all other vertices of G have even degree.

Hamiltonian Circuit

- Given a graph G, a **Hamiltonian circuit** for G is a simple circuit that includes every vertex of G. That is, a Hamiltonian circuit of G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once.

- If a graph G has a Hamiltonian circuit then G has a sub-graph H with the following properties:
  1. H contains every vertex of G;
  2. H is connected;
  3. H has the same number of edges as vertices;
  4. every vertex of H has degree 2.

Connected Graph

- Two vertices v and w of G are **connected** if, and only if, there is a walk from *v* to *w*.
- ***G* is connected** if, and only if, for *any* two vertices *v* and *w* in *G*, there is a walk from *v* to *w*. symbolically:
  G is connected $\Leftrightarrow \forall\ v, w \in V(G), \exists$ a walk from *v* to *w*

Lemma

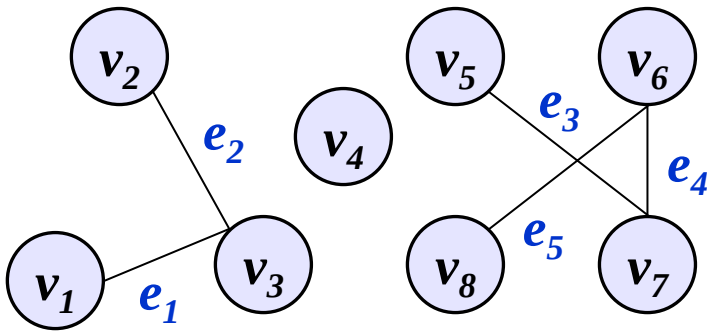
a. If *G* is connected, then any two distinct vertices of *G* can be connected by a simple path.

b. If *v, w* are in circuit and one edge is removed from the circuit, then there still exists a path from *v* to *w*

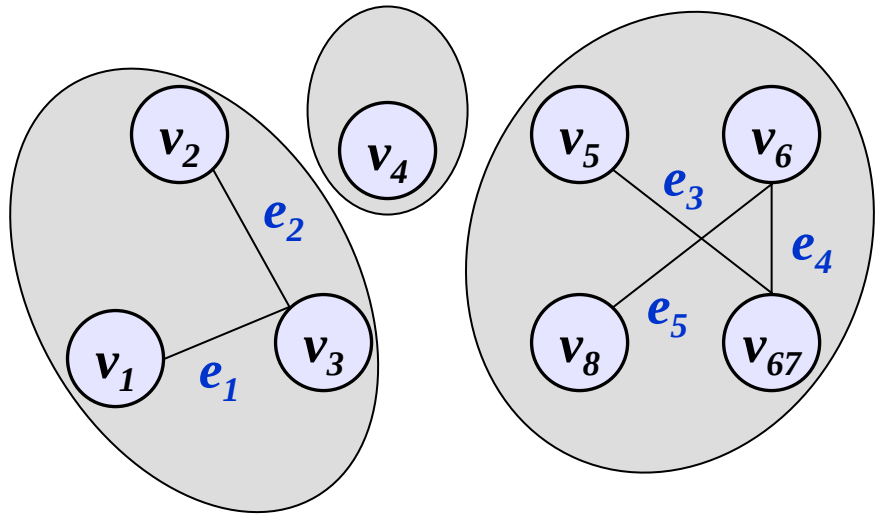
c. If *G* is connected and contains a circuit, then an edge of circuit can be removed without disconnecting *G*.

Connected Component

- A graph *H* is a **connected component** of a graph *G* if, and only if,
  1. *H* is a subgraph of *G;*
  2. *H* is connected;
  3. No connected subgraphs of *G* has *H* as a subgraph and contains vertices or edges that are not in *H*.



The Graph, *G*



Three Connected components of *G*

Isomorphism

- Let G and G' be graphs with vertex sets $V(G)$ and $V(G')$ and edge sets $E(G)$ and $E(G')$, respectively. **G is isomorphic to G'** if, and only if, there exist one-to-one correspondence $g : V(G) \rightarrow V(G')$ and $h : E(G) \rightarrow E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that

  for all $v \in V(G)$ and $e \in E(G)$,

  v is an endpoint of $e \Leftrightarrow g(v)$ is an endpoint of $h(e)$

Isomorphism Invariants

- A property P is called an **isomorphic invariant** if, and only if, given any graphs G and G', if G has property P and G' is isomorphic to G, then G' has property P.
- If G and G' are simple graphs then **G is isomorphic to G'** if, and only if, there exists a one-to-one correspondence g from the vertex set $V(G)$ of G to the vertex set $V(G')$ of G' that preserves the edge-endpoint functions of G and G' in the sense that

  for all vertices u and v of G,

  $\{u, v\}$ is an edge in $G \Leftrightarrow \{g(u), g(v)\}$ is an edge in G'

Isomorphism Invariants

Theorem

- Each of following properties is an invariant for graph isomorphism, *n, m,* and *k* are all nonnegative integers:
  1. has *n* vertices;
  2. has *m* edges;
  3. has a vertex of degree *k;*
  4. has *m* vertices of degree *k;*
  5. has a circuit of length *k;*
  6. has a simple circuit of length *k;*
  7. has *m* simple circuits of length *k*;
  8. is connected;
  9. has an Euler circuit;
  10. has a Hamiltonian circuit.

Trees

- Graph is **circuit-free** $\Leftrightarrow$ it has no nontrivial circuits.
- A graph is a **tree** $\Leftrightarrow$ it is circuit-free and connected.
- A **trivial tree** is a graph that consists of a single vertex
- **Empty tree** that does not have any vertices or edges.
- **Forest** a graph if circuit-free.
- **Terminal vertex (Leaf)** a vertex of degree 1in T
- **Internal vertex** a vertex of degree greater than 1 in T

Lemma

- Any tree that has more than one vertex has at least one vertex of degree 1.

Theorem

Statement

For any positive integer *n*, any tree with *n* vertices has *n* – 1 edges.

Solution

We prove this theorem by mathematical induction

Basis

n = 1, tree has no edge and hence true

Inductive hypothesis

Suppose that id n = k then tree has k – 1 edges

Claim

Now if we add one more vertex to the tree then exactly one edge will be added otherwise it will not remain tree. And hence it will become k edges in the tree. Proved.

Trees

Lemma

- If G is any connected graph, C is any nontrivial circuit in G, and one of the edges of C is removed form G', then the graph that remains is connected.

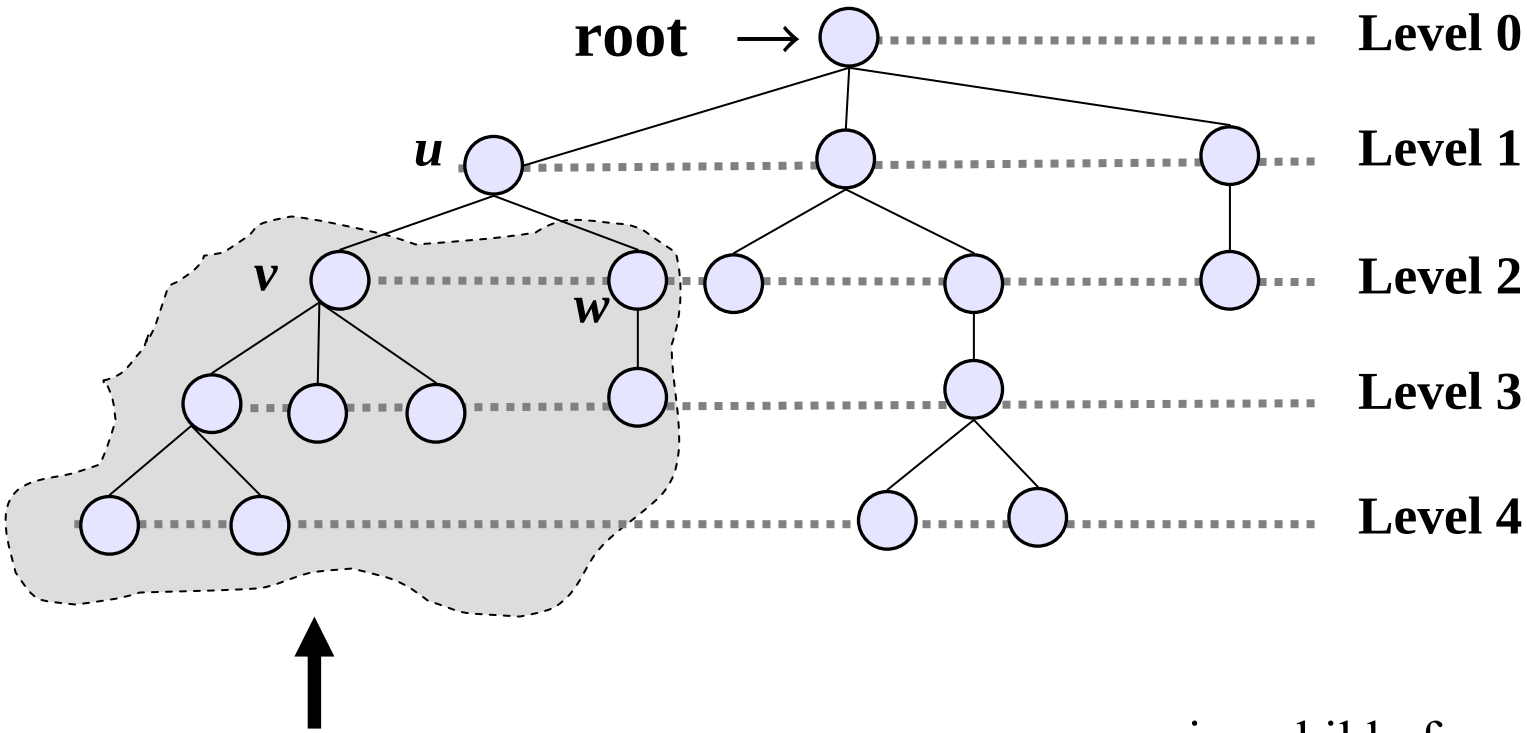
Theorem

- For any positive integer n, if G is a connected graph with n vertices and $n - 1$ edges, then G is a tree.

Rooted Trees

- **Rooted tree** a distinguished vertex
- **Level of a vertex** is the number of edges along the unique path between it and the root.
- **Height** of a rooted tree is maximum level to any vertex
- **Children** of *v* are all those vertices that are adjacent to *v* and are one level farther away from the root than *v*.
- **Parent** if *w* is child of *v*, then *v* its parent
- **Siblings** vertices that are children of same parent
- **Ancestor** and **Descendent** given vertices *v* and *w*, if *v* lies on the unique path between *w* and the root, then *v* is an **ancestor** of *w* and *w* is a **descendent** of *v*.

Rooted Trees



Vertices in enclosed region are descendants of u, which is an ancestor of each

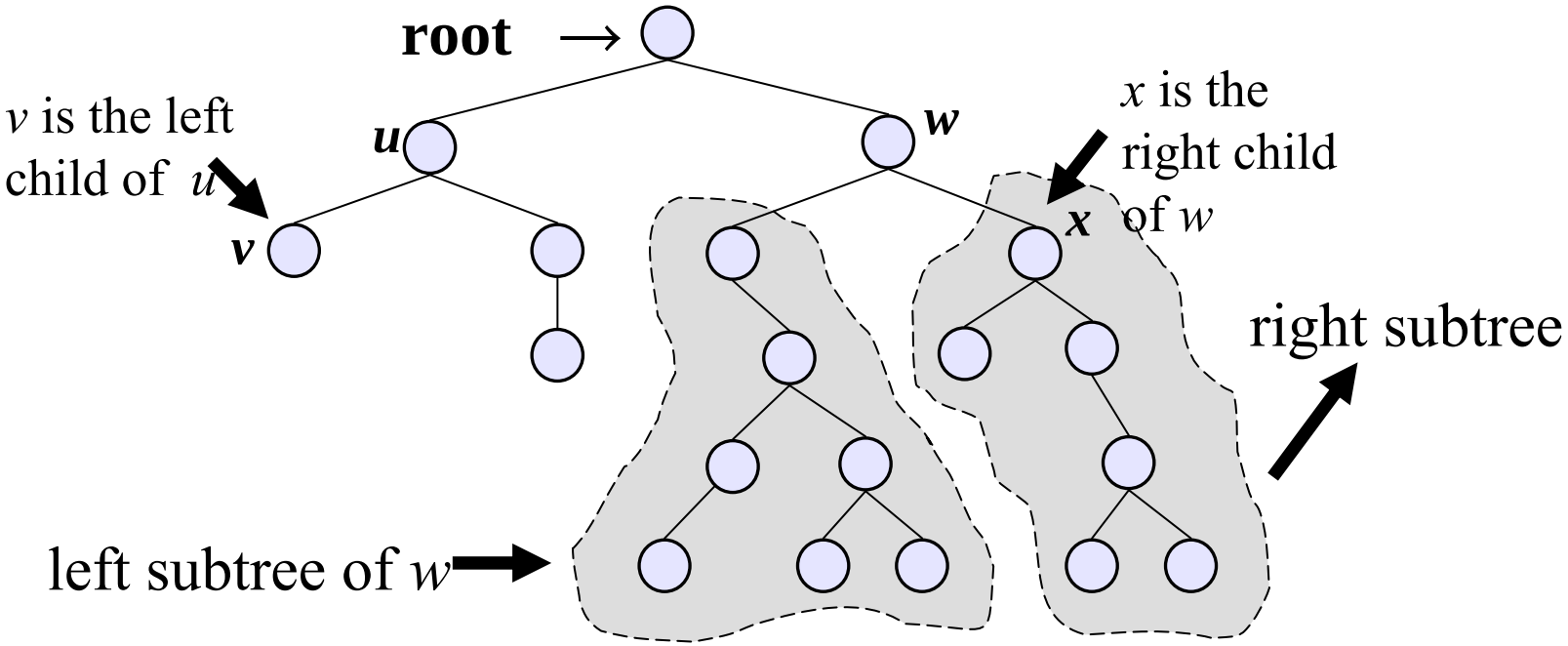
v is a child of *u*

u is the parent of *v*

v and *w* are siblings

Binary Trees

- A **binary tree** is a rooted tree in which every internal vertex has at most two children. Each child in a binary tree is either **left child** or a **right child** (but not both), an internal vertex has at most one left and one right child.
- **Full binary tree** is a binary tree in which each internal vertex has exactly two children.



Binary Trees

Theorem

- If k is a positive integer and T is a full binary tree with k internal vertices, the T has a total of $2k + 1$ vertices and has $k + 1$ terminal vertices.

Theorem

If T is a binary tree that has t number of terminal vertices and height is h, then $t \leq 2^k$ OR $\log_2 t \leq h$

Spanning Trees

- A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

Proposition

1. Every connected graph has a spanning tree.
2. Any two spanning trees for a graph have the same number of edges.

Minimal Spanning Trees

- A **weighted graph** is a graph for which each edge has an associated real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph.

- A **minimal spanning tree** for a weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graphs.

- If G is a weighted graph and e is an edge of G then $w(e)$ denotes the weight of e and $w(G)$ denotes the total weight of G.