

Graph Theory

1 Graphs and Subgraphs

Definition 1.1. A **multigraph** or just **graph** is an ordered pair $G = (V, E)$, consisting of a nonempty **vertex set** V of **vertices** and an **edge set** E of **edges** such that each edge $e \in E$ is assigned to an unordered pair $\{u, v\}$ with $u, v \in V$, possibly $u = v$, called **endpoints** of e .

For each edge e with endpoints u and v , abbreviated $e = uv$ usually, we say that e is **incident** with u and v , or that u, v are **adjacent** by e . The edge e is called a **link** if $u \neq v$ and a **loop** if $u = v$.

Let $G = (V, E)$ be a multigraph, we usually write $V = V(G)$ and $E = E(G)$. Sometimes vertices are called **nodes/sites**, edges are called **connections/bonds**. Each graph can be represented by a picture with dots and lines: Each vertex is represented by a solid dot (or by a small circle sometimes), and each edge e with endpoints u, v is represented by a line connecting u and v . Two or more edges between two vertices u, v are called **multiple edges**. The **degree** of a vertex v is

$$\deg_G(v) = (\text{number of links at } v) + 2(\text{number of loops at } v).$$

A **walk of length** ℓ is a sequence of vertices and edges (not necessarily distinct)

$$W = v_0 e_1 v_1 e_2 \dots v_{\ell-1} e_\ell v_\ell$$

such that each edge e_i is incident with vertices v_{i-1} and v_i , $1 \leq i \leq \ell$, where v_0 is called the **initial vertex** and v_ℓ the **terminal vertex**. If $v_0 = v_\ell$, W is called a **closed walk**. A **trail** is a walk whose edges are distinct. A **path** is a walk whose vertices and edges are distinct, except the initial and terminal vertices.

For instance, in Figure 1, the vertex-edge sequence $v_2 e_9 v_4 e_8 v_3 e_6 v_5$ is a path, but not closed. The sequence $v_4 e_8 v_3 e_6 v_5 e_7 v_3 e_2 v_2$ is a trail, but not closed. The

sequence $v_4e_8v_3e_6v_5e_7v_3e_3v_6e_5v_5e_7v_3e_6v_5e_{12}v_4$ is a closed walk. The closed paths $v_1e_1v_2e_2v_3e_3v_4e_4v_1$, $v_3e_6v_5e_7v_3$, and the loop $v_6e_4v_6$ are all cycles.

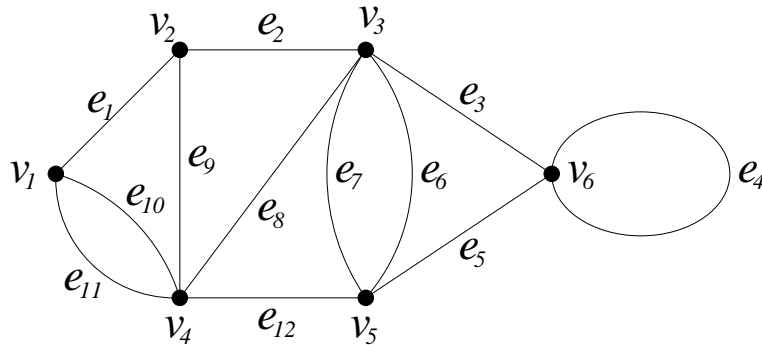


Figure 1: A graph with multiple edges and loops

A **subgraph** of a graph G is graph H such that

$$V(H) \subseteq V(G), \quad E(H) \subseteq E(G).$$

A graph is called **connected** if there exists a path connecting any two vertices, otherwise, it is called **disconnected**. A maximal connected subgraph of G is called a **component**. We denote

$$c(G) = \text{number of connected components of } G.$$

A **cycle** is a connected graph having degree 2 everywhere. The **length** of a cycle is its number of edges. A loop is viewed as a cycle of length 1. Two edges between two distinct vertices are viewed as a cycle of length 2.

Definition 1.2. A graph is said to be **simple** if there are no loops and no multiple edges between two distinct vertices. A simple graph can be viewed as an irreflexive and symmetric binary relation on its vertex set.

Let $G = (V, E)$ be a graph with the vertex set $V = \{v_1, v_2, \dots, v_n\}$ and the edge set $E = \{e_1, e_2, \dots, e_m\}$. The **adjacency matrix** of G is an $n \times n$ matrix $A(G) = [a_{ij}]$, whose (i, j) -entry is

$$a_{ij} = \text{number of edges between } v_i \text{ and } v_j.$$

The **incidence matrix** of G is an $n \times m$ matrix $M(G) = [b_{ij}]$, whose (i, j) -entry is

$$b_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is incident with } e_j, \\ 0 & \text{if } v_i \text{ is not incident with } e_j. \end{cases}$$

Proposition 1.3. *Every graph G has the relation*

$$\sum_{v \in V(G)} \deg_G(v) = 2|E(G)|.$$

Proof. Each edge contributes 2 in the sum of degrees over vertices. It follows that the sum is twice of the number of edges. \square

An **isomorphism** from a graph G to a graph H is a pair of bijections

$$\phi : V(G) \rightarrow V(H), \quad \psi : E(G) \rightarrow E(H)$$

such that for each edge $e \in E(G)$ and two vertices $u, v \in V(G)$, e is incident with u, v if and only if $\phi(e)$ is incident with vertices $\phi(u), \phi(v)$, i.e.,

$$e = uv \iff \psi(e) = \phi(u)\phi(v).$$

Whenever such a pair of bijections exists, we say that G is **isomorphic** to H . Isomorphic graphs have the same number of vertices, the same number of edges, and the same degree sequences. For instance, the graphs G and H in Figure 2 are isomorphic; an isomorphism ϕ is given by

$$\begin{aligned} \phi(u_1) &= u'_1, & \phi(u_2) &= u'_2, & \phi(u_3) &= u'_3, \\ \phi(v_1) &= v'_1, & \phi(v_2) &= v'_2, & \phi(v_3) &= v'_3. \end{aligned}$$

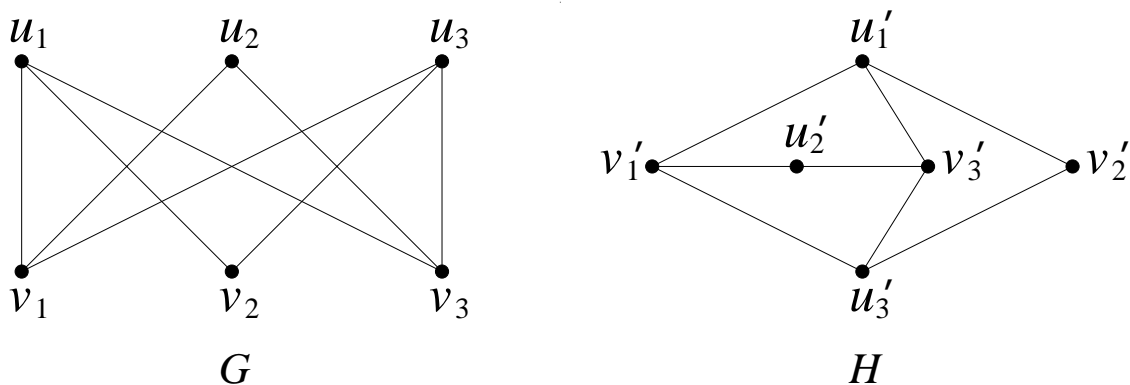


Figure 2: Isomorphic graphs G and H

A **complete graph** is a graph such that every pair of two distinct vertices are adjacent. We denote by K_n the complete graph with n vertices.

A graph $G = (V, E)$ is called **bipartite** if V can be partitioned into two disjoint nonempty subsets V_1, V_2 such that each edge of G is between a vertex

in V_1 and a vertex in V_2 . If every pair of vertices between V_1 and V_2 is an edge of G , then G is called a **complete bipartite graph**. The complete bipartite graph with m vertices in V_1 and n vertices in V_2 is denoted by $K_{m,n}$.

Proposition 1.4. *A graph G is bipartite if and only if every cycle of G has even length.*

Proof. We may assume that G is connected. If G is bipartite, then every edge of G is between two disjoint nonempty sets V_1 and V_2 . It follows that the sequence of any cycle is alternating between V_1 and V_2 . So the length of any cycle must be even.

Conversely, if every cycle of G has even length, we show that G is bipartite. Without loss of generality, we may assume that G is connected. Let $d(u, v)$ denote the **distance** (length of a shortest path) between two vertices u and v . Fix a vertex v_0 . Let

$$\begin{aligned} V_1 &= \{w \in V(G) : d(v_0, w) \text{ is odd}\}, \\ V_2 &= \{w \in V(G) : d(v_0, w) \text{ is even}\}. \end{aligned}$$

We claim that there is no edge whose endpoints belong to V_1 .

Given two vertices $u, v \in V_1$. Let $P(v_0, u)$ be a shortest path from v_0 to u , and $Q(v_0, v)$ a shortest path from v_0 to v . Let v_0, v_1, \dots, v_k be the common vertices on $P(v_0, u)$ and $Q(v_0, v)$, starting from v_0 .

Since $P(v_0, u)$ and $Q(v_0, v)$ are shortest paths, the subpath $P(v_0, v_k)$ of $P(v_0, u)$ from v_0 to v_k must be shortest; so is the subpath $Q(v_0, v_k)$ of $Q(v_0, v)$ from v_0 to v_k . Then $P(v_0, v_k)$ and $Q(v_0, v_k)$ have the same length. Let $P(v_k, u)$ denote the subpath of $P(v_0, u)$ from v_k to u , and $Q(v_k, v)$ the subpath of $Q(v_0, v)$ from v_k to v . It follows that the length of $P(v_k, u)$ and $Q(v_k, v)$ have the same parity; that is, both are even or both are odd.

Now suppose there is an edge e adjacent with vertices u, v . Let $P^{-1}(v_k, v)$ denote the reversed path of $P(v_k, v)$, from v to v_k . Then the closed path $P(v_k, u)eP^{-1}(v_k, v)$ is a cycle of odd length, a contradiction. Hence there is no edge inside V_1 . Likewise, there is no edge inside V_2 . We see that G is bipartite. \square

2 Euler Trail Problem

A trail in a graph G is called an **Euler trail** if it uses every edge exactly once. An **Euler tour** is a closed Euler trail, and an **Euler path** is a non-closed Euler trail.

Theorem 2.1. *A connected graph has an Euler tour if and only if it has even degree everywhere.*

Proof. Let G be connected.

“ \Rightarrow ” Let $W = v_0e_1v_1e_2v_2 \cdots e_nv_nv_n$ be an Euler tour with $v_0 = v_n$. Fix a vertex $v \in V(G)$ and travel along W . The number of times of moving toward v equals the number of times of moving away from v .

Let v appear in W as v_{i_1}, \dots, v_{i_k} . If $v \neq v_0 = v_n$, then $\deg(v) = 2k$. If $v = v_0 = v_{i_1} = v_{i_k} = v_n$ with $k \geq 2$, then $\deg(v) = 2(k - 1)$. This can also be argued as follows: When one travels along the Euler tour and passes by a vertex v , the person must come towards v through one edge and depart from v through another edge. So the number of times coming towards v is the same number of times departing from v . So the degree of v in G must be even.

“ \Leftarrow ” Let G have even degree at all vertices. Consider a longest trail P , whose vertex-edge sequence is $P = v_0e_1v_1e_2v_2 \cdots v_{n-1}e_nv_n$.

(a) We claim $v_0 = v_n$. Suppose $v_0 \neq v_n$. Let G' be the graph obtained from G by removing all edges of P . If v_n appears k times in the subtrail $P_1 = v_0e_1v_1 \cdots e_{n-1}v_{n-1}e_n$ without terminal vertex, then the degree of v_n will be reduced by $2k$ when the edges of P_1 are removed (including one side of the possible loop e_n if $v_{n-1} = v_n$), and the degree of v_n will be further reduced by 1 when the edge e_n is removed at v_n . So the degree of v_n in P is $2k + 1$. Thus the degree of v_n in G' is an odd number. This means that there is at least one edge e in G' adjacent to v_n , say, v_nev . Hence we can construct a longer trail $v_0e_1v_1 \cdots e_nvnev$, which is a contradiction.

(b) We claim that the longest trail uses every edge of G . Suppose there is an edge e which is not used in a longest path P . If u or v is some v_i , say, $v = v_i$, then

$$uev(v_i)e_{i+1}v_{i+1} \cdots e_nv_n(v_0)e_1v_1 \cdots e_iv_i$$

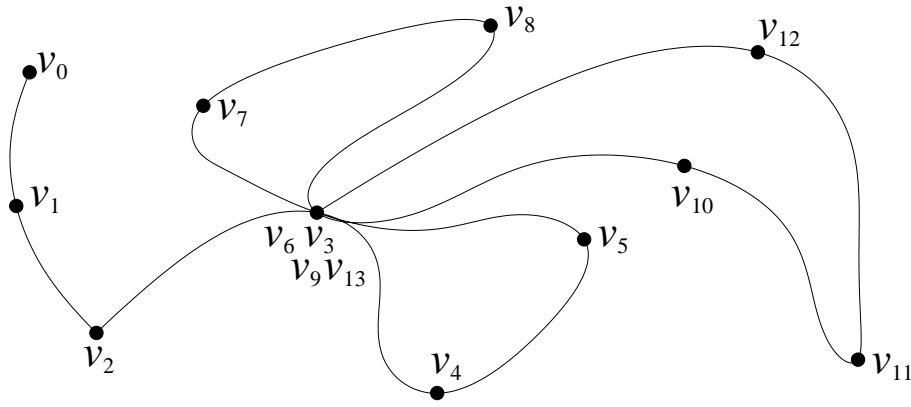


Figure 3: A longest path in G that is not closed.

is a longer trail, which is a contradiction. If neither u nor v is in the trail P , there is a shortest path $ue'_1u_1e'_2\cdots e'_mu_m$ from u to P , where $u_m = v_j$ for some j . It is clear that $ue'_1u_1e'_2\cdots e'_mu_m$ does not contain any edge of $v_0e_1v_1e_2v_2\cdots e_nv_n$. Thus

$$ue'_1u_1e'_2\cdots e'_mu_me_{j+1}v_{j+1}e_{j+2}\cdots e_nv_n(v_0)e_1v_1\cdots e_jv_j$$

is a longer trail, which is again a contradiction. \square

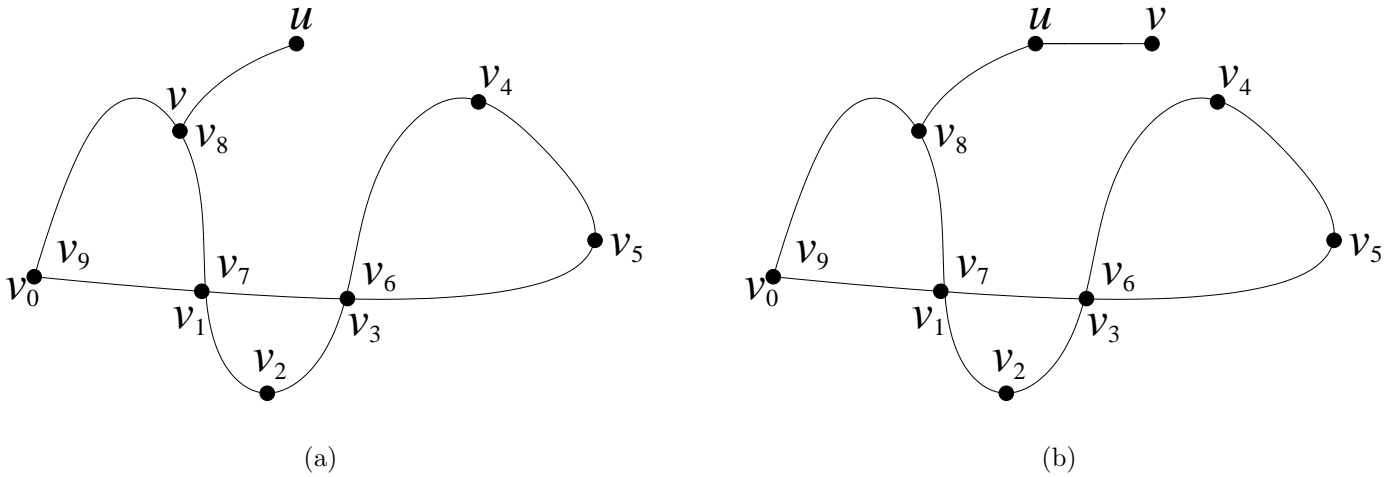


Figure 4: A longest trail in G that uv is not used.

Corollary 2.2. *A connected graph G has an Euler path from one vertex u to another vertex v if and only if u, v have odd degree and other vertices have even degree.*

Proof. Let G' be a graph obtained from G by adding a new edge e between u and v . Then G' has an Euler tour W' by Theorem 2.1. Thus $W = W' \setminus e$ for G is an Euler path from u, v .

Conversely, if G has even degree for every vertex, then it has an Euler tour by Theorem 2.1. Let G be a graph having exactly two vertices u, v of odd degree. We add a new edge e between u and v to G to have graph G' . Then G' has even degree everywhere. Thus G' has an Euler tour by Theorem 2.1. Remove the edge e from the Euler tour for G' . We obtain an Euler path for G . \square

There is an algorithms to find an Euler tour or Euler trail in a graph.

Theorem 2.3. (Fleury's Algorithm) Input: Graph $G = (V, E)$.

Output: Euler tour, or non-closed Euler trail, or none of previous two.

STEP 1: If there are vertices of odd degree, choose such a vertex v . Otherwise, choose any vertex v . Set **SEQ** = v .

STEP 2: If there is no edge remaining at the terminal vertex v of **SEQ**, **STOP**. (**SEQ** forms an Euler trail. If v is the same as the initial vertex of **SEQ**, it is an Euler tour.)

STEP 3: If there is exactly one edge e from v to another vertex w , then remove ve from the graph, and go to **STEP 5**.

STEP 4: If there are two or more edges remaining at v , choose one of these edges, say an edge $e = vw$, in such a way that the removal of e will not disconnect the remaining graph, then remove e from the graph, and go to **STEP 5**. If such an edge can not be selected, **STOP**. (There is neither Euler tour nor Euler trail.)

STEP 5: Add ew to the end of **SEQ**, replace v by w , and return to **STEP 2**.

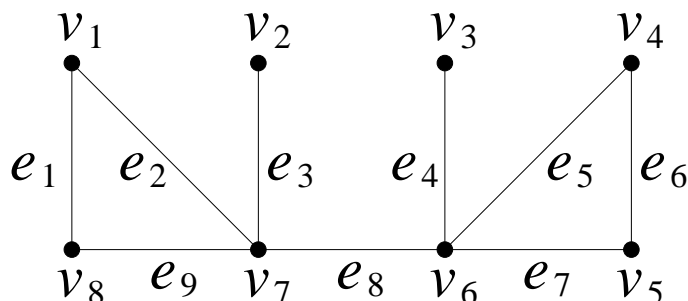


Figure 5: A graph with an Euler path.

Example 2.1. The vertex-edge sequence of an Euler trail for the graph in Figure 5 is given as

$$\text{SEQ} = v_2e_3v_7e_2v_1e_1v_8e_9v_7e_8v_6e_5v_4e_6v_5e_7v_6e_4v_3.$$

3 Hamilton Cycle Problem

A cycle in a graph is called a **Hamilton cycle** if it contains all vertices of the graph. A **Hamilton graph** is graph containing a Hamilton cycle. A **Hamilton path** is a non-closed path that visits every vertex exactly once. Unfortunately, there is no simple necessary and sufficient condition to check whether a graph contains a Hamilton cycle.

Theorem 3.1. *Let G be a simple graph with $|V(G)| = n \geq 3$. If for any pair of non-adjacent vertices $u, v \in V$,*

$$\deg_G(u) + \deg_G(v) \geq n, \tag{1}$$

then G contains a Hamilton cycle.

Proof. Suppose the theorem is not true for some $n \geq 3$. Fix such an n and let G be a counterexample with $|E(G)|$ as large as possible. Note that G is a subgraph of K_n which contains obviously Hamilton cycles, and $G \neq K_n$. Adding one edge of K_n that is not contained in G to the graph G , we obtain a graph G' which still satisfies obviously the degree condition (1). By the choice of G , the graph G' would contain a Hamilton cycle. This means that G must already contain a Hamilton path with vertex sequence, say $v_1v_2 \dots v_n$. Since G contains no Hamilton cycle, the vertices v_1, v_n are not adjacent in G . We then have

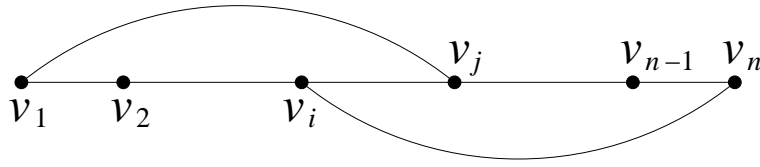
$$\deg_G(v_1) + \deg_G(v_n) \geq n.$$

Let $V_1, V_2 \subseteq \{v_2, \dots, v_{n-1}\}$ be defined by

$$\begin{aligned} V_1 &= \{v_i : (v_1, v_i) \in E(G), i = 2, \dots, n-1\}, \\ V_2 &= \{v_i : (v_i, v_n) \in E(G), i = 2, \dots, n-1\}. \end{aligned}$$

Then $|V_1| = \deg_G(v_1)$, $|V_2| = \deg_G(v_n)$, and

$$|V_1| + |V_2| \geq n, \quad |V_1 \cup V_2| \leq n - 2.$$



It follows that

$$|V_1 \cap V_n| = |V_1| + |V_2| - |V_1 \cup V_2| \geq 2.$$

Let $v_i, v_j \in V_1 \cap V_2$, $i < j$. Then $v_1v_j, v_iv_n \in E(G)$. Thus

$$v_1v_2 \dots v_{i-1}v_iv_nv_{n-1}v_{n-2} \dots v_{j+1}v_jv_1$$

is a Hamilton cycle of G . □

4 Quotient Graphs

Let $G = (V, E)$ be a graph. Let \sim be an equivalence relation on V . The **quotient graph** of G with respect to \sim is a graph whose vertex set is the quotient set V/\sim and two equivalence classes $[u], [v]$ form an edge if and only if uv forms an edge in G .

Example 4.1. Let $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_8\}$, E is defined by

$$v_iv_j \in E \Leftrightarrow |i - j| \geq 3.$$

See Figure 6 (a). Let \sim be a relation on V defined by

$$v_i \sim v_j \Leftrightarrow 5 \mid (i - j).$$

Then \sim is an equivalence relation. The quotient graph G/\sim is demonstrated in Figure 6 (b).

Example 4.2. Let $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_8\}$, E is defined by

$$v_iv_j \in E \Leftrightarrow 3 \leq |i - j| \leq 5.$$

See Figure 7 (a). Let \sim be a relation on V defined by

$$v_i R_2 v_j \Leftrightarrow 6 \mid (i - j).$$

Then \sim is an equivalence relation, and the quotient graph G/\sim is given in Figure 7 (b).

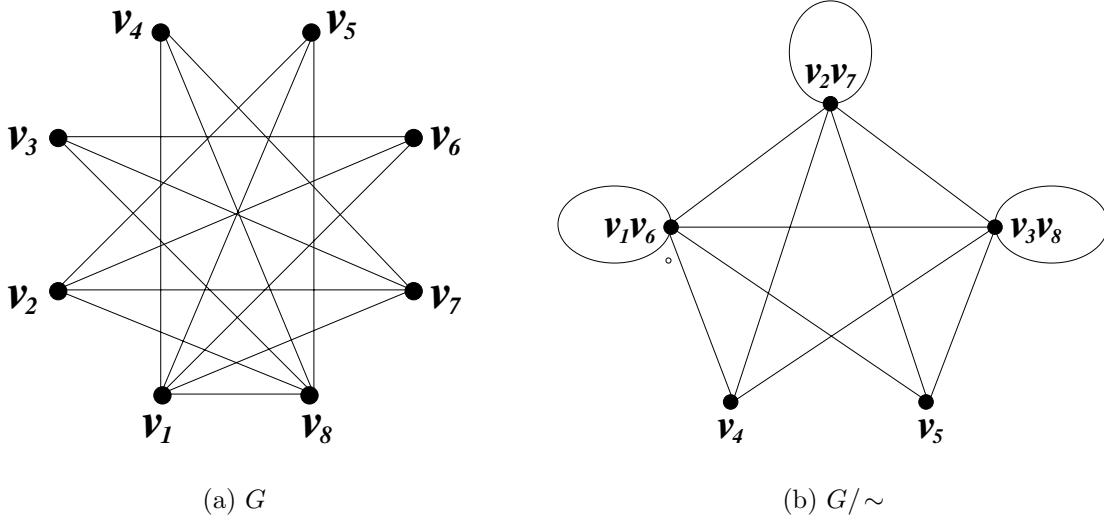


Figure 6: Construction of a quotient graph

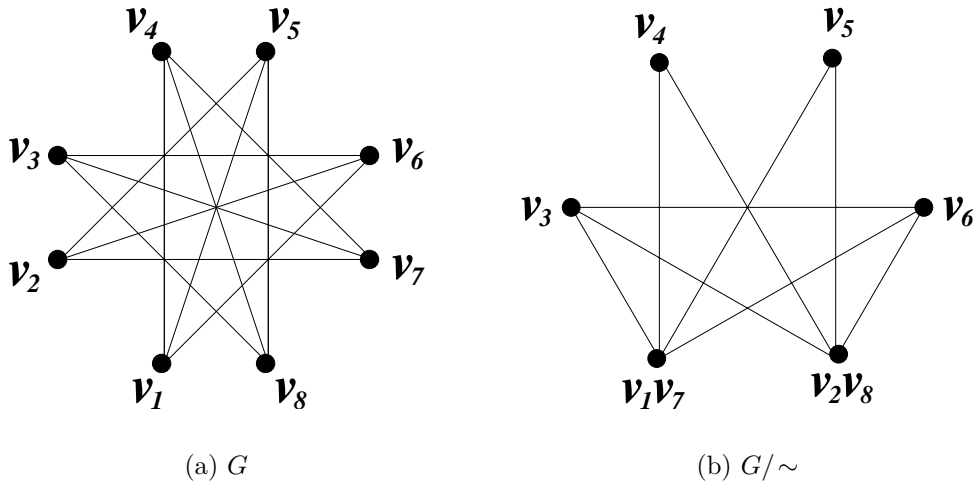


Figure 7: Construction of a quotient graph

5 Trees

Definition 5.1. An **acyclic graph** (or **forest**) is a graph containing no cycles. A **tree** is a connected acyclic graph.

Non-isomorphic trees up to six vertices are listed in Figure 8. It is a difficult problem to construct all non-isomorphic trees. We don't even know how to count the number of non-isomorphic trees with n vertices. However, the number of labeled trees with n vertices is n^{n-2} .

A **spanning subgraph** of a graph G is a subgraph H such that

$$V(H) = V(G).$$

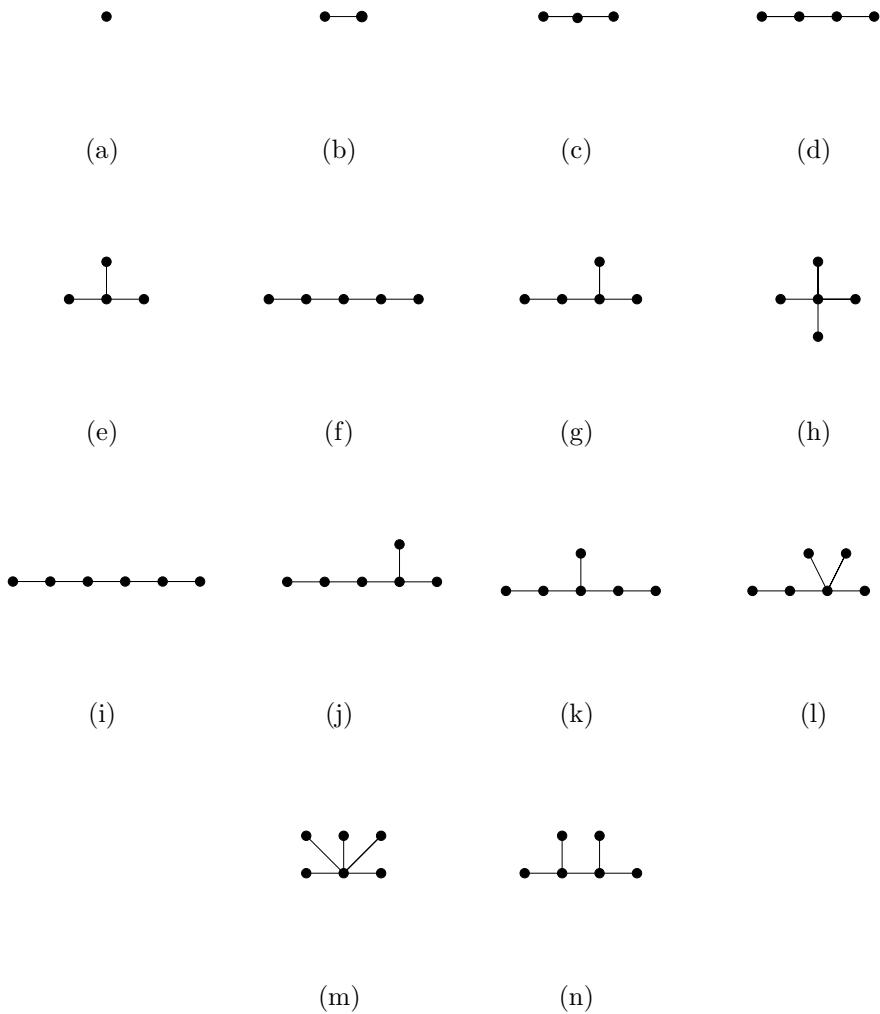


Figure 8: Non-isomorphic trees with up to six vertices

A maximal spanning subgraph without cycles is called a **spanning forest**. A **spanning tree** is a connected spanning subgraph without cycles.

Theorem 5.2. *Every connected graph contains a spanning tree.*

Proof. Given a connected graph G with or without cycles. If G contains no cycles, then G itself is a spanning tree. If G contains cycles, choose a cycle C of G and remove one edge from C . We obtain a connected spanning subgraph G_1 . If G_1 still contains cycles, choose a cycle C_1 of G_1 and remove one edge from C_1 to obtain a connected spanning subgraph G_2 . Continue this procedure when there are still cycles in the newly obtained spanning subgraph of G . Since the number of edges is finite, we eventually obtain a connected spanning subgraph containing no cycles. This final spanning subgraph is a spanning tree of G . \square

Theorem 5.3 (Edge-Vertex Tree Formula). *For any tree $T = (V, E)$,*

$$|E| = |V| - 1. \quad (2)$$

Proof. We apply induction on $|E|$. When $|E| = 0$, the tree T is a single-vertex graph without edges; clearly, $|E| = |V| - 1$. Consider the case $|E(T)| \geq 1$. Choose an edge $e = uv \in E(T)$ and remove it from T . We obtain two subtrees T_1, T_2 of T ; see Figure 9. By induction, we have

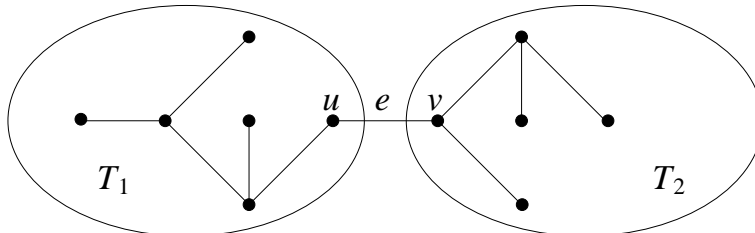


Figure 9: Subtrees T_1, T_2 of T by removing an edge e .

$$|E(T_1)| = |V(T_1)| - 1, \quad |E(T_2)| = |V(T_2)| - 1.$$

It follows that

$$\begin{aligned} |E(T)| &= |E(T_1)| + |E(T_2)| + 1 \\ &= |V(T_1)| + |V(T_2)| - 1 = |V(T)| - 1. \end{aligned}$$

□

Corollary 5.4. *Let G be a forest with $c(G)$ connected components. Then*

$$|E(G)| = |V(G)| - c(G). \quad (3)$$

Proof. Note that each component of G is a tree. Applying the Tree Formula (2) to each component of G , we obtain $|E(G)| = |V(G)| - c(G)$. □

Theorem 5.5 (Tree Characterizations). *The following statements for a graph $G = (V, E)$ are equivalent.*

- (1) G is a tree.
- (2) There exists a unique path from any vertex to any other vertex.
- (3) G is connected, and the removal of any edge disconnects G .

(4) G contains no cycle and $|E| = |V| - 1$.

(5) G is connected and $|E| = |V| - 1$.

(6) For each edge $e \notin E$, $G \cup e$ contains exactly one cycle.

Proof. (1) \Rightarrow (2): Suppose that there are two paths from a vertex u to a vertex v , say,

$$P = u_0x_1u_1x_2 \dots u_{m-1}x_mu_m, \quad Q = v_0y_1v_1y_2 \dots v_{n-1}y_nv_n,$$

where $u_0 = v_0 = u$ and $u_m = v_n = v$. Then PQ^{-1} is a closed walk, and subsequently contains a cycle. This is contradict to that G contains no cycles.

(2) \Rightarrow (3): The graph G is connected by definition. Choose an edge $e = uv \in E$ and remove it from G to have $G' := G - e$. Suppose G' is still connected. Then there is a path P from u to v . Thus G has two paths uv and P , a contradiction.

(3) \Rightarrow (4): Suppose G contains some cycles. Then the removal of any edge from a cycle will not disconnect G ; this is a contradiction. So G contains no cycles. Since G is connected and contains no cycles, then by definition G is a tree. Thus $|E| = |V| - 1$ by the tree formula (2).

(4) \Rightarrow (5): Since G contains no cycles, it follows from (3) that G has only one component. That is G is connected and $|E| = |V| - 1$.

(5) \Rightarrow (6): Suppose $G' = G \cup e$ is not a graph having exactly one cycle. Then G' may have no cycles or may have more than one cycle.

Case 1: If G' has no cycles, then G' is a tree because G' is connected. Thus $1 \leq c(G') \leq c(G) = 1$. So $c(G) = c(G') = 1$. Since G has no cycles, we have

$$|V| - |E| = c(G) = c(G') = |V| - |E| - 1.$$

This is a contradiction.

Case 2: If G' has more than one cycle, then G has some cycles. Removing some edges from G to obtain a spanning tree T of G . Then $|E(T)| = |V| - 1$. Thus $|E| = |E(T)|$. This is impossible because some edges have been removed from G .

(6) \Rightarrow (1): It is clear that G contains no cycles; otherwise joining one edge will result more cycles. It is also clear that G is connected; otherwise, joining an edge between two components will not result any cycle. So G is a tree. \square

A **leaf** in a tree is a vertex of degree one. For a single-vertex tree, there is no leaf. For trees with more than one vertex, we shall see that there must exist at least two leaves.

Theorem 5.6. *A tree with more than one vertex has at least two leaves.*

First Proof. Suppose the theorem is not true. Let $T = (V, E)$ be a tree with $|V| \geq 2$, having only one leaf or no leaves.

Case 1: If T has only one leaf, then T has $|V| - 1$ vertices of degree at least two and one vertex of degree 1. Thus by the tree formula $|E| = |V| - 1$ and the degree-sum formula, we have the contradiction

$$2(|V| - 1) = 2|E| = \sum_{v \in V} \deg(v) \geq 2(|V| - 1) + 1.$$

Case 2: If T has no leaves, then the degree of every vertex is at least two. We thus have the contradiction

$$2(|V| - 1) = 2|E| = \sum_{v \in V} \deg(v) \geq 2|V|.$$

Second Proof. We make induction on the number of edges of trees with more than one vertex. It is obviously true for a tree with only one edge. Let $T = (V, E)$ be a tree with two or more vertices. Choose an edge $x = uv \in E$ and removed it from T to obtain two subtrees T_u and T_v , where u is a vertex of T_u and v is a vertex of T_v .

If both T_u and T_v have more than one vertex, then by induction both T_u and T_v have at least two leaves. Then T_u must have a leaf other than u and T_v must have a leaf other than v . Thus T has at least two leaves (two vertices of degree 1 other than u and v).

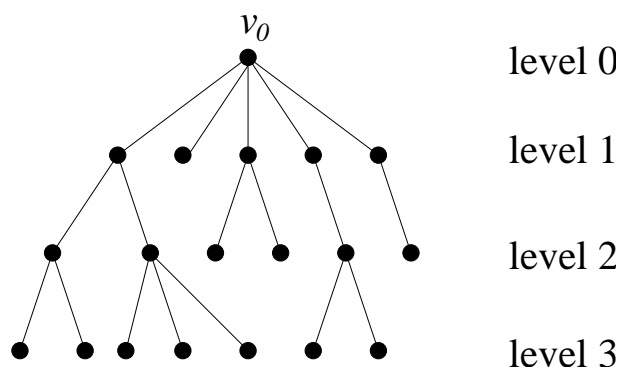
If T_u has the only vertex u and T_v has more than one vertex, then u is a leaf of T and T_v has a leaf other than v . Thus T has at least two leaves. It is similar when T_u has more than one vertex and T_v has the only vertex v .

If both T_u and T_v have only one vertex, then T is a tree with the only edge uv ; the vertices u and v are two leaves of T . \square

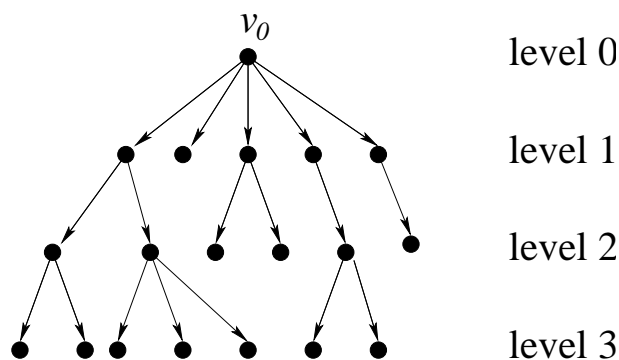
6 Rooted Trees

A **rooted tree** is a tree T with a specified vertex v_0 , called the **root** of T , denoted by (T, v_0) . A vertex v of a rooted tree (T, v_0) is called a vertex of **level** k if the length of the unique path from the root v_0 to the vertex v is k . The largest level number of a rooted tree is called the **height** of the rooted tree.

A rooted tree can be described by a graph



A rooted tree can be also described by a directed tree. This can be done by adding an arrow to each edge from a vertex of level k to a vertex of level $k + 1$. For instance, the rooted tree above can be drawn as a directed graph



In this way a rooted tree can be characterized as a directed graph with a vertex v_0 such that $\text{ideg}(v_0) = 0$ and $\text{ideg}(v) = 1$ for all other vertices v . If (u, v) is an directed edge of a rooted tree from a vertex u to a vertex v , we say that u is the **parent** of v or v a **child** of u . Every vertex except the root has exactly one parent; a parent may have several children. If there is a unique directed

path from a vertex v to a vertex w , we say that v is an **ancestor** of w or w a **descendant** of v .

A rooted tree is called a **binary tree** if each vertex has at most two children: either a left child or a right child; both a left child and a right child; or no children at all. Similarly, for $m \geq 2$, an **m -ary tree** is a rooted tree that every parent has at most m children. The children of a parent can be labelled by the numbers from $\{1, 2, \dots, m\}$. If a parent has less than m children, we say that the i th child is **absent** in case no child is labelled with i . For any vertex v of an m -ary tree, we have $\text{odeg}(v) \leq m$. An m -ary tree is called a **regular m -ary tree** (or **complete m -ary tree**) if the out-degree is either m or 0 for all vertices. A regular m -ary tree is called a **full m -ary tree** if all leaves are in the same level.

Example 6.1. Find the number of vertices for a full m -ary tree of height h .

Solution.

$$1 + m + m^2 + \dots + m^h = \frac{m^{h+1} - 1}{m - 1}.$$

7 Labeled Trees

An **ordered rooted tree** is a rooted tree of which the vertices are labeled in certain linear order.

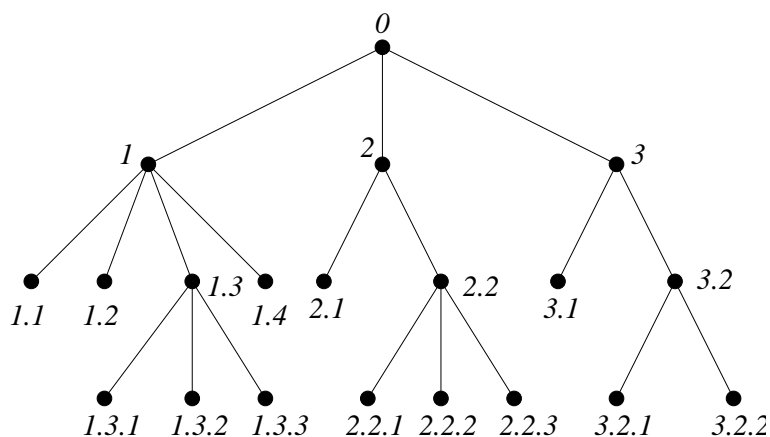
Proposition 7.1 (Universal Address System for Rooted Tree). **Input:** *A rooted tree.* **Output:** *A labeled rooted tree.*

STEP 1. Assign 0 to the root.

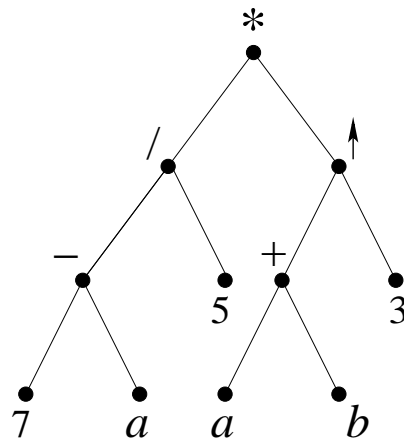
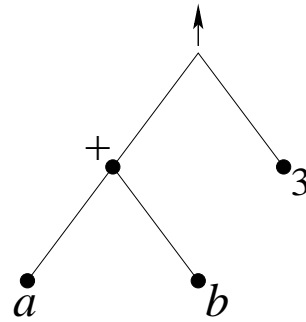
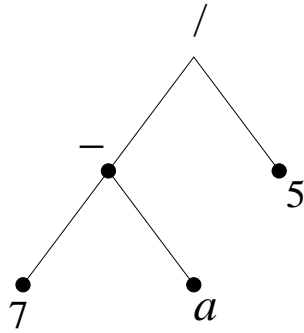
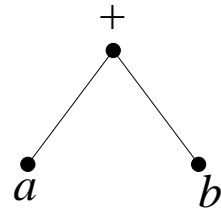
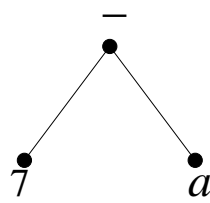
STEP 2. Assign positive integers to the vertices in level 1, going from left to right.

STEP 3. Let v be an interval vertex of level $n \geq 1$. Let v_1, v_2, \dots, v_k denote children of v , going from left to right. If v is labeled ℓ , then label v_1, v_2, \dots, v_k as $\ell.1, \ell.2, \dots, \ell.k$, respectively.

Example 7.1. Following example demonstrates this procedure.



Example 7.2. Polish Notations: $[(7 - a)/5] * [(a + b) \uparrow 3]$



Example 7.3. Consider a linearly ordered finite set Σ of alphabets. Then Σ^* can be made into a rooted tree as follows:

- (i) The empty word λ serves as the root.
- (ii) For each word $w \in \Sigma^*$, the set of its children is

$$C_w = \{wx : x \in \Sigma\}.$$

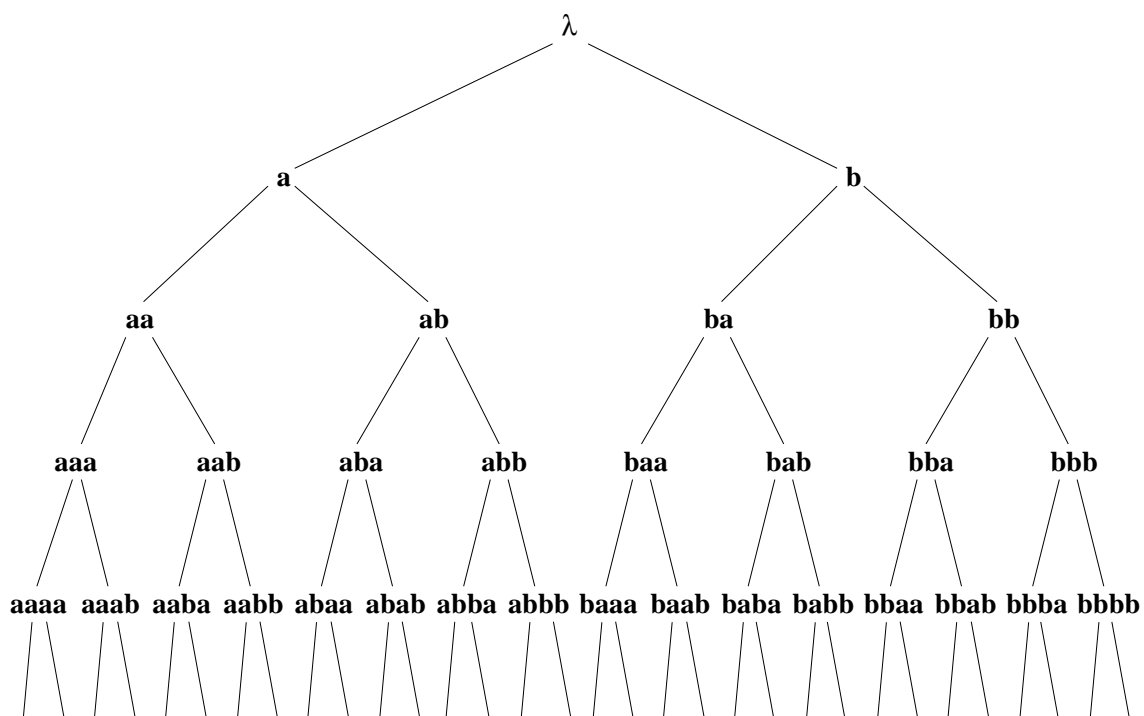
Since Σ is linearly ordered, the elements of C_w can be linearly ordered as follows:
For $x, y \in \Sigma$,

$$wx < wy \iff x < y.$$

In doing so we obtain a rooted infinite tree Σ_w^* having the root w and the vertex set

$$w\Sigma^* = \{wx : x \in \Sigma^*\}.$$

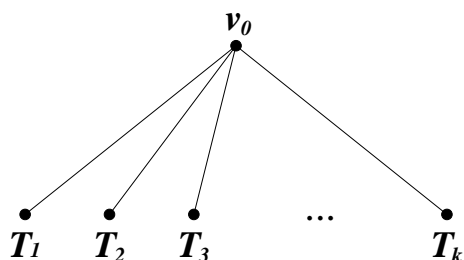
For instance, for $\Sigma = \{a, b\}$ with $a < b$, the rooted tree Σ_λ^* is



8 Tree Searching

Given a rooted tree (T, v_0) . In many occasions we need to visit every vertex exactly once in a specified order. The process to visit vertices of a tree in a specified order is called **searching** or performing a **tree search**. In some other books this process is also called **traversal**.

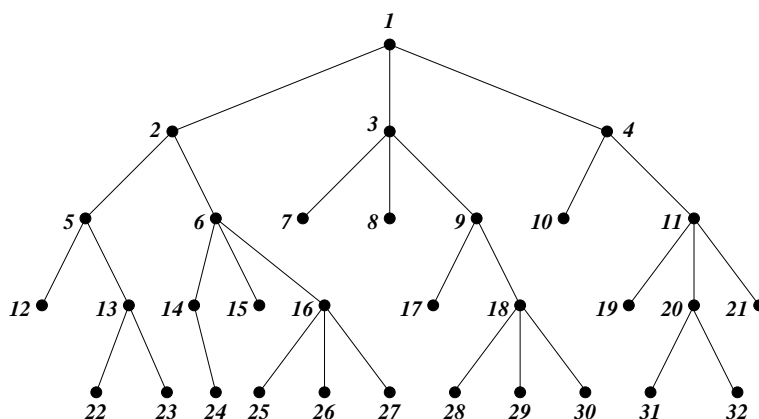
Let (T_0, v_0) be a rooted tree with root v_0 . Let T_1, T_2, \dots, T_k denote the subtrees of T_0 with their roots at the children of v_0 , ordered from left to right as follows:



Preorder Search.

Visit the root v_0 first. Then visit the vertices of T_1 in preorder, the vertices of T_2 in preorder, and so on until the vertices of T_k are visited in preorder. We summarize: **visit a parent first then visit its children next**.

Example 8.1. Find the linear order for the vertices of the tree below in preorder.



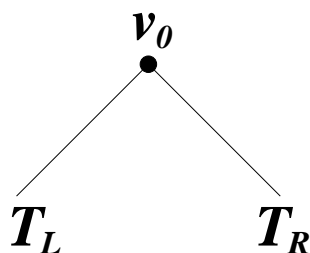
Solution. 1, 2, 5, 12, 13, 22, 23, 6, 14, 24, 15, 16, 25, 26, 27, 3, 7, 8, 9, 17, 18, 28, 29, 30, 4, 10, 11, 19, 20, 31, 32, 21.

Postorder Searching. Let T_1, T_2, \dots, T_k be subtrees of the root v_0 . Visit the vertices of T_1, T_2, \dots, T_k respectively in postorder first. Then visit the root v_0

next. We summarize: **visit children first then visit their parent next.**

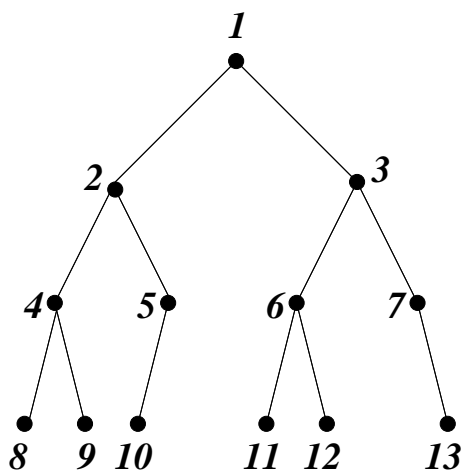
Example 8.2. The linear order for the vertices of the tree above in postorder: 12, 22, 23, 13, 5, 24, 14, 15, 25, 26, 27, 16, 6, 2, 7, 8, 17, 28, 29, 30, 18, 9, 3, 10, 19, 31, 32, 20, 21, 11, 4, 1.

Inorder Search. For a binary rooted tree (T, v_0) , the descendants of the root v_0 can be divided into a left side rooted subtree T_L and a right side rooted subtree T_R as follows:



One can visit the vertices of the subtree T_L in inorder first, then visit the root v_0 next, and then visit the vertices of the subtree T_R in inorder last. This can be summarized as **visit a left child first, then visit the parent next and the right child last.**

Example 8.3. Find the linear order for the vertices of the binary rooted tree below in preorder, postorder, and in inorder respectively.



Solution.

Preorder: 1, 2, 4, 8, 9, 5, 10, 3, 6, 11, 12, 7, 13.

Postorder: 8, 9, 4, 10, 5, 2, 11, 12, 6, 13, 7, 3, 1.

Inorder: 8, 4, 9, 2, 10, 5, 1, 11, 6, 12, 3, 7, 13.

9 Depth-First Search

The three methods of visiting vertices of a rooted tree above have a common feature: Each of them goes as far as it can go by following the edges of the tree away from the root, then it backs up a bit and again goes as far as it can, and so on. The strategy is so-called **depth-first search** (DFS). Let us give another example of such strategy by finding a spanning tree of a connected graph.

Depth-First Search Algorithm. Input: Connected labeled graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$. **Output:** Rooted spanning tree (T, v_1) .

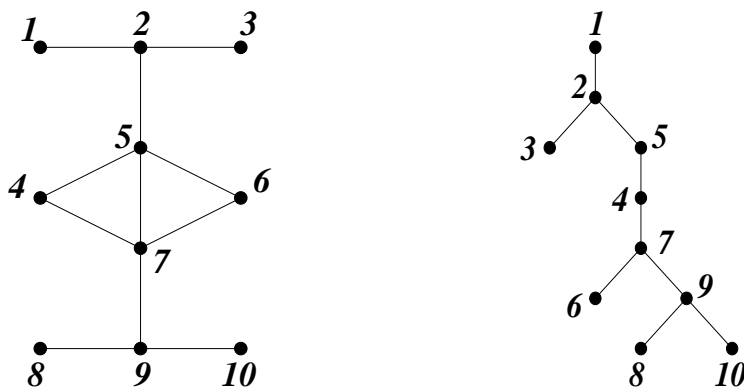
STEP 1. Assign v_1 to a vertex variable v and initialize a rooted tree (T, v_1) consisting of just one vertex.

STEP 2. Select the smallest subscript i ($2 \leq i \leq n$) such that $(v, v_i) \in E$ and v_i has not been visited. If no such subscript is found, go to **STEP 3**. If such subscript exists, attach the edge (v, v_i) to T and assign v_i to v , then return to **STEP 2**.

STEP 3. If $v = v_1$, **STOP**. The tree T is a spanning rooted tree.

STEP 4. If $v \neq v_1$, then backtrack from v to its parent u , assign u to v and return to **STEP 2**.

Example 9.1. A rooted spanning tree for the graph below by DFS Algorithm.



10 Breadth-First Search

Breadth-First Search (BFS) Algorithm. **Input:** Connected graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$. **Output:** Rooted spanning tree (T, v_1) .

STEP 1. Start with a sequence $Q = v_1$ and initialize a rooted tree (T, v_1) consisting of just one vertex.

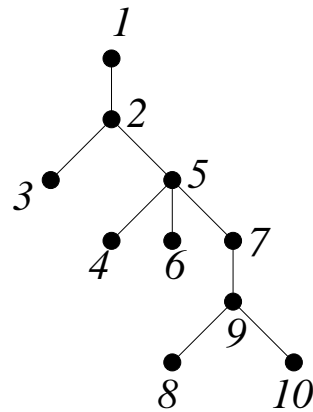
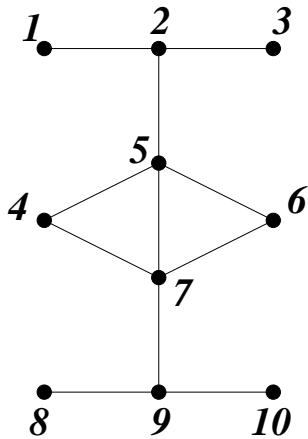
STEP 2. If Q is an empty sequence, then **STOP**.

STEP 3. If Q is nonempty, then delete the vertex v in the front of Q .

STEP 4. If there are vertices w (not yet visited) adjacent to v (the same v in **STEP 3**), then attach all edges (v, w) to T , add all vertices w to the end of Q , and return to **STEP 2**. If there are no vertices adjacent with v that are not yet visited, then return to **STEP 2**.

Example 10.1. A rooted tree for the graph below by BFS Algorithm.

Solution. $Q = 1 \Rightarrow Q = 2 \Rightarrow Q = 3, 5 \Rightarrow Q = 5 \Rightarrow Q = 4, 6, 7 \Rightarrow Q = 6, 7 \Rightarrow Q = 7 \Rightarrow Q = 9 \Rightarrow Q = 8, 10 \Rightarrow Q = 10 \Rightarrow Q = \emptyset$.

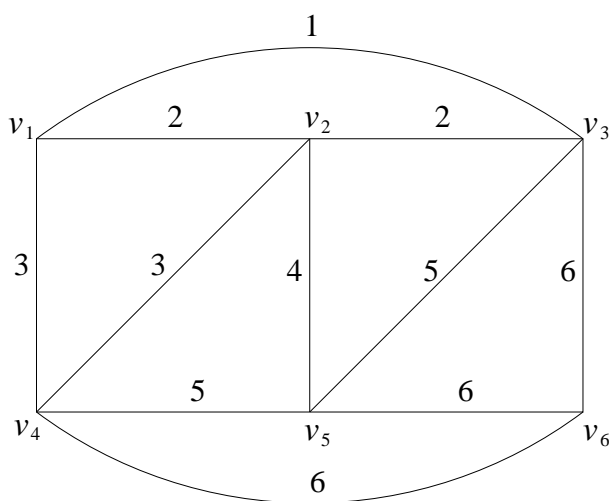


11 Minimum Spanning Tree

A graph $G = (V, E)$ together with a **weight function** $w : E \rightarrow \mathbb{R}$ is called a **weighted graph**. For each edge $e \in E$, the value $w(e)$ is called the **weight** of the edge e . The **weight** of G is the value

$$w(G) = \sum_{e \in E} w(e).$$

If G is connected, then G contains a spanning tree. A **minimum spanning tree (MST)** of a weighted graph G is a spanning tree whose weight is minimum among all spanning trees. We introduce two algorithms to find a minimum spanning tree for a connected weighted graph.



Theorem 11.1 (Kruskal's Algorithm). **Input:** $G = (V, E)$ with $|V| = n$, $w : E \rightarrow \mathbb{R}$. **Output:** MST of G or no spanning tree.

STEP 1. Choose an edge e of G such that $w(e)$ is smallest in E . Initialize a spanning forest F consisting of this single edge e .

STEP 2. If $|E(F)| = n - 1$, **STOP**. (F is an MST of G .)

STEP 3. If $|E(F)| < n - 1$ and $E \setminus E(F) = \emptyset$, **STOP**. (No spanning tree)

STEP 4. If $|E(F)| < n - 1$ and $E \setminus E(F) \neq \emptyset$, then select an edge $e \notin F$ such that

$$w(e) = \min\{e' \notin E(F) : F \cup e' \text{ has no cycle}\},$$

add e to F and return to **STEP 2**.

If $\{e \notin E(F) : F \cup e \text{ has no cycle}\} = \emptyset$, **STOP**. (No spanning tree)

Proof. Whenever G is disconnected, it is clear that the algorithm stops either in **STEP 3** or in the latter case of **STEP 4** with conclusion of non-existence of spanning tree. We assume that G is connected.

Let the edges of G be labeled as e_1, e_2, \dots, e_m and be linearly ordered as

$$w(e_1) \leq w(e_2) \leq \dots \leq w(e_m).$$

Let F_1 denote the forest consisting of the single edge e_1 . Let F_k denote the forest obtained by the algorithm when the k th edge e_k is inspected, where $k \geq 2$, i.e.,

$$F_k = \begin{cases} F_{k-1} & \text{if } F_{k-1} \cup e_k \text{ contains a cycle,} \\ F_{k-1} \cup e_k & \text{if } F_{k-1} \cup e_k \text{ contains no cycle.} \end{cases}$$

We claim that F_k is contained in an MST for each $k \geq 1$ by induction on k .

For $k = 1$, let $e_1 = uv$. Let T_0 be an MST. If $e_1 \in T_0$, nothing is to be proved. If $e_1 \notin T_0$, then $T_0 \cup e_1$ contains a unique cycle C . Obviously,

$$w(e_1) \leq w(e) \quad \text{for all } e \in E(C).$$

Let e be an edge of C other than e_1 . Then the tree $T_1 = (T_0 \cup e_1) \setminus e$ is an MST containing F_1 , since T_1 has the weight

$$w(T_1) = w(T_0) + w(e_1) - w(e) \leq w(T_0).$$

Assume that the forest F_{k-1} is contained in an MST T_{k-1} . Our aim is to construct an MST T_k containing F_k . We divide the situation into two cases.

CASE 1: $e_k \notin F_k$. This means that e_k is not selected at the time when e_k is inspected, i.e., $F_{k-1} \cup e_k$ contains a cycle. Then $F_k = F_{k-1}$. Thus $T_k = T_{k-1}$ is an MST containing F_k .

CASE 2: $e_k \in F_k$. This means that e_k is selected at the time when e_k is inspected, and $F_k = F_{k-1} \cup e_k$. There are two subcases.

SUBCASE 1: $e_k \in T_{k-1}$. Then $T_k = T_{k-1}$ is an MST containing F_k .

SUBCASE 2: $e_k \notin T_{k-1}$. Then $T_{k-1} \cup e_k$ contains a unique cycle C . Clearly, the cycle C is not contained in the forest F_k . So $E(C \setminus F_k) \neq \emptyset$. We claim that $w(e_k) \leq w(e^*)$ for all $e^* \in E(C \setminus F_k)$.

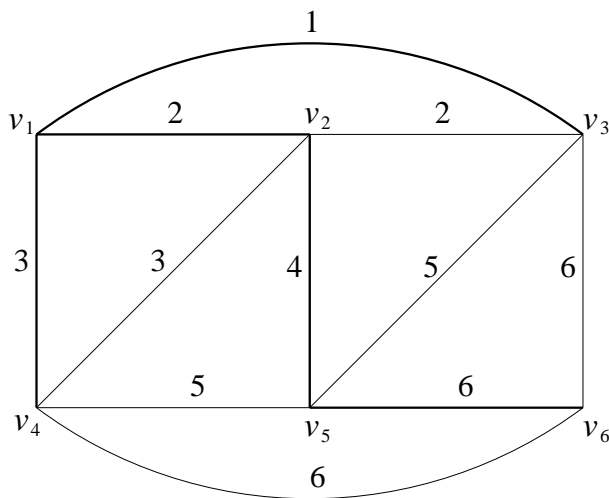
Suppose there exists an edge $e^* \in E(C \setminus F_k)$ such that $w(e^*) < w(e_k)$. Then e^* has been inspected before e_k and is not selected, so $e^* = e_i$ for some

$i < k$. Thus $F_{i-1} \cup e_i$ contains a cycle. Of course, $F_{k-1} \cup e_i$ contains a cycle by $F_{i-1} \subseteq F_{k-1}$. Since $e_i = e^* \in C \subseteq T_{k-1} \cup e_k$, we see that $F_{k-1} \cup e^*$ is contained in $T_{k-1} \cup e_k$. It follows that the cycle in $F_{k-1} \cup e^*$ must be the same cycle C . So C is the unique cycle of $F_{i-1} \cup e_i$. Since $e_k \neq e_i = e^*$, it follows that $e_k \in F_{i-1} \subseteq F_{k-1}$, which is contradictory to $e_k \notin F_{k-1}$.

Now we fix an edge $e^* \in E(C \setminus F_k)$ and define a tree $T_k = (T_{k-1} \cup e_k) \setminus e^*$. Since $w(e^*) \geq w(e_k)$, we see that

$$w(T_k) = w(T_{k-1}) + w(e_k) - w(e^*) \leq w(T_{k-1}).$$

Hence T_k is an MST and it contains $F_k = F_{k-1} \cup e_k$. □



Theorem 11.2 (Prim's Algorithm). **Input:** $G = (V, E)$, $|V| = n$, $w : E \rightarrow \mathbb{R}$. **Output:** An MST tree of G or no spanning tree.

STEP 1. Choose a vertex v_0 of G . Initialize a subtree T consisting of v_0 .

STEP 2. If $|V(T)| = n$, **STOP**. (T is an MST of G .)

STEP 3. If $|V(T)| < n$ and there is no edge from $V(T)$ to $V \setminus V(T)$, **STOP**. (There is no spanning tree.)

STEP 4. If $|V(T)| < n$ and there are edges from $V(T)$ to $V \setminus V(T)$, select an edge e from $V(T)$ to $V \setminus V(T)$ such that

$$w(e) = \min\{w(x) : x = uv, u \in V(T), v \in V \setminus V(T)\},$$

add the edge e with endpoints to T to have a subtree $T = T \cup e$, and return to **STEP 2**.

Proof. If G is disconnected, the algorithm stops in **STEP 3** and there is no spanning tree. Assume that G is connected. It suffices to show that the subtrees T , which are constructed in **STEP 4**, are always contained in some MSTs. We proceed by induction on $|V(T)|$.

At the beginning when a vertex v_0 is selected, $|V(T)| = 1$, it is obvious that $T = (v_0, \emptyset)$ is contained in every MST. Assume that T is contained in an MST T_1 and $|V(T)| < n$. Let e^* be an edge from $V(T)$ to $V \setminus V(T)$ such that

$$w(e^*) = \min\{w(e) : e = uv, u \in T, v \notin T\}.$$

If $e^* \in T_1$, nothing is to be proved. If $e^* \notin T_1$, then $T_1 \cup e^*$ contains a unique cycle C , and $e^* \in C$. We must have $w(e_1) \leq w(e^*)$ for all $e_1 \in C \setminus e^*$. Otherwise, if $w(e_1) > w(e^*)$ for an edge $e_1 \in C \setminus e^*$, the spanning tree $(T_1 \cup e^*) \setminus e_1$ would have weight

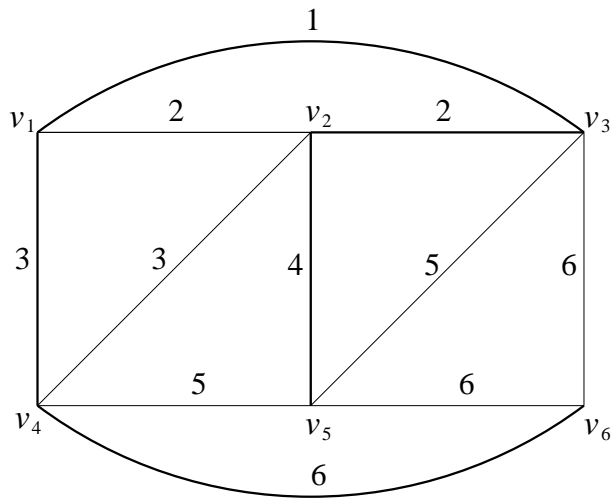
$$w(T_1 \cup e^* \setminus e_1) = w(T_1) + w(e^*) - w(e_1) < w(T_1).$$

This is contradictory to the minimality of T_1 .

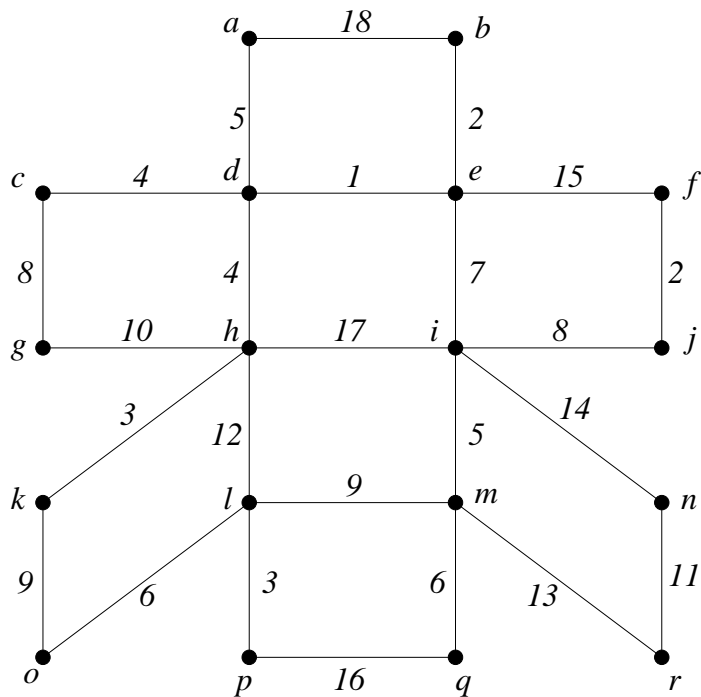
Since the cycle C intersects the cut $[V(T), V \setminus V(T)]$ (the set of all edges between $V(T)$ and $V \setminus V(T)$), the intersection $E(C) \cap [V(T), V \setminus V(T)]$ must contain an edge e_2 other than e^* . Then $w(e_2) \leq w(e^*)$ by the above argument. The minimality of $w(e^*)$ in $[V(T), V \setminus V(T)]$ implies $w(e_2) = w(e^*)$. Thus the spanning tree $T_2 = (T_1 \cup e^*) \setminus e_2$ has weight

$$w(T_2) = w(T_1) + w(e^*) - w(e_2) = w(T_1).$$

Hence T_2 is an MST and contains $T \cup e^*$. □



Example 11.1. Find a minimum spanning tree for the following graph.

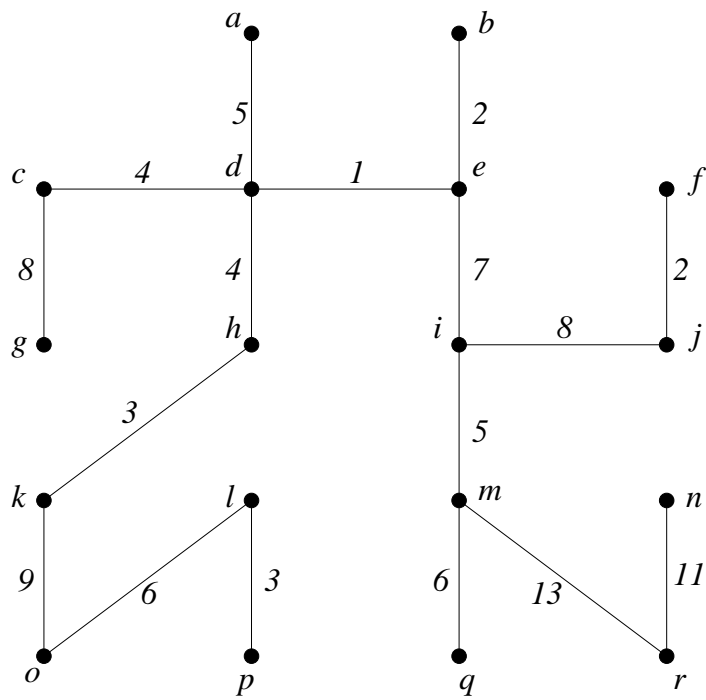


Solution. (i) Using Kruskal's Algorithm.

$$E(T) = \{(d, e), (b, e), (f, j), (h, k), (l, p), (c, d), (d, h), (a, d), (i, m), (l, o), (m, q), (e, i), (c, g), (i, j), (k, o), (n, r), (m, r)\}.$$

(ii) Using Prim's Algorithm.

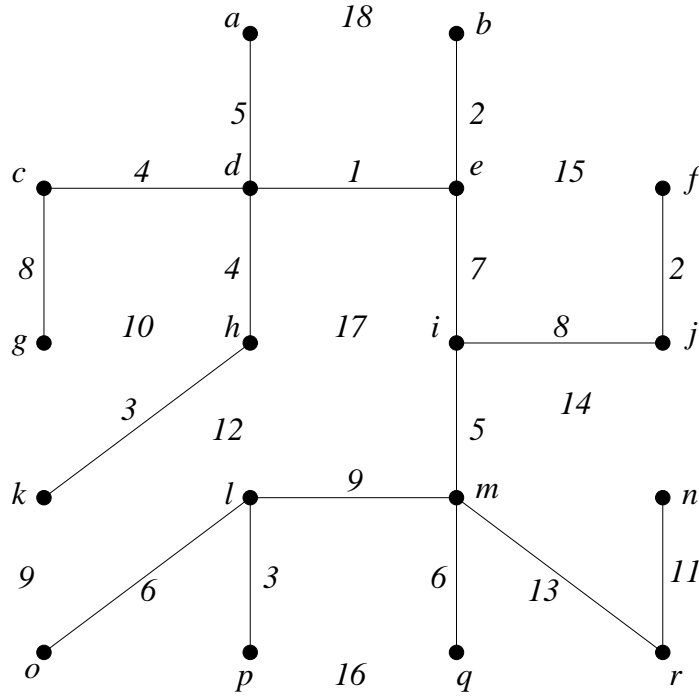
$$E(T) = \{(a, d), (d, e), (e, b), (d, c), (d, h), (h, k), (e, i), (i, m), (m, q), (c, g), (i, j), (j, f), (m, l), (l, p), (l, o), (m, r), (r, n)\}.$$



The weight of the minimum spanning tree is

$$\begin{aligned}
 w(T) &= 1 + 2 + 2 + 3 + 3 + 4 + 4 + 5 + 5 \\
 &\quad + 6 + 6 + 7 + 8 + 8 + 9 + 11 + 13 = 97.
 \end{aligned}$$

Example 11.2. Show that if a connected graph has distinct weights on its edges then its minimum spanning tree is unique.



First Proof. Let G be a connected graph whose weight function has distinct values on the edges. Suppose G has two distinct minimum spanning trees T_1 and T_2 . Let e_1^* be the minimum weight edge of T_1 among all edges not in T_2 , and let e_2^* be the minimum weight edge of T_2 among all edges not in T_1 , i.e.,

$$w(e_1^*) = \min\{w(e_1) : e_1 \in E(T_1 \setminus T_2)\},$$

$$w(e_2^*) = \min\{w(e_2) : e_2 \in E(T_2 \setminus T_1)\}.$$

Clearly, $e_1^* \neq e_2^*$. Then $T_1 \cup e_2^*$ contains a unique cycle C_1 and $e_2^* \in C_1$; $T_2 \cup e_1^*$ contains a unique cycle C_2 and $e_1^* \in C_2$. We have

$$w(e_2^*) > w(e_1) \quad \text{for all } e_1 \in T_1 \cap C_1.$$

Otherwise, $T_1^* = (T_1 \setminus e_1) \cup e_2^*$ would be a spanning tree having weight smaller than that of T_1 , which is a contradiction. Now, if there exists an edge $e_1 \in C_1 \setminus e_2^*$ such that $e_1 \notin E(T_2)$, then $e_1 \in E(T_1 \setminus T_2)$, thus $w(e_1^*) < w(e_1) < w(e_2^*)$; likewise, if there exists an edge $e_2 \in C_2 \setminus e_1^*$ such that $e_2 \notin E(T_1)$, then $w(e_2^*) < w(e_2) < w(e_1^*)$. We see that if both cases were valid, we would have the contradiction $w(e_1^*) < w(e_2^*) < w(e_1^*)$.

Therefore we must have either $e_1 \in E(T_2)$ for all $e_1 \in C_1 \setminus e_2^*$, or, $e_2 \in E(T_1)$ for all $e_2 \in C_2 \setminus e_1^*$. This means that either C_1 is contained in T_2 , or C_2 is contained in T_1 , contradicting to that none of T_1, T_2 contains a cycle.

Suppose $T_1 \neq T_2$. Let e^* be the smallest weight edge in $T_1 \Delta T_2$, i.e.,

$$w(e^*) = \min\{w(e) : e \in (T_1 \setminus T_2) \cup (T_2 \setminus T_1)\}.$$

Without loss of generality, we may assume $e^* \in T_1 \setminus T_2$. Then $T_2 \cup e^*$ contains a unique cycle C . There exists at least one edge $e' \in C$ such that $e' \notin T_1$; otherwise, C is contained in T_1 . Clearly, $e' \neq e^* \in T_1$ and $e' \in T_2 \setminus T_1$. Thus $w(e') > w(e^*)$ by the minimality of e^* in $T_1 \Delta T_2$. Hence $T' = (T_2 \cup e^*) \setminus e'$ is a spanning tree and $w(T') < w(T_2)$, which is contradictory to the minimality of $w(T_2)$.

Second Proof. It is true for connected graph without edge; the MST is the single vertex and the minimum weight is zero. Consider a connected graph G with some edges. Let e^* be the edge of maximum weight in G . If e^* is contained in a cycle of G , then every MST of G is an MST of $G \setminus e^*$. By induction, the MST for $G \setminus e^*$ is unique, so is for G . If e^* is not on a cycle, i.e., e^* is a bridge of G , then e^* must be contained in every MST of G . Let G_1, G_2 denote connected components of $G \setminus e^*$, and T_1^*, T_2^* the unique MSTs of G_1, G_2 respectively. Then $T^* = T_1^* \cup T_2^* \cup e^*$ is a spanning tree of G .

Let T be an MST of G . Then $T \setminus e^*$ is decomposed into spanning trees T_1, T_2 of G_1, G_2 respectively. Clearly, $w(T_1^*) \leq w(T_1)$ and $w(T_2^*) \leq w(T_2)$. Then

$$w(T^*) = w(T_1^*) + w(T_2^*) + w(e^*) \leq w(T_1) + w(T_2) + w(e^*) = w(T).$$

The minimality of $w(T)$ implies $w(T^*) = w(T)$. Consequently, $w(T_1^*) = w(T_1)$ and $w(T_2^*) = w(T_2)$. By induction, $T_1^* = T_1$ and $T_2^* = T_2$. Hence $T = T^*$, the MST of G is unique.

Third Proof. Applying Two Facts:

(a) The minimum weight edge e_0 must be used by any MST. Let F denote the spanning forest consisting of the single edge e_0 .

(b) If F is a spanning forest whose edges are used by any MST, then the edge e^* such that

$$w(e^*) = \min\{e \in G : F \cup e \text{ contains no cycle}\}$$

must be used by any MST. □

12 Planar Graphs

Definition 12.1. A graph G is said to be **planar** if it can be drawn on a plane in such a way that no two edges cross each other.

For instance, the complete graph K_4 , the bipartite complete graph $K_{2,n}$, and the cubic graph Q_3 are planar. They can be drawn on a plane without crossing edges; see Figure 10. However, by try-and-error, it seems that the complete graph K_5 and the complete bipartite graph $K_{3,3}$ are not planar. We will prove this fact rigorously by using the Euler Formula in Theorem 4.

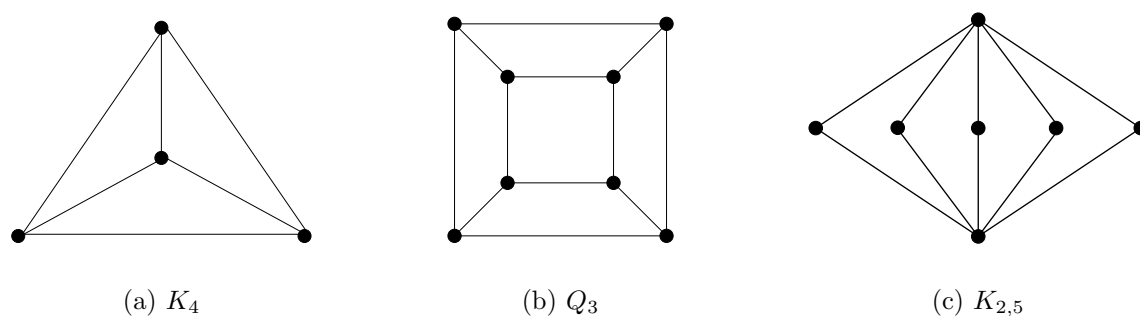


Figure 10: Planar Graphs

Planar graphs are important because in the design of circuits on a board we wish to have no routes crossing each other, or to keep the crossing routes as fewer as possible. Given a connected planar graph G and draw it on a plane without crossing edges; the plane is divided into some **regions** (or **faces**) of G ; we denote by $r(G)$ the number of regions. Strictly speaking, r may depend on the ways of drawing G on the plane. Nevertheless, we will see that $r(G)$ is actually independent of the ways of drawing G on any plane. The relation

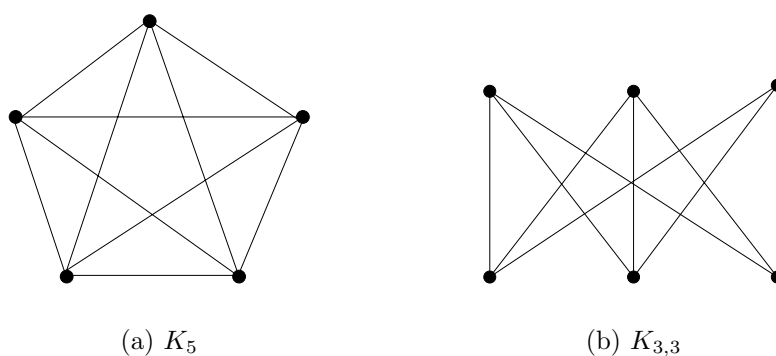


Figure 11: Nonplanar Graphs

between $r(G)$ and the number of vertices and the number of edges of G are related by the well-known Euler formula in the following theorem.

Theorem 12.2. (Euler Formula) *If G is a connected planar multigraph with v vertices, e edges, and r regions, then*

$$v - e + r = 2. \quad (4)$$

Proof. We prove the statement by induction on the number of edges. When $e = 0$, the graph G must be a single vertex, and in this case the number of regions is one. Obviously, $v - e + r = 2$.

Assume that the theorem is true for all connected planar graphs with k edges. We consider a connected planar graph G with $k + 1$ edges. The graph G may or may not contain cycles. If G contains no cycle, then it is a tree. Thus $e = v - 1$ and $r = 1$, so $v - e + r = 2$. If G contains cycles, choose a cycle C and an edge $x \in C$, remove x from G to obtain a new planar graph G' . Since x is on a cycle, G' is still connected and has the same vertices of G , but has k edges. By induction we have $v' - e' + r' = 2$. Note that x bounds two regions, one is inside C and the other is outside C . So $v' = v$, $e' = e - 1$, $r' = r - 1$. It follows that $v - e + r = 2$. \square

Planar graphs can be equivalently described as graphs drawn on a sphere without edges crossing each other. Note that the boundary of any polyhedron is homeomorphic to a sphere. A **polyhedron graph** consists of vertices and edges of a polyhedron, and is a planar graph. The first two graphs in Figure 10 are polyhedron graphs, coming from the boundary of a tetrahedron and a cube. The three graphs in Figure 12 are polyhedron graphs from octahedron, dodecahedron and icosahedron, respectively.

For a connected planar graph G drawn on the plane, let D_1, D_2, \dots, D_r denote the regions of G , and let $e(D_i)$ denote the number of edges bounding the region D_i . If one counts the edges along the boundary of each region of G , every edge of G is counted exactly twice. This means that

$$\sum_{i=1}^r e(D_i) = 2e(G). \quad (5)$$

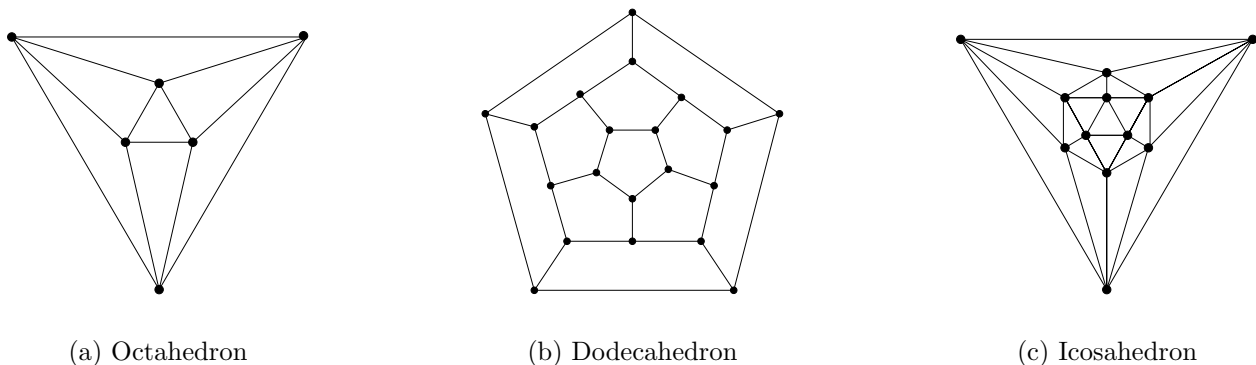


Figure 12: Regular Polyhedra

If G contains no multiple edges and loops, then $e(D_i) \geq 3$. We thus obtain

$$3r(G) \leq 2e(G).$$

Combining the inequality and the Euler formula (4), we obtain the following corollary.

Corollary 12.3. *If G is a connected, simple, planar graph with v vertices and e edges, then*

$$e \leq 3v - 6. \tag{6}$$

Example 12.1. The complete graph K_5 is not planar.

Proof. Notice that K_5 has 5 vertices and $\binom{5}{2}$ ($= 10$) edges. Suppose K_5 is planar. Then by Corollary 12.3, we have $10 \leq 3 \cdot 5 - 6 = 9$, a contradiction. \square

Example 12.2. The complete bipartite graph $K_{3,3}$ is not planar.

Proof. Note that every region of a graph bounds a cycle. Since $K_{3,3}$ is bipartite, it has only cycles of even length. Since there is no cycles of length 2, every cycle of $K_{3,3}$ has length at least 4. Suppose $K_{3,3}$ is a planar graph. Then every region of $K_{3,3}$ bounds at least 4 edges. By the equation (5), we have

$$4r \leq 2e.$$

Combine this inequality and the Euler Formula (4). We obtain

$$e \leq 2v - 4.$$

Since $v(K_{3,3}) = 6$ and $e(K_{3,3}) = 9$, we have $9 \leq 2 \cdot 6 - 4 = 8$, which is a contradiction. \square

Example 12.3. The **Petersen graph** is not planar. See Figure 13.

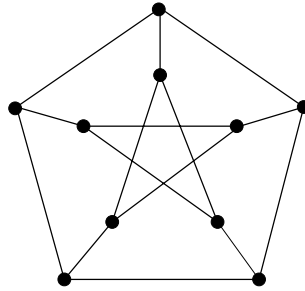


Figure 13: Petersen Graph

Proof. Note that each cycle of the Petersen graph has at least 5 edges. So if it is planar, then $5r \leq 2e$. It follows from the Euler formula that $3e \leq 5v - 10$. However, the graph has 10 vertices and 15 edges. Thus $45 = 3 \cdot 15 \leq 5 \cdot 10 - 10 = 40$, a contradiction. \square

13 Classification of Regular Polyhedra

A polyhedron is said to be **regular** if all vertices have the same degree and all faces have the same number of edges. Regular polyhedra are known as **Platonic solids**.

Theorem 13.1. *Let k be the number of edges of a regular polyhedron at a vertex, and l the number of edges bounding a face. Then*

$$(k, l) \in \{(3, 3), (3, 4), (3, 5), (4, 3), (5, 3)\}$$

and $(v, e, f) = (4, 6, 4), (8, 12, 6), (20, 30, 12)$.

Proof. Note that $k \geq 3$, $l \geq 3$, and

$$kv = 2e = lf, \quad v - e + f = 2.$$

Then $e = kv/2$, $f = kv/l$. Put them into the Euler Formula to have

$$v - kv/2 + kv/l = 2, \quad \text{i.e.,} \quad \frac{2l - kl + 2k}{2l}v = 2.$$

Thus

$$v = \frac{4l}{2k + 2l - lk}, \quad e = \frac{kv}{2}, \quad f = \frac{kv}{l}.$$

For $k = 3$, we have $v = \frac{4l}{6-l}$. Since $l \geq 3$, there are three choices $l = 3, 4, 5$. It follows that

$$(v, e, f) = (4, 6, 4), (8, 12, 6), (20, 30, 12).$$

For $k = 4$, we have $v = \frac{4l}{8-2l}$. There is only one choice, i.e., $l = 3$. Thus

$$(v, e, f) = (6, 12, 8).$$

For $k = 5$, we have $v = \frac{4l}{10-3l}$. There is only one choice, i.e., $l = 3$. We have

$$(v, e, f) = (12, 30, 20).$$

For $k \geq 6$, we have $v = \frac{4l}{2k+2l-kl}$. Since $l \geq 3$, we see that

$$2k + 2l - kl = 2k - l(k - 2) \leq 2k - 3(k - 2) = -k + 6 \leq 0.$$

This means that there is no choice for v . □

A **spherical fullerene** is a simple 3-regular planar graph whose faces are only pentagons and hexagons. We shall see that every spherical fullerene has exactly 12 pentagons.

Proposition 13.2. *If a simple 3-regular planar graph has only pentagons and hexagons, then the number of pentagons must be 12.*

Let f_5, f_6 denote the number of pentagons and hexagons of a fullerene respectively. The vertex-edge relation, the edge-face relation, and the Euler Formula are stated by the equations

$$3v = 2e = 5f_5 + 6f_6, \quad v - e + f_5 + f_6 = 2.$$

Then $e = 3v/2$ implies

$$5f_5 + 6f_6 = 3v, \quad f_5 + f_6 = 2 + v/2.$$

Since $6f_5 + 6f_6 = 12 + 3v$, we obtain $f_5 = 12$. Thus

$$v = 20 + 2f_6, \quad e = 30 + 3f_6, \quad f_5 = 12, \quad f_6 = n \text{ (free)}.$$

For each integer $n \geq 0$, there exist crystal structures corresponding to the fullerene with n hexagons. The buckyball (carbon sixty C_{60} , football or soccer) has a fullerene structure of $f_6 = 20$. The buckyball has an extra property that each pentagon is surrounded by 6 hexagons, and each hexagon is surrounded by 3 pentagons and 3 hexagons. This extra condition leads to the relation

$$5f_5 = 3f_6.$$

14 Matching

Let $G = (V, E)$ be a bipartite graph, i.e., $V = X \cup Y$, $X \cap Y = \emptyset$, $X \neq \emptyset \neq Y$, and $E(G) \subset X \times Y$. A **matching** of G is a subset of $E(G)$ such that no two edges share a common vertex in X or Y . A matching is called **complete** if every vertex of X is an end vertex of an edge of the matching.

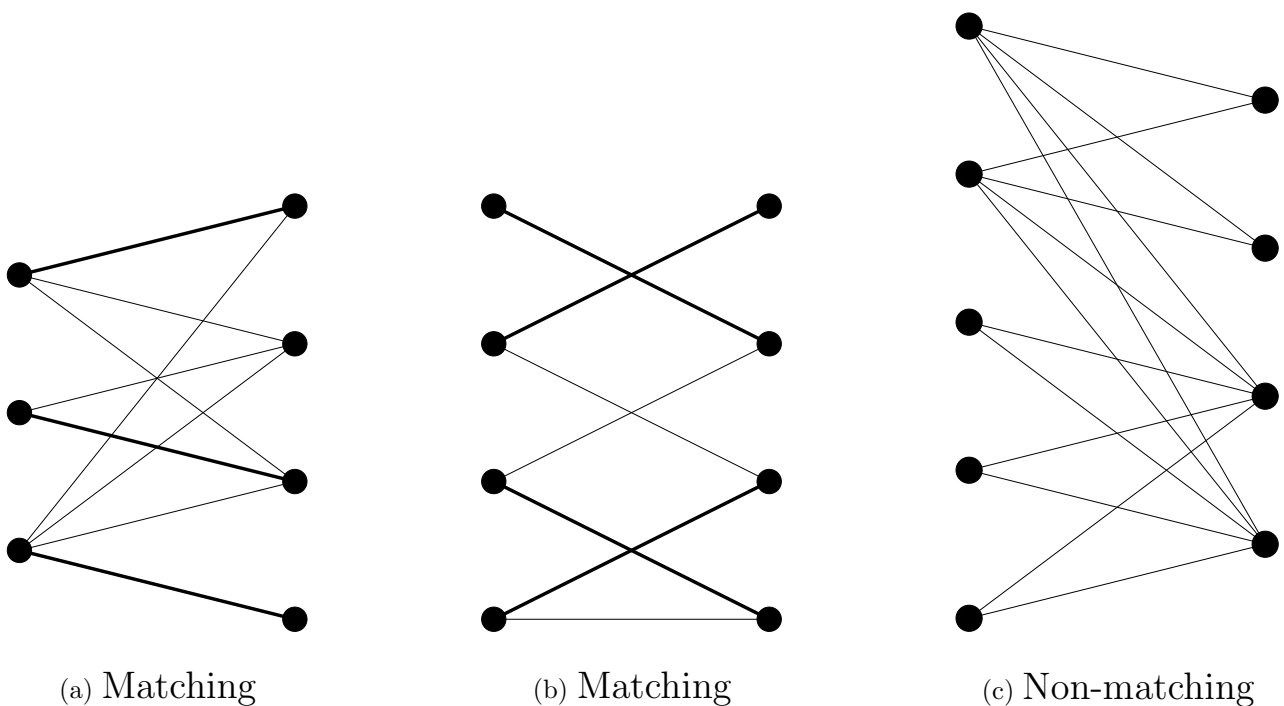


Figure 14: Matchings and non-matching

Theorem 14.1. *Let R be the binary relation on from X to Y for the bipartite graph $G = (V, E)$ with $V = X \cup Y$. Then G has a **complete matching** if and only if for any $A \subseteq X$, $|A| \leq |R(A)|$.*

Proof. Let $X = \{x_1, x_2, \dots, x_m\}$ and $Y = \{y_1, y_2, \dots, y_n\}$. We construct a network $N = (V, A)$ as follows:

$$c(s, x_i) = 1; \quad c(y_j, t) = 1; \quad c(x_i, y_j) = M \text{ for } (x_i, y_j) \in E$$

where M is an integer larger than $|X|$.

Note that G has a complete matching $\iff N$ has a maximal flow that uses all edges (s, x_i) , $1 \leq i \leq m$. \square

15 Markov Chain and Page Rank

Page rank is an analysis of measuring importance of objects connected or linked such as web sites linked by hyperlinks. The system of objects linked in some way is usually a directed graph (or digraph) $G = (V, E)$ with vertex set V and directed edge set E . Let $V = \{v_1, v_2, \dots, v_n\}$. Mathematically, a **page rank** is just a function $r : V \rightarrow [0, 1]$ determined by the digraph G such that $\sum_{j=1}^n r(v_j) = 1$, the value $r(v_j)$ is the relative importance of the vertex v_j .

Let $\text{ideg}_G(v)$ denote the in-degree of a vertex v in G and $\text{odeg}_G(v)$ the out-degree. It is natural to assume that the importance of a vertex v_j depends on hyperlinks from each vertex v_i to v_j and the importance of v_i . For each vertex v_i , the importance of v_i to v_j is proportional to $r(v_i)$, and this proportion is the number of links from v_i to v_j out of all out links from v_i . Thus we require

$$r(v_j) = \sum_{i=1, i \neq j}^n \min\{\text{odeg}(v_i), a_{ij}/\text{odeg}(v_i)\}r(v_i), \quad j = 1, \dots, n \quad (7)$$

where a_{ij} is the number of links from v_i to v_j . We may assume that for each site v_i is linked to at least one site v_{j_0} (i.e. $\text{odeg}_G(v_i) > 0$), and each site v_j is linked from at least one site v_{i_0} (i.e. $\text{ideg}_G(v_j) > 0$). Let $r_i = r(v_i)$ and

$$p_{ij} = \min\{\text{odeg}(v_i), a_{ij}/\text{odeg}(v_i)\}.$$

We have the vector $\mathbf{r} = [r_1, \dots, r_n]$ and $n \times n$ matrix $P = [p_{ij}]$. We have

$$\sum_{j=1}^n p_{ij} = 1, \quad i = 1, \dots, n.$$

The condition $\text{odeg}_G(v_i) > 0$ means that the page v_i is linked to web site and we only consider the ranks of web sites linked to the web system. The system of PageRank equations becomes

$$\mathbf{r} = \mathbf{r}P. \quad (8)$$

Theorem 15.1 (Fundamental Theorem of Markov Chain). *Let \mathbf{P} be the transition matrix of a Markov chain. Then there exists a unique stationary distribution $\boldsymbol{\pi}$, i.e., $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$.*

Proof. Given an initial distribution vector \mathbf{p}_0 . The k step distribution is

$$\mathbf{p}_t = \mathbf{p}_{t-1}\mathbf{P} = \mathbf{p}_0\mathbf{P}^t, \quad t \geq 1.$$

Let $\mathbf{1}$ denote the vector of constant coordinates 1. Since $\mathbf{P}\mathbf{1} = \mathbf{1}$, we see that

$$\langle \mathbf{p}_t, \mathbf{1} \rangle = \mathbf{p}_0\mathbf{P}^t\mathbf{1} = \mathbf{p}_0\mathbf{1} = \langle \mathbf{p}_0, \mathbf{1} \rangle = 1.$$

So all \mathbf{p}_t are distributions for $t \geq 1$. Consider the average distribution

$$\mathbf{a}_t = (\mathbf{p}_0 + \mathbf{p}_1 + \cdots + \mathbf{p}_{t-1})/t = \mathbf{p}_0(\mathbf{I} + \mathbf{P} + \cdots + \mathbf{P}^{t-1})/t.$$

Note that $\mathbf{a}_t(\mathbf{P} - \mathbf{I}) = (\mathbf{p}_t - \mathbf{p}_0)/t \rightarrow \mathbf{0}$ ($t \rightarrow \infty$) and

$$\mathbf{a}_t[\mathbf{P} - \mathbf{I}, \mathbf{1}] = [(\mathbf{p}_t - \mathbf{p}_0)/t, \mathbf{1}].$$

Recall $\text{rank}(\mathbf{P} - \mathbf{I}) = n - 1$. Let \mathbf{B} denote the $n \times n$ submatrix of $[\mathbf{P} - \mathbf{I}, \mathbf{1}]$ without its first column, and let \mathbf{c}_k denote the $(n - 1)$ -dimensional vector from $(\mathbf{p}_k - \mathbf{p}_0)/k$ by deleting its first entry. Then $\text{rank}(\mathbf{B}) = n$ and $\mathbf{a}_k\mathbf{B} = [\mathbf{c}_k, 1]$. Thus

$$\mathbf{a}_k = [\mathbf{c}_k, 1]\mathbf{B}^{-1} \rightarrow [\mathbf{0}, 1]\mathbf{B}^{-1} \quad (k \rightarrow \infty).$$

A **Markov chain** is a stochastic discrete model describing a sequence of events in which the probability of each event depends only on the state attained in the previous event. In the continuous case it known as **Markov process**. It is named by the Russian mathematician Andrey Markov himself.

A **discrete-time Markov chain** (DTMC) is a sequence of random variables X_0, X_1, X_2, \dots , satisfying the **Markov property**: the probability of moving to the next state depends only on the present state and not on the previous states. In math notation of conditional probability,

$$P(X_t = x \mid X_s = x_s, s < t) = P(X_t = x \mid X_{t-1} = x_{t-1}).$$

The possible values of X_t form a countable set S , called the **state space** of the chain. We usually assume $S = \{1, 2, \dots, n\}$ or $S = \mathbb{Z}_+ = \{1, 2, \dots\}$.

Markov chains are often described by directed graphs, where at each time t it is represented by a digraph whose vertex set is S and there is directed edges from a state i to a state j if the the **transition probability**

$$p_{ij}^{(t,t+1)} = P\{X_{t+1} = j | X_t = i\}$$

from state i to state j at time t is positive. The matrix $\mathbf{P}(t) = [p_{ij}(t)]$ is a stochastic matrix, called the **transition matrix** at time t , whose row sums are 1. We also have (s, t) -**transition probability** (from time s to time t)

$$p_{ij}^{(s,t)} = P\{X_t = j | X_s = i\}, \quad s < t$$

and (s, t) -transition matrix $\mathbf{P}^{(s,t)}$, $\mathbf{P}^{(t,t)} = \mathbf{I}$. For $s \leq r \leq t$,

$$P\{X_t = j | X_s = i\} = \sum_{k \in S} P\{X_t = j | X_r = k\} P\{X_r = k | X_s = i\}$$

This means that $p_{ij}^{(s,t)} = \sum_{k \in S} p_{ik}^{(s,r)} p_{kj}^{(r,t)}$. Then we have

$$\mathbf{P}^{(s,t)} = \mathbf{P}^{(s,r)} \mathbf{P}^{(r,t)}, \quad s \leq r \leq t.$$

The distribution of X_t is given by

$$P\{X_t = j\} = \sum_{i \in S} P\{X_t = j | X_{t-1} = i\} P\{X_{t-1} = i\}.$$

Let $\mathbf{p}_t = (P\{X_t = j\} : j \in S)$ denote a row vector index by S . Then

$$\mathbf{p}_t = \mathbf{p}_{t-1} \mathbf{P}^{(t-1,t)} = \mathbf{p}_0 \mathbf{P}^{(0,1)} \mathbf{P}^{(1,2)} \dots \mathbf{P}^{(t-1,t)}.$$

A Markov chain is **time-homogeneous** if $\mathbf{P}^{(t-1,t)}$ is a constant matrix \mathbf{P} , independent of time t . From now on we assume all Markov chains are time-homogeneous.

A set of distributions on the state space S is a simplex Δ of \mathbb{R}^n spanned by the coordinate unit vectors $\mathbf{e}_1, \dots, \mathbf{e}_n$. The transition matrix \mathbf{P} acts on Δ . A distribution $\boldsymbol{\pi}$ is **stationary** if $\boldsymbol{\pi} = \boldsymbol{\pi} \mathbf{P}$, which is a left eigenvector with eigenvalue 1 and lies in the simplex spanned by the coordinate

The **period** of a state j is the positive integer

$$d_j = \gcd\{t \in \mathbb{Z}_+ : p_{jj}^{(t)} > 0\}.$$

State j is **aperiodic** if $d_j = 1$ and **periodic** if $d_j > 1$.

The **hitting time** of an event $A \subseteq S$ is the first time that the chain is in A , i.e.,

$$T_A = \min\{t \in \mathbb{Z}_+ : X_t \in A\},$$

which is also a random variable. The **hitting probability** of A from state i is

$$h_{iA} = P\{T_A < \infty | X_0 = i\} = P\left(\bigcup_{t=1}^{\infty} \{X_t \in A | X_0 = i\}\right).$$

The **hitting probability** of state j from state i is

$$h_{ij} = P\{T_j < \infty | X_0 = i\} = P\left(\bigcup_{t=1}^{\infty} \{X_t = j | X_0 = i\}\right),$$

and **t -step hitting probability** of state j from state i is the probability of reaching state j from state i for the first time in t steps, i.e.,

$$\begin{aligned} h_{ij}^{(t)} &= P\{T_j = t | X_0 = i\} \quad (t \geq 1) \\ &= P\{X_1 \neq j, \dots, X_{t-1} \neq j, X_t = j | X_0 = i\}. \end{aligned} \quad (9)$$

Definition 15.2 (Recurrence and transience). A state $j \in S$ of a discrete-time Markov chain is **recurrent** (or persistent) if it returns to state j with probability 1, i.e.,

$$h_{jj} = P\{T_j < \infty | X_0 = j\} = \sum_{t=1}^{\infty} h_{jj}^{(t)} = 1.$$

If $h_{jj} < 1$, the state j is said to be **transient**, i.e., it has positive probability of never return to state j .

A state j is said to be **accessible** from a state i , denoted $i \rightarrow j$, if there exists an integer $t \geq 0$ (depending on i and j) such that

$$p_{ij}^{(t)} = P(X_t = j | X_0 = i) > 0,$$

A state i is said to **communicate** with a state j , denoted $i \leftrightarrow j$, if $i \rightarrow j$ and $j \rightarrow i$. Communication is an equivalence relation on the state space; the equivalence classes are known as **communication classes**. A communicating class is **closed** if the probability of leaving the class is zero. A Markov chain is **irreducible** if it communicates from any state to any another state, i.e., there is exactly one communicating class.

Definition 15.3 (Closed communication classes). A communication class C for a Markov chain is said to be **closed** if it never leaves the class once get in, i.e., $\sum_{j \in C} p_{ij} = 1$ for all $i \in C$. If C has only finitely many elements then C is closed if the submatrix of transition probabilities restricted to C has all row sums equal to 1.

Theorem 15.4 (Hitting probability formula). *The hitting probability h_{ij} from a state i to a state j in a time-homogeneous discrete Markov chain is given by the transition probabilities as follows:*

$$h_{ij} = \lim_{T \rightarrow \infty} \frac{\sum_{t=1}^T p_{ij}^{(t)}}{1 + \sum_{t=1}^T p_{jj}^{(t)}}.$$

In particular,

$$h_{jj} = \begin{cases} \sum_{t=1}^{\infty} p_{jj}^{(t)} / (1 + \sum_{t=1}^{\infty} p_{jj}^{(t)}) < 1 & \text{if } \sum_{t=1}^{\infty} p_{jj}^{(t)} < \infty \\ 1 & \text{if } \sum_{t=1}^{\infty} p_{jj}^{(t)} = \infty \end{cases}$$

Proof. The n -step transition probability can be written as

$$p_{ij}^{(t)} = P\{X_t = j | X_0 = i\} = \sum_{s=1}^t P\{X_t = j, T_j = s | X_0 = i\}.$$

Note that $T_j = s$ is equivalent to $X_1 \neq j, X_2 \neq j, \dots, X_{s-1} \neq j, X_s = j$. Then

$$\begin{aligned} P\{X_t = j, T_j = s | X_0 = i\} &= \frac{P\{X_t = j, T_j = s, X_0 = i\}}{P\{X_0 = i\}} \\ &= \frac{P\{X_t = j, T_j = s, X_0 = i\}}{P\{T_j = s, X_0 = i\}} \cdot \frac{P\{T_j = s, X_0 = i\}}{P\{X_0 = i\}} \\ &= P\{X_t = j | T_j = s, X_0 = i\} P\{T_j = s | X_0 = i\}. \end{aligned}$$

Since $P\{X_t = j | T_j = s, X_0 = i\} = P\{X_t = j | X_s = j\}$, it follows that

$$P\{X_t = j, T_j = s | X_0 = i\} = p_{jj}^{(t-s)} h_{ij}^{(s)};$$

consequently, $p_{ij}^{(t)} = \sum_{s=1}^t p_{jj}^{(t-s)} h_{ij}^{(s)}$. We have

$$\begin{aligned} \sum_{t=1}^T p_{ij}^{(t)} &= \sum_{t=1}^T \sum_{s=1}^t p_{jj}^{(t-s)} h_{ij}^{(s)} \\ &= \sum_{s=1}^T h_{ij}^{(s)} \sum_{t=s}^T p_{jj}^{(t-s)} \\ &= \sum_{s=1}^T h_{ij}^{(s)} \sum_{r=0}^{T-s} p_{jj}^{(r)} \end{aligned} \quad (10)$$

$$\leq \sum_{s=1}^T h_{ij}^{(s)} \sum_{r=0}^T p_{jj}^{(r)}. \quad (11)$$

Choose positive integers $S \leq T$, we obtain from (10)

$$\sum_{t=1}^T p_{ij}^{(t)} \geq \sum_{s=1}^S h_{ij}^{(s)} \sum_{r=0}^{T-s} p_{jj}^{(r)} \geq \sum_{s=1}^S h_{ij}^{(s)} \sum_{r=0}^{T-S} p_{jj}^{(r)}. \quad (12)$$

Divide both sides of (11) and (12) by $\sum_{r=0}^T p_{jj}^{(r)}$, we obtain

$$\sum_{s=1}^T h_{ij}^{(s)} \geq \frac{\sum_{t=1}^T p_{ij}^{(t)}}{\sum_{r=0}^T p_{jj}^{(r)}} \geq \sum_{s=1}^S h_{ij}^{(s)} \cdot \frac{\sum_{r=0}^{T-S} p_{jj}^{(r)}}{\sum_{r=0}^T p_{jj}^{(r)}}.$$

Note that $\sum_{r=0}^{T-S} p_{jj}^{(r)} = \sum_{r=0}^T p_{jj}^{(r)} - \sum_{r=T-S+1}^T p_{jj}^{(r)}$. We obtain

$$\sum_{s=1}^T h_{ij}^{(s)} \geq \frac{\sum_{t=1}^T p_{ij}^{(t)}}{1 + \sum_{r=1}^T p_{jj}^{(r)}} \geq \sum_{s=1}^S h_{ij}^{(s)} \left(1 - \frac{\sum_{r=T-S+1}^T p_{jj}^{(r)}}{\sum_{r=0}^T p_{jj}^{(r)}} \right). \quad (13)$$

Case 1. $\sum_{t=0}^{\infty} p_{jj}^{(t)} = \infty$. Then

$$\frac{\sum_{r=T-S+1}^T p_{jj}^{(r)}}{\sum_{r=0}^T p_{jj}^{(r)}} \leq \frac{S}{\sum_{r=0}^T p_{jj}^{(r)}} \rightarrow 0 \quad (T \rightarrow \infty).$$

In (13), let $T \rightarrow \infty$ first and $S \rightarrow \infty$ next, we have

$$h_{ij} \geq \frac{\sum_{t=1}^{\infty} p_{ij}^{(t)}}{\sum_{r=0}^{\infty} p_{jj}^{(r)}} \geq \sum_{s=1}^S h_{ij}^{(s)} \rightarrow \sum_{s=1}^{\infty} h_{ij}^{(s)} = h_{ij} \quad (S \rightarrow \infty).$$

Thus

$$h_{ij} = \frac{\sum_{n=1}^{\infty} p_{ij}^{(n)}}{1 + \sum_{k=1}^{\infty} p_{jj}^{(k)}}.$$

Case 2. $\sum_{t=0}^{\infty} p_{jj}^{(t)} < \infty$. Take $S = \lfloor T/2 \rfloor$, then

$$\frac{\sum_{r=T-S+1}^T p_{jj}^{(r)}}{\sum_{r=0}^T p_{jj}^{(r)}} = \frac{\sum_{r=\lfloor T/2 \rfloor + 1}^T p_{jj}^{(r)}}{1 + \sum_{r=1}^T p_{jj}^{(r)}} \rightarrow 0 \quad (T \rightarrow \infty).$$

Let $T \rightarrow \infty$ in (13), we have

$$h_{ij} \geq \frac{\sum_{t=1}^{\infty} p_{ij}^{(t)}}{1 + \sum_{r=1}^{\infty} p_{jj}^{(r)}} \geq h_{ij}, \quad \text{i.e.,} \quad h_{ij} = \frac{\sum_{t=1}^{\infty} p_{ij}^{(t)}}{1 + \sum_{r=1}^{\infty} p_{jj}^{(r)}}.$$

In particular, $h_{jj} = \frac{\sum_{t=1}^{\infty} p_{jj}^{(t)}}{1 + \sum_{r=1}^{\infty} p_{jj}^{(r)}} < 1$.

We see that $h_{jj} = 1$ if and only if $\sum_{k=1}^{\infty} p_{jj}^{(k)} = \infty$. □

Corollary 15.5 (Recurrence and transience criterion 1). *A state j is recurrent (transient) if and only if $\sum_{t=1}^{\infty} p_{jj}^{(t)} = \infty$ ($< \infty$).*

If a state i is recurrent and $i \rightarrow j$, then $j \rightarrow i$. In fact, suppose $j \not\rightarrow i$, i.e., there is no directed path from j to i . Then there is a way (through a directed path from i to j) leaving i and never return back to i . Thus $h_{ii} < 1$, contradicting to that i is recurrent.

Theorem 15.6 (Recurrence is a class property). *All states in a communication class are either all recurrent or all transient.*

Proof. Let i, j be communicating states each other. There exist some positive integers s, t such that $p_{ij}^{(s)}, p_{ji}^{(t)} > 0$. For each integer $r \geq 1$, $p_{ij}^{(s)} p_{jj}^{(r)} p_{ji}^{(t)}$ is the

probability of a closed directed walk of length $s + r + t$ from i to itself. Then

$$\sum_{u=1}^{\infty} p_{ii}^{(u)} \geq \sum_{r=1}^{\infty} p_{ii}^{(s+r+t)} \geq \sum_{r=1}^{\infty} p_{ij}^{(s)} p_{jj}^{(r)} p_{ji}^{(t)} = p_{ij}^{(s)} \left(\sum_{r=1}^{\infty} p_{jj}^{(r)} \right) p_{ji}^{(t)}. \quad (14)$$

If state i is transient, i.e., $\sum_{n=1}^{\infty} p_{ii}^{(n)} < \infty$, then $\sum_{m=1}^{\infty} p_{jj}^{(m)} < \infty$ by (14). Thus state j is also transient.

For each communication class, if one of its state is transient, so are the others. If no one is transient, then all states are recurrent. \square

Theorem 15.7 (Number of visits). *Let $N_j = \sum_{n=1}^{\infty} 1\{X_n = j\}$ denote the total number of visits of state j .*

(a) *If state j recurrent, then state j is visited infinitely many times with probability 1, i.e., $P\{N_j = \infty | X_0 = j\} = 1$.*

(b) *If state j is transient, then N_j follows a geometric distribution*

$$P\{N_j = n\} = q(1 - q)^n,$$

where $q = P\{T_j = \infty | X_0 = j\}$, the probability that state j is never back.

Proof. Note that $P\{N_j \geq 1 | X_0 = i\} = h_{ij}$, the hitting probability from state i to state j . Then

$$\begin{aligned} P\{N_j \geq n | X_0 = i\} &= \sum_{k=1}^{\infty} P\{N_j \geq n - 1 | X_0 = i\} P\{T_j = k | X_0 = i\} \\ &= P\{N_j \geq n - 1 | X_0 = i\} \sum_{k=1}^{\infty} P\{T_j = k | X_0 = i\} \\ &= P\{N_j \geq n - 1 | X_0 = i\} h_{ij} = \dots \\ &= P\{N_j \geq 1 | X_0 = i\} h_{ij}^{n-1} = h_{ij}^n. \end{aligned}$$

It follows that

$$\begin{aligned} P\{N_j = n | X_0 = i\} &= P\{N_j \geq n | X_0 = i\} - P\{N_j \geq n + 1 | X_0 = i\} \\ &= h_{ij}^n - h_{ij}^{n+1} = (1 - h_{ij}) h_{ij}^n. \end{aligned}$$

In particular, $P\{N_j = n | X_0 = j\} = (1 - h_{jj}) h_{jj}^n$.

Now if state j is recurrent, then $h_{jj} = 1$, consequently, $P\{N_j = n | X_0 = j\} = 0$ for $n \geq 0$. It follows that $P\{N_j = \infty | X_0 = j\} = 1$. If state j transient, i.e., $h_{jj} < 1$, set $q = 1 - h_{jj}$, then $P\{N_j = n | X_0 = j\} = q(1 - q)^n$. \square

Lemma 15.8. *Let P be an $n \times n$ stochastic matrix whose row sum are 1. Then $\text{rank}(P - I) = n - 1$ if and only if the digraph corresponding to P is strongly connected.*

Proof. Note that $\text{rank}(I - P) \leq n - 1$ since $(I - P)\mathbf{1} = 0$. Suppose $\text{rank}(I - P) < n - 1$. Then there exists a nonzero vector $\mathbf{u} = (u_1, \dots, u_n)$, independent of $\mathbf{1}$, such that $(I - P)\mathbf{u} = 0$. Let u_{i_0} and u_{i_m} be minimal and maximal values of u_1, \dots, u_n respectively. Consider a directed path $i_0 i_1 \dots i_m$ from i_0 to i_m . There exists an i_k such that $u_{i_0} = \dots = u_{i_k} < u_{i_{k+1}}$. Then $p_{i_k i_{k+1}} > 0$. Thus

$$u_{i_0} = u_{i_k} = \sum_{j=1}^n p_{i_k j} u_j > \sum_{j=1}^n p_{i_k j} u_{i_0} = u_{i_0},$$

which is a contradiction. Hence $\text{rank}(I - P) = n - 1$.

Suppose the underlying graph is not strongly connected, i.e., there are at least two strongly connected components. The components can be linearly arranged as C_0, C_1, \dots, C_m such that if there exist states $i \in C_\alpha, j \in C_\beta$ with $p_{ij}^{(t)} > 0$ for some $t \geq 1$ then then C_α is ahead of C_β . So the matrix of P is blocked by the partition C_0, C_1, \dots, C_m into an upper block-triangular matrix. Since DTMC is connected, for each α there exist $i \in C_\alpha, j \in C_\beta$ such that $\alpha < \beta$ and $p_{ij}^{(t)} > 0$ for some $t \geq 1$

, choose one source component C_0 and one sink component C_1 . Then $p_{ji}^{(t)} = 0$ for all $i \in C_0, j \in C_1$ and $t \geq 1$. \square

\square

Final Exam Review

Boolean Algebra definition and examples

How to write a given Boolean function as a Boolean expression

Logic gates and networks

Karnaugh maps for simplifying Boolean expressions

Minimum spanning tree (MST)

Kruskal's algorithm and Prim's algorithm for MST

Distinct weights imply the uniqueness of MST (proof of the fact)

Planar and non-planar graphs

Euler formula and Platonic polyhedra (five type such polyhedron graph)

Relation between the number of vertices and the number of edges

Relation between the number of edges and the number of faces (or regions)

Estimation using the length of cycles for non-planarity

Football graph and the planar graphs with only pentagons and hexagons

Depth first search for spanning tree

Breadth first search for spanning tree

Characterizations of trees

Tree formula and forest formula

Eulerian tour and Eulerian path

Fleury's algorithm for finding an Eulerian tour or Eulerian path.

Hamilton tour and path

Degree-sum condition theorem and its generalization

Characterization of bipartite graphs

Quotient graphs

Permutations and combinations

Combination with repetition

Permutations of a multiset

Binomial and multinomial coefficients and theorems

Inclusion-exclusion principle

Pigeonhole principle

Linear recurrence relation

Characteristic equation

General solutions with distinct roots, and with equal roots

Divide and conquer method

$$x_n = ax_{\frac{n}{2}} + b.$$

Application of recurrence relations

Transitive closure of binary relation, Warshall's algorithm

Equivalence relation

Mathematical induction

Propositional logic

Modular integers

Finding all integer solutions for equation like $ax + by = c$ for integers a, b, c .

Solving equation $ax = b$ modulo n

($x = bu/d + kn/d$ with $k \in \mathbb{Z}$, where $d = au + bv = \gcd(a, b)$).