6 Graph Theory

Basic Definitions of Graphs

- Graph is a commonly used mathematical model for representing how things are connected. In graphs, there are two basic types of elements: **vertices (结点)** and **edges (边)**. Intuitively, a vertex can represent a place and an edge can present the road connecting two places. Depending on whether the edges consider direction or self-loops are allowed, we categorize graphs are follows.

	simple (简单)	multi (多重)	pseudo (图)	directed (有向)	simple directed
direction	×	×	×	√	√
multi-edge	×	√	√	√	×
self-loop	×	×	√	√	×

- As we can see, directed graph is the most general notion as it allows direction, multi-edges and self-loops. It can be defined as follows.

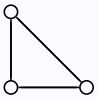
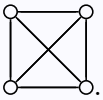
Definition: Graphs

A graph is a 3-tuple $G = \langle V, E, \mathcal{E} \rangle$, where $V = \{v_1, \cdots, v_n\}$ is the set of vertices, $E = \{e_1, \cdots, e_m\}$ is the set of edges and $\mathcal{E} : E \to V \times V$ is a function that determines the starting vertex and the ending vertex of each edge.

The above definition can be simplified if we consider special graphs. For example,

- for pseudo-graph, since there is no direction, we do not need to assign each edge an ordered pair. Instead, we can define $\mathcal{E} : E \to \{\{v, v'\} : v, v' \in V\}$.
- furthermore, for simple-graph, since there is no self-loop, we can define $\mathcal{E} : E \to \{\{v, v'\} : v, v' \in V\} \setminus I_V$. Moreover, since there is no multi-edge, function $\mathcal{E}$ is actually an *injection* because we cannot have two different edges starting from the same vertex and ending at the same vertex.

- For directed graph, $e_k = \langle v, v' \rangle$ means that e_k is an edge from v to v'. For undirected graph, $e_k = \{v, v'\}$ means that e_k is an edge connecting v and v'. For simplicity, we usually just write both cases as $e_k = (v, v')$. Sometimes, we also just write a graph as a 2-tuple $G = \langle V, E \rangle$ and how each edge connects vertices is just shown by the figure.

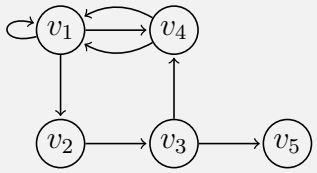
- We say a simple graph is an **empty graph (空图)** if it has no edges and we denote by N_n the empty graph with n vertices. Also we say a simple graph is a **complete graph (全图)** if each pair of vertices are connected and we denote by K_n the complete graph with n vertices. For example, K_3 is  and K_4 is  .

Terminologies of Graphs

- Let $G = \langle V, E\rangle$ be an undirected graph. We say two vertices v, v' are **adjacent (相邻的)** if $e = (u, v)$ for some $e \in E$. Also, we say edge e is **incident (关联)** with v and v'. Given vertex $v \in V$, we denote by $\Gamma(v)$ the set of all its adjacent vertices (or neighbours).

- Furthermore, for directed graphs, if there exists an edge of form $e = \langle v, v'\rangle$, then we say v is the **predecessor (前驱)** of v' and v' is the **successor (后继)** of v. For vertex $v \in V$, we denoted by $\Gamma^-(v)$ and $\Gamma^+(v)$ the set of predecessors of v and the set of successors of v, respectively.

- The **degree (度)** of a vertex v in an undirected graph, denoted by $d(v)$, is the number of edges incident with it, for which a self-loop at v contributes twice to $d(v)$. For directed graphs, we further define $d^+(v) = |\{e \in E : e = \langle v, v'\rangle\}|$ as its **out-degree (出度)** and $d^-(v) = |\{e \in E : e = \langle v', v\rangle\}|$ as its **in-degree (入度)**.



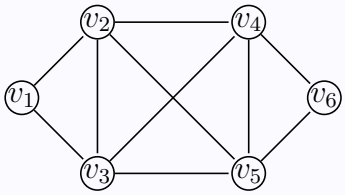
$$d(v_1) = 6, d^+(v_1) = 3, d^-(v_1) = 3$$
$$d(v_2) = 2, d^+(v_2) = 1, d^-(v_2) = 1$$
$$d(v_3) = 3, d^+(v_3) = 2, d^-(v_3) = 1$$
$$d(v_4) = 4, d^+(v_4) = 2, d^-(v_1) = 2$$
$$d(v_5) = 1, d^+(v_5) = 0, d^-(v_5) = 1$$

- Regaridng the degrees of vertices, we have the following basic results:
  ① For any graph $G = \langle V, E\rangle$, where $|V| = n, |E| = m$, we have $\boxed{\sum_{v\in V} d(v) = 2m}$.
  ② For any directed graph $G = \langle V, E\rangle$, we further have $\sum_{v\in V} d^+(v) = \sum_{v\in V} d^-(v) = m$.
  ③ The number of edges in K_n is $\frac{1}{2}n(n-1)$
  ④ Any non-empty simple graph must have two vertices with the same degree.

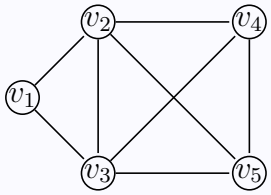
- $G' = \langle V', E'\rangle$ is said to be a **sub-graph (子图)** of $G = \langle V, E\rangle$, denoted as $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. Given graph $G = \langle V, E\rangle$ and a set of vertices $V' \subseteq V$, the **sub-graph induced (诱导子图)** by V' is a new graph $G' = \langle V', E'\rangle$ such that

$$(\forall v, v' \in V')(e = (v, v') \in E \to e \in E')$$

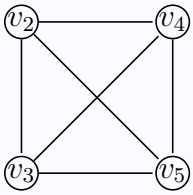
For example, for the following graph G, G_1 is the sub-graph induced by $\{v_1, v_2, v_3, v_4, v_5\}$ and G_2 is the sub-graph induced by $\{v_2, v_3, v_4, v_5\}$. Note that G_3 is a just sub-graph of G but is not an induced sub-graph because edges (v_2, v_5) and (v_3, v_4) are missing.



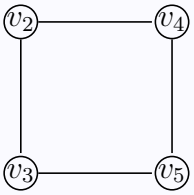
G



$G_1 \subseteq G$



$G_2 \subseteq G$



$G_3 \subseteq G$

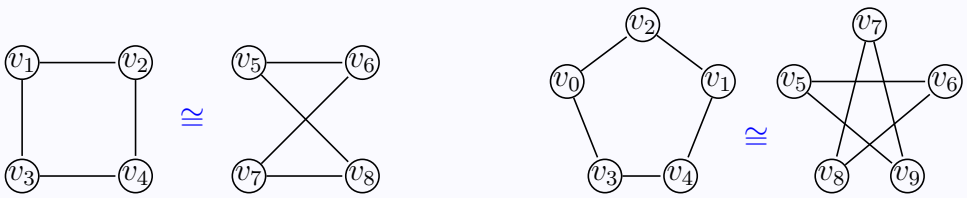
Graph Isomorphism

- Note that two graphs may looks different but essentially they are the "same" in terms of renaming of vertices and edges.

Definition: Graph Isomorphism

Let $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ be two graphs. We say G_1 and G_2 are **isomorphic (同构)**, denoted by $G_1 \cong G_2$, if there exists a bijection $f : V_1 \to V_2$ such that $(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2$.

For example, the following pairs of graphs are isomorphic:



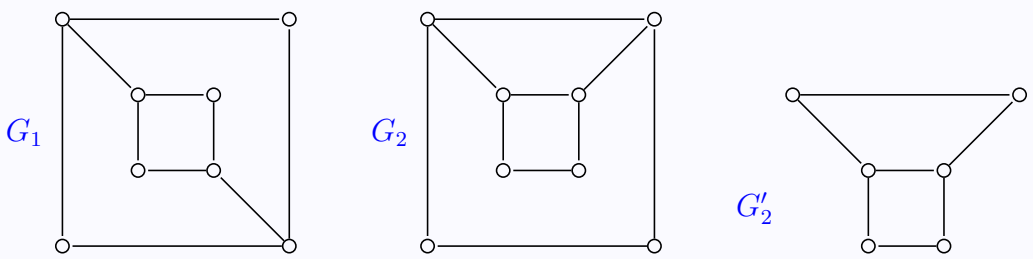
- In general, it is difficult to check whether or not two graphs are isomorphic. However, we can find *necessary conditions* two graphs must hold if they are isomorphic.

Theorem: Necessary Conditions for Graph Isomorphism

For any $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ such that $G_1 \cong G_2$, then

1. $|V_1| = |V_2|$ and $|E_1| = |E_2|$.
2. The non-increasing sequences of vertices degrees of G_1 and G_2 are the same.
3. For any induced graph of G_1, there is an induced sub-graph of G_2 such that these two induced sub-graphs are isomorphic.

- Note that, the above conditions are just necessary. If one of them is not satisfied, then we can conclude that they are not isomorphic. However, we cannot make any conclusion if they are satisfied. For example, let us consider graphs G_1 and G_2. For both graphs, their non-increasing sequences of vertices degrees are both $\{3, 3, 3, 3, 2, 2, 2, 2\}$. However, they are not isomorphic. To see this, let us consider induced sub-graph $G_2' \subseteq G_2$. One can check that we cannot find an induced sub-graph $G_1' \subseteq G_1$ such G_1' and G_2' have the same non-increasing sequences of vertices degrees. Therefore, we conclude that $G_1 \ncong G_2$.



Paths and Connectivity

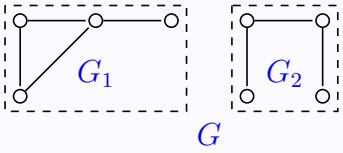
- Given a graph $G = \langle V, E \rangle$, a **path (道路)** is a sequence $v_0 e_1 v_1 \cdots e_n v_n$ such that $e_i = (v_{i-1}, v_i)$. The length of a path is the number of edges in it, e.g., the length of the above path is n. A single isolated vertex v is also considered as a path with length zero. Note that, in general, path considers directions. We say a path $v_0 e_1 v_1 \cdots e_n v_n$ is
  - a **simple path (简单道路)** if it has no repeated edges;
  - a **elementary (初级道路)** if it has no repeated vertices;
  - a **circuit (回路)** if $v_0 = v_n$.
  - We call circuit $v_0 e_1 v_1 \cdots e_n v_n$ a simple circuit (简单回路) if it has no repeated edges and an elementary circuit (初级回路) if it has no repeated vertices except v_0 and v_n.

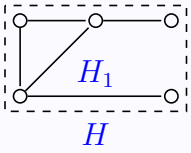
- Given an undirected graph $G = \langle V, E \rangle$, we say G is **connected (连通的)** if there exists a path between each pair of vertices; otherwise, we say G is disconnected. In general, a graph can be decomposed into a set of sub-graphs that are connected.

Definition: Connected Components

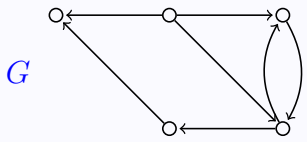
Given an undirected graph $G = \langle V, E \rangle$, we say sub-graph $G' \subseteq G$ is a **connected component (连通分支)** of G if (i) it is connected; and (ii) for any other connected $G'' \subseteq G$, we have $G' \subseteq G''$ implied $G' = G''$.

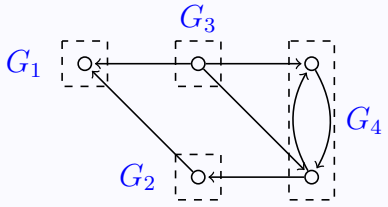
Intuitively, a connected component is a *maximal* connected sub-graph of G. For example, for graph G, it has two connected components G_1 and G_2. However, for graph H, since it is already connected, it only has one connected component H_1, which is itself.





- Similarly, for a directed graph $G = \langle V, E \rangle$, we say G is **strongly connected (强连通的)** if there exists a path between each ordered pair of vertices by considering directions. For directed graph, we can further talk about **strongly connected component (强连通分支)** (SCC). The definition of SCC is exactly the same as the connected component. The only difference is that connectivity in SCC considers directions. For example, for the following directed graph G, it has four SCCs G_1, G_2, G_3 and G_4.





Euler Paths and Circuits

- In some problems, we want to find a path that passes through *all edges*, e.g., to clean up snows in all streets. Furthermore, for the purpose of efficiency, we want that no edge is repeated in the path. Such a path is called an Euler path.

Definition: Euler Paths and Euler Circuits

Let $G = \langle V, E \rangle$ be a pseudo-graph. Then

- an **Euler circuit (欧拉回路)** is a simple circuit containing all edges.
- an **Euler path (欧拉道路)** is a simple path containing all edges.

- An interesting question is how can we determine whether or not a given graph contains an Euler circuit. In fact, the existence of an Euler circuit can be checked easily by the following result.

Theorem: Necessary and Sufficient Condition for Euler Circuit

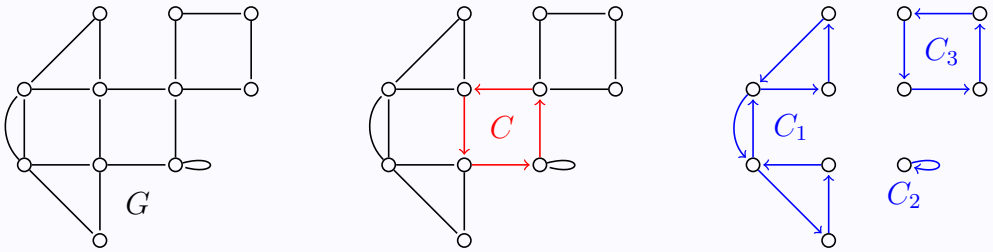
A connected pseudo-graph $G = \langle V, E \rangle$ has an Euler circuit *if and only if*

$$\forall v \in V : d(v) \text{ is even}$$

Proof: ($\Rightarrow$) Let P be an Euler path. Each time P visits v, it contributes two degrees. Since P contains exactly all edges, $d(v)$ must be even.

($\Leftarrow$) We prove this direction by induction on the number of edges $|E| = m$.

- **Induction Basis:** When $m = 1$, G has to be  . Here, we have $d(v) = 2$ and there exists an Euler circuit.
- **Induction Step:** Now, let us assume that for any graph having $m \leq k$ edges, if all vertices have even degrees, then there exists an Euler circuit. We want to show that the implication holds for graphs with $m = k + 1$ edges. The arguments are as follows.
  1. Since $d(v)$ is even for any $v \in V$, there exists a simple circuit C in G.
  2. By removing all edges in C from G, we obtain p connected components $G_1, \cdots, G_p$.
  3. For each $G_i = \langle V_i, E_i \rangle$, we have $|E_i| \leq k$. Furthermore, the degree of each vertex in G_i is still even. Therefore, there exists an Euler path C_i for each G_i.
  4. By connecting $C_1, \cdots, C_n$ together by C, we obtain an Euler path for G.



Euler Paths and Circuits

- Our necessary and sufficient condition for Euler circuits can be extended easily to the case of Euler paths. In an Euler path, we still require that whenever each vertex has an in-degree, it should have an out-degree, except the starting and the ending vertices. Therefore, we have the following result.

Theorem: Necessary and Sufficient Condition for Euler Path

A connected pseudo-graph $G = \langle V, E\rangle$ has an Euler path if and only if

$$|\{v \in V : d(v) \text{ is odd}\}| \leq 2$$

- The proof of the above result is rather straightforward. Specifically, when there is no odd-degree-vertex, there exists an Euler circuit. When there are two different odd-degree vertices, by connecting these two vertices, we get a graph in which the degrees of all vertices are even. Then we can find an Euler circuit in the new graph. By removing the newly-added edge from this Euler circuit, we get an Euler path in the original graph.

- The above results are developed for undirected graphs. In fact, we can extend the conditions to directed graphs as follows.

Theorem: Conditions for Euler Paths and Circuits in Directed Graphs

① A strongly connected directed-graph has an Euler circuit if and only if

$$\forall v \in V : d^+(v) = d^-(v)$$

② A strongly connected directed-graph has an Euler path if and only if at most one vertex v has $d^+(v) - d^-(v) = 1$, at most one vertex has $d^-(v) - d^+(v) = 1$ and every other vertex has equal in-degree and out-degree.

- Let $G = \langle V, E\rangle$ be a simple graph. An edge $e \in E$ is said to be a **bridge/cut-edge (桥/割边)** if $G - e$ has more connected components than G, where $G - e$ is the sub-graph obatined by removing edge e from G. Then Euler circuits can be constructed by the following algorithm. We use notation $E(v)$ to denote the set of all edges incident with v.

Fleury's Algorithm

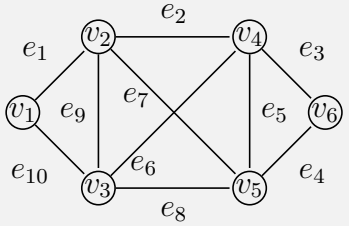
Step 1: Choose an arbitrary $v_0 \in V$ and $P_0 \leftarrow v_0$

Step 2: Suppose that $P_k = v_0 e_1 v_1 \cdots e_k v_k$.
Choose $e_{k+1} \in E(v_k) \setminus \{e_1, \ldots e_k\}$ such that

$$e_{k+1} \text{ is not a bridge in } G - \{e_1, \ldots, e_k\}$$

If there is no above choice, then pick $e_{k+1} \in E(v_k) \setminus \{e_1, \ldots e_k\}$. Add e_{k+1} to P_k to obtain P_{k+1}.

Step 3: Repeat the above until $k = |E|$.



A possible Euler circuit:
$C = e_1 e_7 e_4 e_3 e_2 e_9 e_6 e_5 e_8 e_{10}$

Hamilton Paths and Circuits

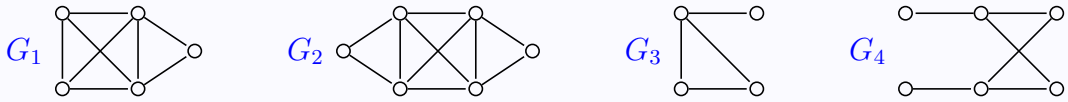
▶ In some problems, we want to find a path that passes through *all vertices*, e.g., to search all buildings in a campus. Similarly, for the purpose of efficiency, we want that no vertex is repeated in the path. Such a path is called a Hamilton path.

Definition: Hamilton Paths and Hamilton Circuits

Let $G = \langle V, E \rangle$ be a simple-graph. Then

- a **Hamilton circuit (哈密顿回路)** is an elementary circuit containing all vertices.
- a **Hamiltonr path (哈密顿道路)** is an elementary path containing all vertices.

▶ Note that, for Hamilton path/circuit, it suffices to consider simple graphs because self-loops or multi-edges do not matter in visiting vertices. Also, it is without loss of generality to consider simple graphs with more than 2 vertices. For example, for G_1 and G_2 shown below, we can find Hamilton circuits. For G_3, we can only find a Hamilton path. However, for G_4, there exists even no Hamliton path.



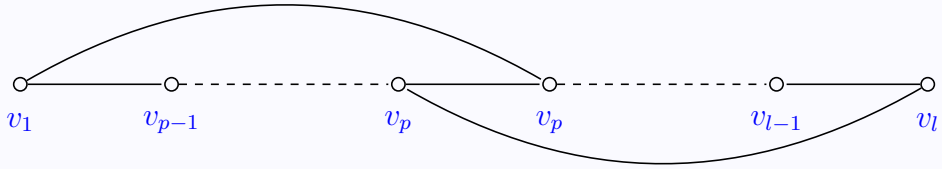
Unfortunately, unlike Euler circuit/path, for which we can find simple condition for the existence, the problem of deciding the existence of Hamilton circuit/path is NP-hard. Therefore, we seek to find simple but only sufficient condition.

▶ We say an elementary path $v_1 v_2 \cdots v_l$ is **maximal (极长)** if v_1 and v_l are not adjacent to any vertex that is not on the path. In other words, we cannot further extend the path either from v_1 or from v_l. Then we have the following result.

Lemma: Property of Maximal Elementary Paths

Let $P = v_1 v_2 \cdots v_l$ be a *maximal elementary path*. If $d(v_1) + d(v_l) \geq l$, then there must exist an *elementary circuit* that passes through $v_1, v_2, \cdots, v_l$.

Proof: By contraposition. Suppose that there is no elementary circuit that passes through $v_1, v_2, \cdots, v_l$. Then we know that, for any $p \in \{1, \ldots, l\}$, we have $(v_1, v_p) \in E \Rightarrow (v_l, v_{p-1}) \notin E$. Otherwise, it will form an elementary circuit containing $v_1, v_2, \cdots, v_l$. The argument can be seen from the following figure.



Now suppose that $d(v_1) = k$. Then we have $d(v_l) \leq l - k - 1$ because (i) v_l is not adjacent to any vertex $v_1, \ldots, v_{l-1}$; and (ii) v_l is not adjacent to any v_{p-1} such that $(v_1, v_p) \in E$. This means that $d(v_1) + d(v_l) \leq l - 1$.

Hamilton Paths and Circuits

▶ Here, we provide a sufficient condition for the existence of Hamilton path. The idea is based on the previous lemma, which shows that if the connectivity of the graph is greater than some certain number, then we can always change maximal path to a circuit containing the same vertices.

Theorem: Sufficient Condition for Hamilton Path

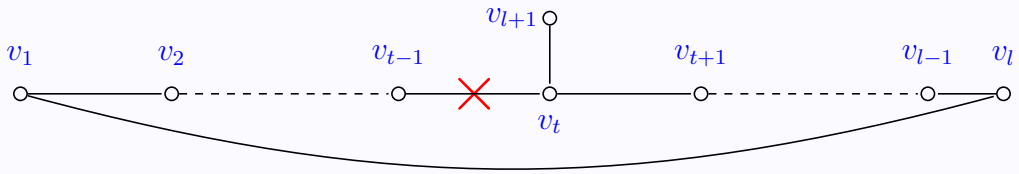
A simple-graph $G = \langle V, E\rangle$ has a Hamilton path **if**

$$\forall v, v' \in V : d(v) + d(v') \geq n - 1$$

Proof: First, we show by contradiction that G is connected. Assume that G is not connected. Then it has at least two connected components, say G_1 and G_2 and suppose that G_1 has n_1 vertices and G_2 has n_2 vertices. Let us pick two vertices $v_1 \in V_{G_1}$ and $v_2 \in V_{G_2}$. Then we have $d(v_1) \leq n_1 - 1, d(v_2) \leq n_2 - 1$, which means that $d(v_1) + d(v_2) \leq n - 2 < n - 1$. This contradicts to our condition, which means that G is connected. Next, we show the existence of Hamilton path by construction.

First, we pick an arbitrary maximal elementary path $P = v_1 v_2 \cdots v_l$, where $l \leq n$. Such a path can be obtained by extending the starting /ending vertex until we cannot do so.

- if $l = n$, then P is a Hamilton path.
- if $l < n$, then $d(v_1) + d(v_l) \geq n - 1 \geq l$. Then the previous lemma applies. That is, there exists an elementary circuit, say C, passing through $v_1, \ldots, v_l$. Since G is connected, there exists a new vertex v_{l+1} out of C and is adjacent to some vertex v_t in C. Then we can obtain a new longer elementary path P' containing $v_1, \cdots, v_l, v_{l+1}$ by first connecting v_{l+1} to C and then breaking C. The procedure is shown as follows



By repeating the above procedure, we can extend an elementary path longer and longer until all vertices have been included.

▶ In the above proof, we can use elementary path $P = v_1 v_2 \cdots v_l$ to find an elementary circuit only when $l < n$; otherwise condition $d(v_1) + d(v_l) \geq n - 1 \geq l$ does not hold. If we further have $d(v_1) + d(v_l) \geq n$, then we can actually extend the Hamilton path found to a Hamilton circuit by using the lemma one more time. This leads to the following result.

Theorem: Sufficient Condition for Hamilton Circuit

A simple-graph $G = \langle V, E\rangle$ has a Hamilton circuit **if**

$$\forall v, v' \in V : d(v) + d(v') \geq n$$

Trees

- In graph theory, a tree is an undirected graph in which any two vertices are connected by *exactly one* path, or equivalently a connected acyclic undirected graph.

Definition: Tree

A (undirected) **tree (树)** is a connected undirected graph with no circuit.

Because no circuit is allowed in a tree, we have the following results

- a tree is a simple graph; otherwise, self-loops or multi-edges will make circuit.
- each edge in a tree is a bridge/cut-edge; otherwise, a circuit can be found.

In trees, we call vertices with degree one **leaves (树叶)**. Disconnected graphs with no circuit are called **forest (森林)**.

- In fact, we have many equivalent definitions of tree as follows:

Theorem: Equivalent Definitions of Trees

Let $T = \langle V, E\rangle$ be an undirected graph. Then the following statements are equivalent definitions for trees.

① T is a connected and $|E| = |V| - 1$;
② T has no circuit and $|E| = |V| - 1$;
③ T is connected and each edge is a bridge;
④ T has no circuit and by adding any edge, we will have a circuit;
⑤ the path between any two vertices is unique.

Let us prove ① by induction on the number of vertices.

- *Induction Basic:* when $|V| = 1$, we know that T has no circuit iff $|E| = 0 = |V| - 1$.
- *Induction Step:* We assume that T is a tree iff ① holds for $|V| \leq k$ and we consider the case of $|V| = k + 1$. We remove an arbitrary vertex $v \in V$ from T. Let $v_1, \cdots, v_k \in V$ be the adjacent vertices of v. Let $C_1, \cdots, C_k$ be the connected components that contain $v_1, \cdots, v_k$, respectively, in the removed graph. Then we have

$$\begin{aligned} T \text{ is a tree iff } & C_1, \cdots, C_k \text{ are distinct and each } C_i \text{ is a tree;} \\ \text{iff } & C_1, \cdots, C_k \text{ are distinct and } |E_i| = |V_i| - 1 \\ \text{iff } & |\bigcup E_i| = |\bigcup V_i| - k \\ \text{iff } & |E| = |V| - 1 \end{aligned}$$

- We can also show the following properties by contradiction using the handshake theorem.

Theorem:

① For tree T with $|V| \geq 2$, there are at least two leaves,
② For forest F with $|V| = n$ and k connected components, we have $|E| = n - k$.

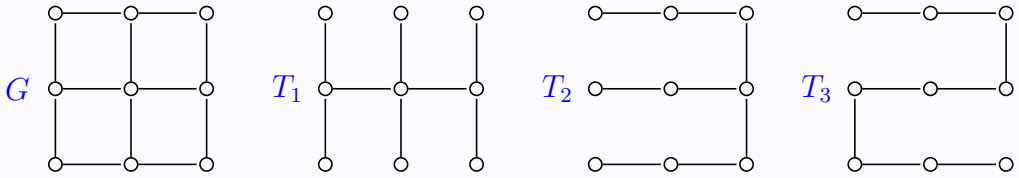
Spanning Trees

- In some applications, for example building electrical lines, we need the tree to connect all vertices. Such a tree is called a spanning tree.

Definition: Spanning Trees

A sub-graph of G is called a **spanning tree (生成树)** of G if (i) it is a tree; and (ii) it contains all vertices of G.

We note that a simple graph G has a spanning tree if and only if it is connected because we can always remove edges until it is a spanning tree. Also, the spanning tree may not be unique. For example, for G, sub-graphs T_1, T_2 and T_3 are all spanning trees.

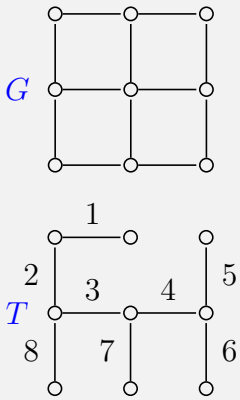


- Given a connected graph, we can construct a spanning tree using the Depth-First Search (深度优先搜索). The basic idea is to (i) keep adding new edges until no edge can be added; (ii) track back to the vertice at which some edge can be added, and repeat.

Depth-First Search Algorithm

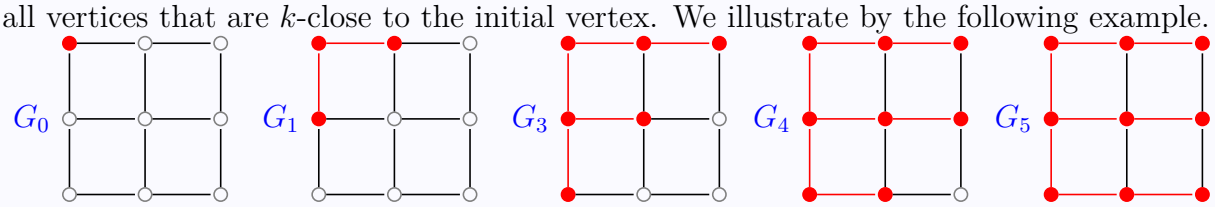
Input: connected graph $G = \langle V, E \rangle$
Output: spanning tree $T = \langle V_T, E_T \rangle$
Pick $v_0 \in V, V_T \leftarrow \{v_0\}, E_T \leftarrow \phi$;
`DFS` (v_0, E_T, V_T, G);

procedure `DFS` (v, E_T, V_T, G);
  for $v' \in \Gamma(v)$ **do**
    if $v' \notin V_T$ **then**
      $V_T \leftarrow V_T \cup \{v'\}, E_T \leftarrow E_T \cup \{(v', v)\}$;
      `DFS` (v', E_T, V_T, G);



The number of each edge denote when it is added.

- An alternative approach for constructing a spanning tree using the Breadth-First Search (广度优先搜索). The basic idea is to start from a vertex and at the kth iteration, add all vertices that are k-close to the initial vertex. We illustrate by the following example.



Minimum Spanning Trees

- In practice, edges in a graph may correspond to roads or lines in physical systems. Therefore, it makes sense to consider the *weight* of a graph. Formally, a **weighted graph (加权图)** is a tuple $\langle G, w\rangle$, where $G = \langle V, E\rangle$ is a graph and $w : E \to \mathbb{R}$ is a weight function. The weight of a graph, denoted by $w(G)$, is the sum of the weights of all vertices, i.e., $w(G) = \sum_{e\in E} w(e)$. It is of interest to consider spanning tree with minimum weight.

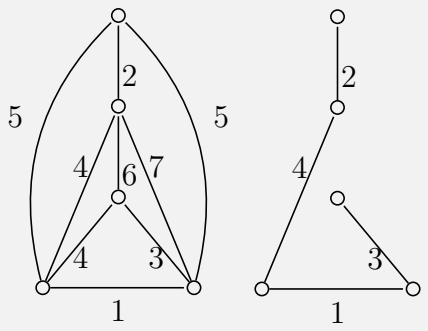
Definition: Minimum Spanning Tree

Given a weighted graph $\langle G, w\rangle$, a **minimum spanning tree (最小生成树)** T is spanning tree such that, for any spanning tree T', we have $w(T) \le w(T')$.

- Hereafter, we will present a set of different algorithms for finding a minimum spanning tree (MST). The first one is the Kruskal's algorithm. Its basic idea is to add edges with minimum weights while avoiding circuits.

Kruskal's Algorithm

Input: connected graph $\langle G, w\rangle$
Output: $T = \langle V, E_T\rangle$
while $E \neq \emptyset$ *and* $|E_T| < n - 1$
do
  Pick minimum $e \in E$;
  $E \leftarrow E \setminus \{e\}$;
  if $T + e$ *has no circuit* **then**
    $E_T \leftarrow E_T \cup \{e\}$



G_1 MST T

Proof: The correctness of the Kruskal's algorithm follows from the following arguments.

First, the resulting graph is a tree because it is connected and has no circuit. Moreover, it is a spanning tree because $V_T = V$. Next, we show that $w(T)$ is minimum. To this end, let T^* be an MST such that $T^* \neq T$. We consider an edge $e \in E_{T^*} \setminus E_T$. By adding e to T, $T + e$ has a circuit C such that:

$$\text{(i)}(\forall a \in C)(\omega(a) \le \omega(e)); \text{ and (ii) there exist an edge } f \text{ in } C \text{ but not in } T^*$$

Then we consider a new tree $T_2 = T - f + e$. We note that T_2 is still a spanning tree and

$$w(T_2) = w(T) - w(f) + w(e) \ge w(T)$$

Furthermore, T_2 has more common edge with T^* than T has. By repeating the above procedure, we obtain $T_3, T_4, \cdots, T^*$ such that

$$w(T) \le w(T_2) \le w(T_3) \le w(T_4) \le \cdots \le w(T^*)$$

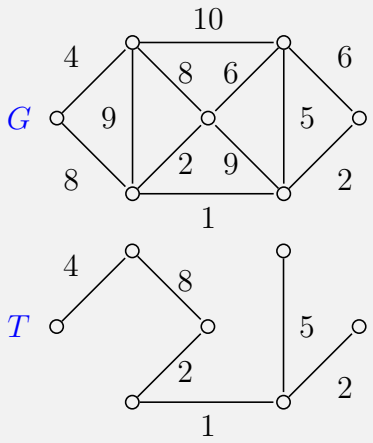
However, T^* is already minimum, which means $w(T) = w(T^*)$ and T is also minimum.

Minimum Spanning Trees

▶ Here, we present another algorithm for finding MST called the Prim's Algorithm. The idea is to start from a selected vertex and at each instant, grow the tree to the unexplored region with minimum weight.

Prim's Algorithm

Input: connected graph $\langle G, \omega \rangle$
Output: $T = \langle V_T, E_T \rangle$
pick $v \in V$, $V_T \leftarrow \{v\}, E_T \leftarrow \emptyset$;
while $V_T \neq V$ **do**
  Find minimum $e = (v_1, v_2) \in E$ such that $v_1 \in V_T$ and $v_2 \notin V_T$;
  $V_T \leftarrow V_T \cup \{v_2\}$;
  $E_T \leftarrow E_T \cup \{e\}$;



Proof: The correctness of the Prim's algorithm follows from the following arguments.

Let V_k and E_k be the set of vertices and edges after k-th iteration, respectively, where $k = 0, 1, \cdots, n-1$ and initially we have $V_0 = \{v\}, E_0 = \emptyset$. Let e_k be the edge added to E_T in the k-th iteration, i.e., $E_k = \{e_1, e_2, \cdots, e_k\}$.

Claim: For each $k \leq n-1$, there exists an MST $T^* = \langle V, E^* \rangle$ such that $E_k \subseteq E^*$.

We prove the above claim by induction:

- *Induction Basis:* For $k = 0$, we have $E_0 = \emptyset$, which is trivially correct
- *Induction Step:* We assume that for E_k, there exists an MST $T^* = \langle V, E^* \rangle$ such that $E_k \subseteq E^*$. Now we consider the following cases of E_{k+1}:
  - if $e_{k+1} \in E^*$, then $E_{k+1} = \{e_1, \cdots, e_k, e_{k+1}\} \subseteq E^*$;
  - if $e_{k+1} \notin E^*$, then suppose $e_{k+1} = (u, v), u \in V_k$ and $v \notin V_k$. Let P be the unique path in T^* between u and v. Then there must exist an edge $e' = (u', v')$ in P such that $u' \in V_k$ and $v' \notin V_k$. Using this edge, we can construct a new tree by

$$T' = T^* \cup \{e_{k+1}\} - \{e'\}$$

Clearly, T' is a spanning tree because it has exactly $n-1$ edges and is connected. Furthermore, T' is MST since

$$\omega(T') = \omega(T^*) + \omega(e_{k+1}) - \omega(e') \leq \omega(T^*).$$

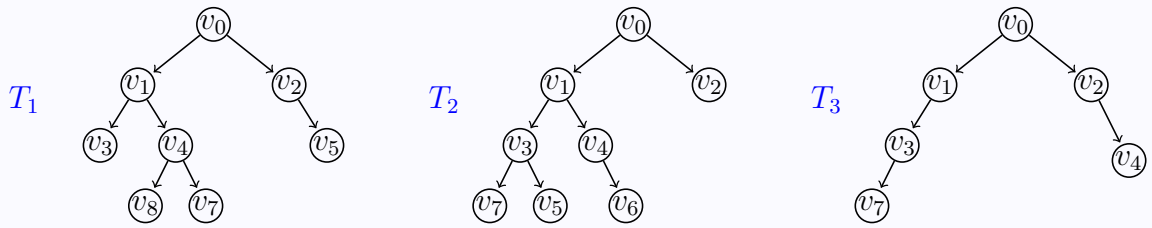
Therefore, we know that each E_k is contained in an MST, which means that the resulting graph at final step $k = n-1$ is itself an MST.

Rooted Trees

- Previously, we consider trees with no direction. Here, we further investigate directed tree (有向树). In directed trees, if there exists an edge $e = (u, v)$ from u to v, then we say u is the parent (父结点) of v and v is the child (子结点) of u. Vertices with the same parent are called siblings (兄弟). Therefore, a directed tree will start from a **root (根结点)**, which has no parent and will end up with **leaves (叶结点)**, which have no child. A rooted tree is called a **binary tree (二叉树)** if every vertex has no more than 2 children.

- Given a rooted tree, the **level (层)** of vertex $v \in V$ is the length of the path from the root to v. The **height (高度)** of a tree is the maximum level of all its vertices. A rooted tree with height h is said to be **balanced (平衡的)** if for any vertex, the difference between the heights of the left and right sub-trees are within 1. One can conclude that for any balanced tree with height h, the level of each vertex is either h or $h-1$.

  For example, T_1 is a balanced tree. However, T_2 and T_3 are not balanced trees. For example, for leaf v_2 in T_2, we have $l(v_2) = 1$ but the height of T_2 is 3. For T_3, the height of the left sub-tree of v_1 is one, but the height of the right sub-tree of v_1 is -1.



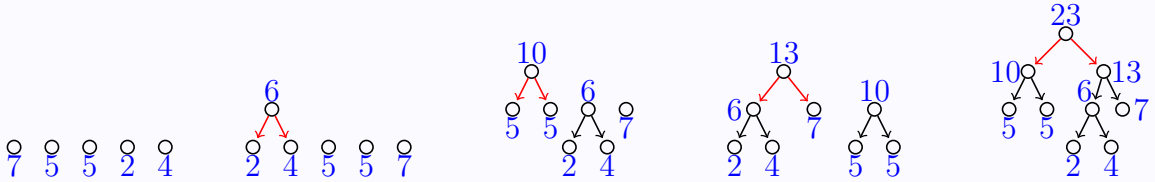
- In some problems, we want to put our information on leaves, e.g., coding letters. It will be more convenient to put those information used more frequently or with more cost at leaves with smaller levels. Specifically, suppose that we have n leaves $\{v_1, \ldots, v_n\}$ and each leaf v_i is associated with a weight w_i. We want to build a binary tree T such that
  (i) the leaves of tree T are exactly $\{v_1, \ldots, v_n\}$; and
  (ii) the weighted path length (WPL) $\sum_{i=1,\ldots,n} l_i w_i$ is minimized, where l_i denotes the level of leaf v_i in T.

  Such a binary tree with minimum WPL is called the **Huffman Tree (哈夫曼树)**, which can be constructed by the following Huffman's algorithm.

  Step 1: Ordering the weights of leaves such that $w_{i_1} \le w_{i_2} \le \cdots \le w_{i_n}$.

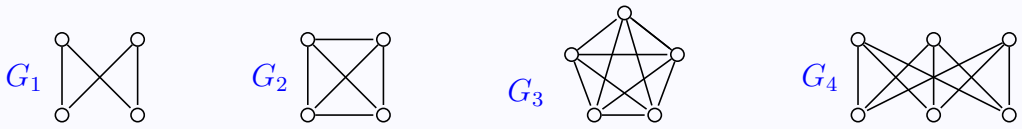
  Step 2: Construct a tree rooted at new vertex v_i whose left child is v_{i_1} and right child is v_{i_2}. Define $w_i = w_{i_1} + w_{i_2}$ as the weight of v_i.

  Step 3: If $n = 1$, then the tree rooted at v_i is the Huffman tree. Otherwise, we delete w_{i_1} and w_{i_2} from the weight list and add w_i. Then go to Step 2 with $n \leftarrow n - 1$.

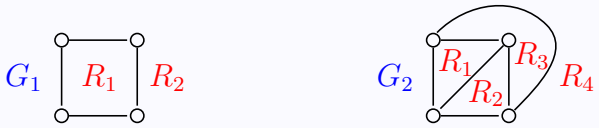


Planar Graphs

- An undiredted multi-graph is called a **planar graph (平面图)** if it can be drawn in the plane without any edges crossing (called planar representation). For example, G_1 and G_2 are planar graphs, but G_3 and G_4 are not planar graphs.



- The planar representation of a graph splits the plane into **regions (区域)** (sometimes also called faces). For example, a tree always splits the plane into a single region. Graph G_1 splits the plane into two regions and graph G_2 splits the plane into four regions.



In the above examples, G_1 has four vertices, four edges and two regions; G_2 has four vertices, six edges and four regions. The relationship among vertices, edges and regions is characterized by the well-known Euler's Formula.

Theorem: Euler's Formula

Let $G = \langle V, E \rangle$ be a connected planar graph with regions R. Then we have

$$|R| = |E| - |V| + 2$$

Proof: First, we consider the case where G has no circuit. Then G is a tree, which means that $|E| = |V| - 1$. Also, it only has one region. Therefore, we have

$$|R| = 1 = (|V| - 1) - |V| + 2.$$

Next, we consider the case where G has at least one circuit. We prove this case by induction on the number of edges:

- *Induction Basis:* When G has one edges, it is a self-loop and we have $2 = 1 - 1 + 2$.
- *Induction Step:* Suppose Euler's formula holds for connected planar graphs with k edges and at least one circuit. We consider the case of $k + 1$ edges.

  We remove an arbitrary edge from a circuit of G and obtain a new graph G', which has k edges. Furthermore, we observe that

  * If G' has no circuit, it is a tree. By the first case, we have $|R_{G'}| = k - |V| + 2$.
  * If G' has circuits, by the induction hypothesis, we still have $|R_{G'}| = k - |V| + 2$.

  Note that G has $|R_{G'}| + 1$ regions because one edge is removed from a circuit. Therefore, we conclude that

$$|R| = |R_{G'}| + 1 = k + 1 - |V| + 2.$$