Outline

I. Notion of a graph

II. Special types of graphs

III. Connectivity

IV. Planar graphs

V. Trees

I. Notion of a graph

1. A (undirected) **graph** $G = (V, E)$ consists of V, a nonempty set of **vertices** (or *nodes*) and E, a set of unordered pairs of elements of V called **edges**.

   Each edge has either one or two vertices associated with it, called its endpoints. The edge $e = \{u, v\}$ is called **incident** with the vertices u and v.

   Two vertices u and v in G are called **adjacent** if $\{u, v\}$ is an edge of G.

   The **degree** $\deg(v)$ of a vertex v is the number of edges incident with it. (note: a loop at a vertex contributes twice to the degree of that vertex.)

2. A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called **simple graph**. Graphs that may have multiple edges connecting the same vertices are called **multigraphs**.

3. **The handshaking theorem.** Let $G = (V, E)$ be a graph, then

   $$2 \cdot n(E) = \sum_{v \in V} \deg(v).$$

   Note that this applies even if multiple edges and loops are present.

   Question: How many edges are there in a graph with 10 vertices and each of degree 6?

4. Graph representations

   - By a picture
   - By an **adjacency list**.
   - Using an **adjacency matrix**: Suppose that a simple graph $G = (V, E)$ with n vertices $\{v_1, v_2, \ldots, v_n\}$. The adjacency matrix $A = (a_{ij})$ is the $n \times n$ zero-one matrix,

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

     Note that A is symmetric.

- Using an **incidence matrix**: Suppose that a simple graph $G = (V, E)$ with n vertices $\{v_1, v_2, \ldots, v_n\}$ and m edges $e_1, e_2, \ldots, e_m$. The incidence matrix is an $n \times m$ matrix $B = (b_{ij})$, where

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

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

6. The **union** of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted as $G_1 \cup G_2$, is the simple graph with vertex set $V_1 \cup V_2$ and edges $E_1 \cup E_2$.

7. Graph isomorphism

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

     In other words, when two simple graphs are isomorphic, there is one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.
   - **Theorem:** Simple graphs G_1 and G_2 are isomorphic *if and only if* for some orderings of their vertices, their adjacency matrices are equal.
   - There is no efficient algorithm known to decide whether two simple graphs are isomorphic. There are $n!$ possible one-to-one correspondences. Most computer scientists believe that no such algorithm exists.
   - To show that two simple graphs are not isomorphic, we can show that they do not share an invariant property that isomorphic simple graphs must both have, such as the same number of vertices, edges, and degrees.

II. Special types of graphs

1. **Complete graph** K_n on n vertices: a simple graph that contains exactly one edge between each pair of distinct vertices.

   Examples: K_1, K_2, K_3, K_4, K_5

2. **Cycle** C_n with $n \geq 3$: consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, ..., $\{v_{n-1}, v_n\}$, $\{v_n, v_1\}$.

   Examples: C_3, C_4, C_5

3. **Wheel** W_n with $n \geq 3$: add an additional vertex to the cycle C_n and connect this new vertex to each of the n vertices in C_n by new edges.

   Examples: W_3, W_4, W_5

4. n-**cube** Q_n: has vertices representing the 2^n bit strings of length n. Two vertices are adjacent if and only if the bit strings that they represent differ one bit position.

   Examples: Q_1, Q_2, Q_3

5. **Bipartite graph:** a simple graph G is called *bipartite* if its vertex set V can be partitioned into two disjoint nonempty sets V_1 and V_2 such that every edge connects a vertex in V_1 and a vertex in V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2).

   Theorem: A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex of the graph so that no two adjacent vertices are assigned the same color.

6. **Complete bipartite graph** $K_{m,n}$: each vertex of V_1 of m vertices is connected to each vertex of V_2 of n vertices.

   Examples: $K_{2,3}$, $K_{3,3}$.

III. Connectivity

1. A **path** *of length* n from v_0 to v_n in a graph $G = (V, E)$, where $V = \{v_i\}$ and $E = \{\, e_i = \{v_i, v_j\}\,\}$, is an alternating sequence of $n+1$ vertices and n edges beginning with v_0 and ending with v_n:

   $$(v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n),$$

   where the edge e_i is incident with v_{i-1} and v_i. When the graph is simple, we denote this path by its vertex sequence $v_0, v_1, v_2, \ldots, v_{n-1}, v_n$.

2. The path is a **cycle** (or *circuit*) if it begins and ends at the same vertex. A path or cycle is *simple* if it does not contain the same edge more than once.

3. A graph is called **connected** if there is a path between every pair of distinct vertices of the graph.

4. Counting paths between vertices

   Theorem: let G be a graph with adjacency matrix A with respect to the ordering $v_1, v_2, \ldots, v_n$. The number of different paths of length k from v_i to v_j equals to the (i, j) entry of A^k.

   Proof: The theorem can be proven using mathematical induction. Let G be a graph with the adjacency matrix A. Basis step: the number of paths from v_i and v_j of length 1 is the (i, j)th entry of A, because this entry is the number of edges from v_i to v_j.

   Inductive step: Assume that the (i, j)th entry of A^k is the number of different paths of length k from v_i to v_j. (This is the induction hypothesis.)

   Because $A^{k+1} = A^k A$, the (i, j)th entry of A^{k+1} is $b_{i1}a_{1j} + b_{i2}a_{2j} + \cdots + b_{in}a_{n,j}$, where b_{ik} is the (i, k)th entry of A^k. By the induction hypothesis, b_{ik} is the number of paths of length k from v_i to v_k. Next, we know that a path of length $k+1$ from v_i to v_j is made up of a path of length k from v_i to some intermediate vertex v_k, plus an edge from v_k to v_j. By the product rule for counting, the number of such paths is the product of the number of paths of the length k from v_i to v_k, namely b_{ik}, and the number of edges from v_k to v_j, namely a_{kj}. When these products are added for all possible intermediate vertices, the desired results follows by the sum rule for counting.

   By the principle of mathematical induction, the theorem is proven. □

5. An **Eulerian path (cycle)** in G is a path (cycle) containing every **edge** of G exactly once.

   Theorem. A connected graph has an Euler cycle *if and only if* each of its vertices has even degree.

Proof. "$\Rightarrow$": Suppose G has an Eulerian cycle T. For any vertex v of G, T enters and leaves v the same number of times without repeating any edge. Hence v has even degree.

"$\Leftarrow$: We use the proof by construction. Suppose that each vertex of G has even degree. Let us construct an Eulerian cycle. We begin a path T_1 at any edge e. We extend T_1 by adding one edge after the other. If T_1 is not closed at any step, say T_1 begins at u and ends at $v \neq u$, then only an odd number of the edges incident on v appear in T_1. Hence we can extend T_1 by another edge incident on v. Thus we can continue to extend T_1 until T_1 returns to its initial vertex u, i.e., until T_1 is closed.

If T_1 includes all the edges of G, then T_1 is our Euler cycle. If T_1 does not include all the edges of G. Consider the graph H obtained by deleting all edges of T_1 from G. H may not be connected, but each vertex of H has even degree since T_1 contains an even number of the edges incident on any vertex. Since G is connected, there is an edge e' of H which has an endpoint u' in T_1. We construct a path T_2 in H beginning at u' and using e'. Since all vertices in H have even degree, we can continue to extend T_2 in H until T_2 returns to u'. We can clearly put T_1 and T_2 together to form a larger closed path in G. We can continue this process until all edges of G are used, and obtained an Eulerian cycle, and so G is Eulerian. $\Box$

Theorem. A connected graph has an Euler path but not an Euler cycle if and only if it has exactly two vertices of odd degree.

Example: use Eulerian paths and cycles to solve the graph puzzles that ask you to draw a picture in a continuous motion without lifting a pencil so that no parts of the pictures is retraced.

6. A **Hamiltonian path (cycle)** in G is a path (cycle) that containing every **vertex** of G exactly once.

   Example (traveling salesperson problem): is there a simple cycle contains every vertex exactly once?

   Although it is clear that only connected graphs can be Hamiltonian, there is no simple criterion to tell us whether or not a graph is Hamiltonian as there is for Eulerian graphs. We have the following sufficient condition.

   Dirac's Theorem. Let G be a simple graph with n vertices and $n \geq 3$ such that the degree of every vertex in G is at least $n/2$, then G has a Hamilton cycle.

   Amazing fact: there are no known efficient algorithms to decide if a graph is Hamiltonian. Most computer scientists believe that no such algorithm exists.

IV. Planar graphs

1. A graph is called **planar** if it can be drawn in the plane without any edges crossing. Such a drawing is called a *planar representation* of graph.

   Questions: (a) Is K_4 planar? (b) Is Q_3 planar? (c) Is $K_{3,3}$ planar?

2. A pictural representation of a planar graph splits the plane into *regions (faces)*, including an unbounded region.

   Euler's formula: let G be a connected planar graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $v - e + r = 2$.

Proof. Suppose G consists of a single vertex v. Then $v = 1$, $e = 0$ and $r = 1$. Hence $v - e + r = 1 - 0 + 1 = 2$. Otherwise G can be built up from a single vertex by the following two constructions:

(a) Add a new vertex w, and connect it to an existing vertex u by an edge e which does not cross any existing edge.

(b) Connect two existing vertices w and v by an edge e which does not cross any existing edge.

Neither operations changes the value of $v - e + r$. Hence G has the same value of $v - e + r$ as G consisting of a single vertex, that is, $v - e + r = 2$. Thus the theorem is proved. $\square$

3. A graph $G = (V, E)$ is k**-colorable** if we can paint the vertices using "colors" $\{1, 2, \ldots, k\}$ such that no adjacent vertices have the same color.

   Theorem (Appel and Haken, 1976). Every planar graph is 4-coloable.

4. If a graph is planar, so will be any graph obtained by removing an edge $\{u, v\}$ and adding a new vertex $\{w\}$ together with edges $\{u, w\}$ and $\{w, v\}$. Such an operation called **elementary subdivision**.

   Two graphs G_1 and G_2 are called **homeomporhic** if they can be obtained from the same or isomorphic graph by a sequence of elementary subdivisions.

   Kuratowaski's Theorem. A graph is **nonplanar** if and only if it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

V. Trees

1. A **tree** is a connected graph with no cycles.

2. **Theorem.** Let G be a graph with $n \geq 1$ vertices. Then the following statements are equivalent:

   (a) G is a tree.

   (b) G is a cycle-free (acyclic) and has $n - 1$ edges

   (c) G is connected and has $n - 1$ edges.

   Proof: by mathematical induction on n. The theorem is certainly true for the graph with only one vertex and hence no edges. That is the theorem holds for $n = 1$. We now assume that $n > 1$ and that the theorem holds for graphs with less than n vertices.

   (a) ⇒ (b): Suppose G is a tree. Then G is cycle-free, so we only need to show that G has $n - 1$ edges. Since G is cycle-free, G has a vertex of degree 1. Delete this vertex and its edge, we obtain a tree T which has $n - 1$ vertices. By the hypothesis, T has $n - 2$ edges. Then G has $n - 1$ edges.

   (b) ⇒ (c): Suppose G is a cycle-free (acyclic) and has $n - 1$ edges. We show that G is connected. Suppose T is disconnected and has k components $T_1, \ldots, T_k$, which are trees since each is connected and cycle-free. Say T_i has n_i vertices. Note that $n_i < n$. Hence the theorem holds for T_i, so T_i has $n_i - 1$ edges. Thus $n = n_1 + n_2 + \cdots + n_k$ and $n - 1 = (n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) = n - k$. Hence $k = 1$. But this contradicts the assumption that G is disconnected and has $k > 1$ components. Hence G is connected.

(c) $\Rightarrow$ *(a)*: Suppose G is connected and has $n-1$ edges. We need to show that G is cyclc-free. Suppose a cycle containing an edge e. Delete e we obtain the graph $H = G - e$ which is also connected. But H has n vertices and $n-2$ edge, and this contradicts the fact that a connected graph with n vertices must have at least $n-1$ edges. Thus G is cycle-free and hence a tree. $\square$

3. A **rooted tree** is a tree in which a particular vertex is designated as the root.

   The terminology for trees has botanical and genealogical origins – parent, child, sibilings, ancestors and descendants.

4. An rooted tree is called an m**-ary tree** if every internal vertex has no more than m children.

   The tree is called a **full** m**-ary tree** if every internal vertex has exactly m children.

   An m-ary tree with $m = 2$ is called a **binary tree**.

5. A *binary search tree* T is a binary tree in which data are associated with vertices. Furthermore, the data are assigned so that for each vertex v in T, each data items in the left of v is less than the data item in v, and each item in the right subtree of v is greater than the data item in v.

   - Construct a binary search tree T.

     Example: form a binary search tree for the following words using alphabetical order:

     `mathematics, physics, geography, zoology, meterology, psychology, chemistry`

   - Searching data in a binary search tree T.

   - Complexity of the worst-case adding and searching of a binary search tree: if T has n vertices, then adding or searching a data item requires no more than $\lceil \log n \rceil$ comparisons. For example, if $n = 1,000,000$, then $\lceil \log n \rceil = 21$.