Discrete Math for Computer Science

Peter Schaefer

Freshman - Kutztown

Contents

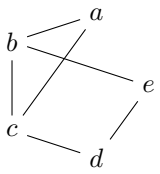
1 Graphs

1.1 Introduction to Graphs

Graphs are fundamental objects in discrete mathematics that model relationships between pairs of objects. Graphs arise in a wide array of disciplines but play an especially important role in computer science.

In an **undirected graph**, the edges are unordered pairs of vertices, which is useful for modeling relationships that are symmetric. A graph consists of a pair of sets (V, E), where V is a set of vertices and E is a set of edges. A graph is **finite** if the vertex set is finite. A single element of V is called a **vertex** and is usually represented pictorially by label, or a dot with a label. Each edge in E is a set of two vertices from V and is drawn as a line connecting the two vertices.

An undirected graph

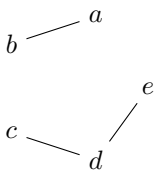


In the above graph, two edges cross each other, but there is no vertex at the crossing. The crossing is just a byproduct of how the graph is drawn on a two-dimensional surface. The graph above can be described by listing the vertex set and the edge set:

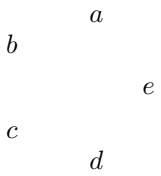
$$V = \{a, b, c, d, e\}$$
$$E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$

A graph may appear to be disconnected into more than one piece but is still considered to be one graph. Here are two graphs, each with 5 vertices.

Graph A

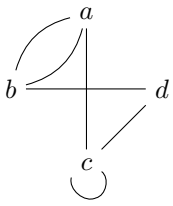


Graph B



Parallel edges are multiple edges between the same pair of vertices. In defining graphs with parallel edges, it *might* be important to have additional label besides the two endpoints to specify an edge in order to distinguish between different parallel edges. A graph can also have a **self-loop** which is an edge between a vertex, and itself.

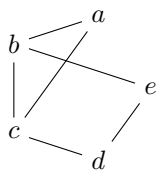
A graph with parallel edges and a self-loop



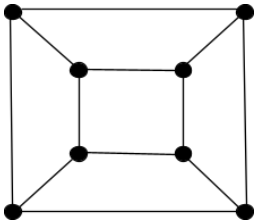
If a graph does not have parallel edges or self-loops, it is said to be a **simple graph**.

Graph terminology

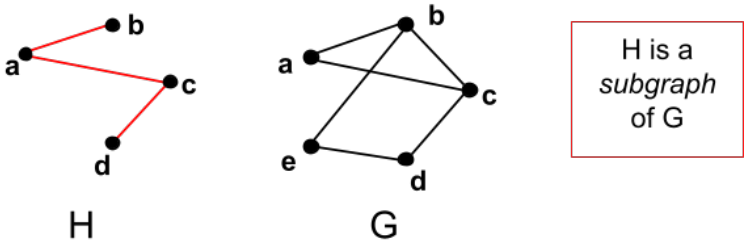
Undirected graph example



- If there is an edge between two vertices, they are said to be **adjacent**. In the graph above, d and e are adjacent, but d and b are not adjacent.
- Vertices b and e are the **endpoints** of edge $\{b, e\}$. The edge $\{b, e\}$ is **incident** to vertices $btande$.
- A vertex c is a **neighbor** of vertex b if and only if $\{b, c\}$ is an edge. In the graph above, the neighbors of b are the vertices $a, c,$ and e.
- In a simple graph, the **degree** of a vertex is the number of neighbors it has. In the graph above, the degree of b is 3 and the degree of vertex a is 2. The degree of vertex b is denoted by $\deg(b)$.
- The **total degree** of a graph is the sum of the degrees of all of the vertices. The total degree of the above graph is $2 + 3 + 3 + 2 + 2 = 12$.
- In a **regular graph**, all the vertices have the same degree. In a d**-regular graph**, all the vertices have degree d. The graph above is not regular because $\deg(a) \neq \deg(b)$. However, the graph below is 3-regular.



- A graph $H = (V_H, E_H)$ is a **subgraph** of a graph $G = (V_G, E_G)$ if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. Note that any graph G is a subgraph of itself.



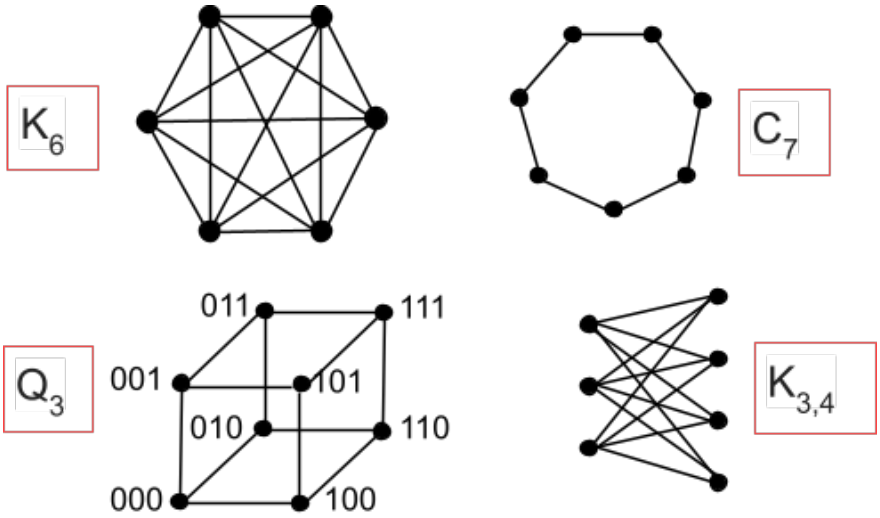
Theorem: Number of Edges and Total Degree

Let $G = (V, E)$ be an undirected graph, simple or not. Then, twice the number of edges is equal to the total degree:

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

Common Graphs

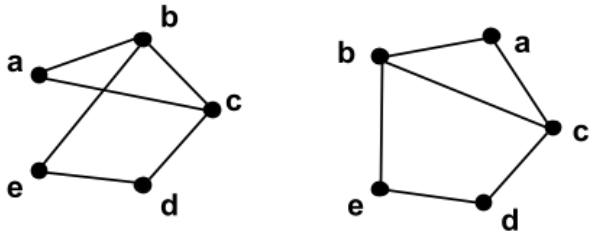
Some graphs with special structure are given their own name and notation because they come up so frequently in graph theory.



For the definition of all of these graphs, $n \in \mathbb{Z}^+$.

- K_n is called the **complete graph** on n vertices. K_n has an edge between every pair of vertices. K_n is sometimes called a **clique** of size n or an n**-clique**.
- C_n is called a cycle on n vertices. The edges connect the vertices in a ring. Note that C_n is well defined only for $n \geq 3$.
- The n-dimensional hypercube, denoted Q_n, has 2^n vertices. Each vertex is label with an n-bit string. Two vertices are connected by an edge if their corresponding labels differ only by one bit.
- $K_{n,m}$ has $n + m$ vertices, and $2nm$ edges. The vertices are divided into two sets: one with m vertices and one set with n vertices. There are no edges between vertices in the same set, but there is an edge between every vertex in one set and every vertex in the other set.

1.2 Graph representations



These two graphs look different, but that is only because they are drawn differently. The two graphs are actually the same graph because they have the same vertex and edge sets as shown below:

$$V = \{a, b, c, d, e\}$$
$$E = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$

The way a graph is drawn is not part of the graph itself.

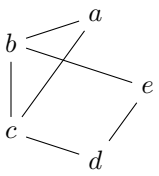
Adjacency List Representation for Graphs

In the **adjacency list representation** of a graph, each vertex has a list of all its neighbors. Note that since the graph is undirected if vertex a is in b's list of neighbors, then b must also be in a's list of neighbors.

If a graph is represented using adjacency lists, the time required to list the neighbors of a vertex v is proportional to $\deg(v)$, the number of vertices to be listed. In order to determine if $\{a, b\}$ is an edge, it is necessary to scan the list of a's neighbors or the list of b's neighbors. In the worst case, the time required is proportional to the larger of $\deg(a)$ or $\deg(b)$.

Matrix Representation for Graphs

The **matrix representation** for a graph with n vertices is an $n \times n$ matrix whose entries are all either 0 or 1, indicating whether or not each edge is present. The matrix representation of an undirected graph is symmetric.



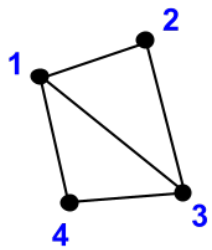
$$\Rightarrow \quad \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

1.3 Graph isomorphism

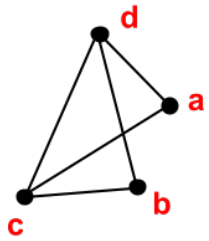
Two graphs are said to be **isomorphic** if there is a correspondence between the vertex sets of each graph such that there is an edge between two vertices of one graph if and only if there is an edge between the corresponding vertices of the second graph. Essentially, the graphs are not identical but the vertices can be relabeled so that they are identical.

Definition of Isomorphic Graphs

Let $G = (V, E)$ and $G' = (V', E')$. G and G' are isomorphic if there is a bijection $f : V \to V'$ such that for every pair of vertices $x, y \in V$, $\{x, y\} \in E$ if and only if $\{f(x), f(y)\} \in E'$. The function f is called an **isomorphism** from G to G'.



G_1



G_2

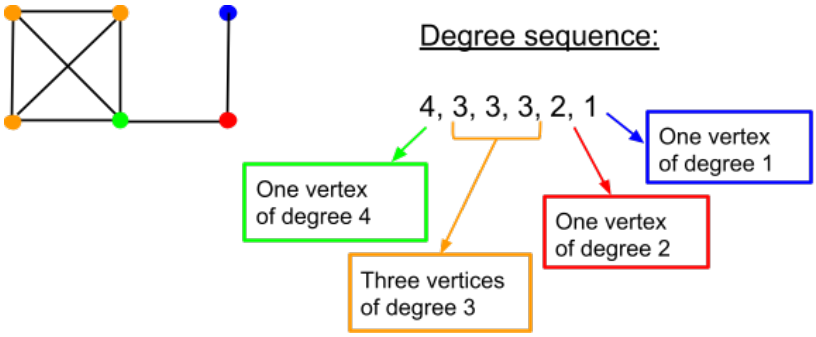
The following function is an isomorphism from the vertices of G_1 to G_2:

$$f(1) = d \qquad f(2) = a \qquad f(3) = c \qquad f(4) = b$$

Theorem: Vertex Degree Preserved under Isomorphism

Consider two graphs, G and G'. Let f be an isomorphism from G to G'. For each vertex v in G, the degree of vertex v in G is equal to the degree of vertex $f(v)$ in G'.

The **degree sequence** of a graph is a list of the degree of all of the vertices in non-increasing order.



Theorem: Degree Sequence Preserved under Isomorphism

The degree sequence of a graph is preserved under isomorphism.

1.4 Walks, trails, circuits, paths, and cycles

A **walk** from v_0 to v_l in an undirected graph G is a sequence of alternating vertices and edges that starts and ends with a vertex:

$$\langle v_0, \{v_0, v_1\}, v_1, \{v_1, v_2\}, \ldots, v_{\ell-1}, \{v_{\ell-1}, v_\ell\}, v_\ell \rangle$$

Since the edges in a walk are completely determined by the vertices, a walk can also be denoted by the sequence of vertices:

$$\langle v_0, v_1, \ldots, v_\ell \rangle.$$

However, keep in mind the sequence of vertices is a walk only if $\{v_{j-1}, v_j\} \in E$ for each $j = 1, 2, \ldots, \ell$. The **length of a walk** is ℓ, the number of edges in the walk. An **open walk** is a walk in which the first and last vertices are not the same. A **closed walk** is a walk in which the first and last vertices are the same.

- A **trail** is a walk in which no *edge* occurs more than once.
- A **path** is a walk in which no *vertex* occurs more than once.
- A **circuit** is a closed *trail*.
- A **cycle** is a *circuit* is length at least 1 in which no vertex occurs more than once, except the first and last vertices which are the same.

Here are some examples of closed walks:

- $\langle A, C, D, A \rangle$ is a *trail*, a *circuit*, and a *cycle*.
- $\langle A, C, B, A, D, E, A \rangle$ is a *trail* and a *circuit*.
- $\langle A, B, A \rangle$ is **not** a trail, circuit, or a cycle.

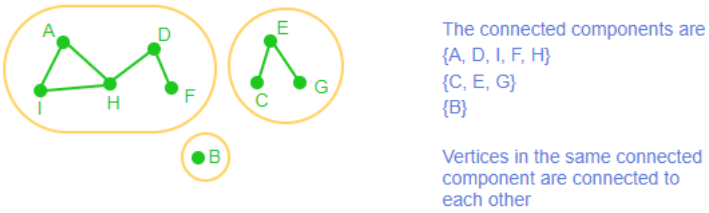
Since paths and cycles do not have any repeated edges, if a graph is simple, any cycle *must* have length at least 3. The sequence $\langle v \rangle$ is not a cycle because a cycle, by definition, must have length at least 1. The sequence $\langle v, v \rangle$ is only a walk if there is a self-loop at vertex v.

1.5 Graph connectivity

Two vertices, v and w, are **connected** if there is a path from v to w (and thus also a path from w to v). A vertex is always considered to be connected to itself. The property of being connected can be extended to sets of vertices and the entire graph:

- A set of vertices in a graph is said to be connected if every pair of vertices in the set is connected.
- A graph is said to be connected if every pair of vertices in the graph is connected, and is **disconnected** otherwise.

A **connected component** consists of a *maximal* set of vertices that are connected as well as all the edges between two vertices in the set. A vertex that is not connected with any other vertex is called an **isolated vertex** and is therefore a connected component with only one vertex.



k-Connectivity

In some networks, it is important to be able to guarantee connectivity, even if one or more vertices or edges are removed from a graph. The definition of connectivity can be extended to encompass resilience to vertex or edge failures.

Definition of a K-vertex-connected Graph

An undirected graph G is **k-vertex-connected** if the graph contains at least $k+1$ vertices and remains connected after any $k-1$ vertices are removed from the graph. The **vertex connectivity** of a graph is the largest k such that the graph is k-vertex-connected. The vertex connectivity of a graph G is denoted $\kappa(G)$.

The vertex connectivity of a graph is the minimum number of vertices whose removal disconnects the graph into more than one connected component.

When the graph is a complete graph, there is no set of vertices whose removal disconnects the graph. For the special case of K_n, the vertex connectivity $\kappa(K_n)$ is just defined to be $n-1$.

Definition of a K-edge-connected Graph

An undirected graph G is **k-edge-connected** if removing any $k-1$ or fewer edges results in a connected graph. The **edge connectivity** of a graph is the largest k such that the graph is k-edge-connected. The edge connectivity of a graph G is denoted $\lambda(G)$.

The edge connectivity of a graph is the minimum number of edges whose removal disconnects the graph into more than one connected component.

Theorem: Upper bound for Vertex and Edge Connectivity

Let G be an undirected graph. Denote the minimum degree of any vertex in G by $\delta(G)$. Then,

$$\kappa(G) \leq \delta(G) \quad \text{and}$$
$$\lambda(G) \leq \delta(G).$$

1.6 Euler circuits and trails

1.7 Hamiltonian cycles and paths

1.8 Planar coloring

1.9 Graph coloring

2 Trees