Table of Contents

[Familiar Graphs 2](#_Toc110950633)

[Terminology 2](#_Toc110950634)

[Graph 2](#_Toc110950635)

[Vertex 2](#_Toc110950636)

[Edge 2](#_Toc110950637)

[Example 3](#_Toc110950638)

[Path 3](#_Toc110950639)

[Graph Classifications 4](#_Toc110950640)

[Directed Graphs 4](#_Toc110950641)

[Undirected Graphs 5](#_Toc110950642)

[Weighted and Unweighted 6](#_Toc110950643)

[Reachable, Connected, and Complete 6](#_Toc110950644)

[Reachable Node 6](#_Toc110950645)

[Connected Graphs 6](#_Toc110950646)

[**Connected Undirected Graph** 7](#_Toc110950647)

[**Strongly Connected Directed Graph** 7](#_Toc110950648)

[**Weakly Connected Directed Graph** 8](#_Toc110950649)

[Disconnected Graphs 8](#_Toc110950650)

[Complete Graphs 8](#_Toc110950651)

[Loops and Cycles 9](#_Toc110950652)

[Linked Lists, Trees, Graphs 9](#_Toc110950653)

# **Familiar Graphs**

* You are undoubtedly familiar with graphs. Here are some common examples:
  + line graphs,
  + bar graphs,
  + pie charts,
  + etc…
* The following figure is a simple line graph. This is an example of a type of graph.
* It is a set of points that are joined by lines.

Chart, line chart

Description automatically generated

* Clearly, graphs provide a way to illustrate data.
* However, graphs also represent the relationships among data items, and it is this feature of graphs that is important here.

# **Terminology**

## **Graph**

* A graph ***G = (V, E)***is defined by a pair of sets:

1. V = a set of **vertices**
2. E = a set of **edges**

## **Vertex**

* The set ***V*** inside of a graph ***G = (V, E)*** is defined by a **set of unique vertices** (nodes).

## **Edge**

* The set ***E*** inside of a graph ***G = (V, E)*** is defined by a **pair** **of vertices**.
* A pair of vertices can be represented like (*v*, *w*), where *v*, *w* ∈ *V*.
  + The symbol ∈ indicates set membership and means “is an element of”.
  + The statement (*v*, *w*)∈ *V* means both vertices in the pair *v*, *w* are in the set *V*.
* Edges can be labeled (weighted) or unlabeled (unweighted)
  + e.g., nodes are cities, edges are the roads between cities, edge label is the distance of the road.
* If the graph contains an edge (*v*, *v*) from a vertex to itself, then the path *v*, *v* is sometimes referred to as a **loop**.

## **Example**

A picture containing text, clock, clipart

Description automatically generated

V = {a,b,c,d}

E = {(a,c), (b,c), (b,d), (c,d)}

## **Path**

* A ***path*** from vertex ***a*** to ***b*** is a **sequence of edges** that can be followed starting from ***a*** to reach ***b***.
  + Can be represented as edges taken
  + Can be represented as vertices visited **{*v*1, *v*2, *v*3,...,*vN}***such that

(*vi*, *vi*+1) ∈ *E* for 0 ≤ *i* < *N*

* Note that there can be **multiple possible paths** between any two nodes.
* The ***path*** ***weight*** is the **sum of the weights of the edges**.
  + For a weighted graph, find the shortest path (using edge weights).
* The ***path*** ***length*** is the **number of vertices or edges on the path**.
  + For an unweighted graph, find the shortest path (fewest edges required).
* Two vertices are ***neighbors*** or ***adjacent*** if they are connected by an edge.

A picture containing text, watch, clock

Description automatically generated

**Examples**:

**V to Z:**

{b, h} or {V, X, Z}

**List two possible paths from U to Y:**

{c, f} or {a, b, g} …

{U, W, Y} or {U, V, X, Y} …

**Are V and X adjacent nodes?**

Yes

* A ***simple path* repeats no vertices** (except the 1st can be the last).
  + p = {Seattle, Salt Lake City, San Francisco, Dallas}
  + p = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}
* A ***cycle*** is a path that **starts and ends at the same vertex**.
  + p = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}
* A ***simple cycle*** is a cycle that does notrepeat edges or vertices, except the **first is also the last**.
* A graph with no cycles is called an **Acyclic Graph**.
  + A directed graph with no cycles is called a **Directed** **Acyclic Graph**.
  + A graph with no cycles is called an **Undirected** **Acyclic Graph**.

# **Graph Classifications**

* There are many different types of graphs.
* Commonly, graphs fall into the following categories:
  1. Directed or Undirected
  2. Reachable, Connected, Unconnected, or Complete
  3. Weighted or Unweighted
  4. Cyclic or Acyclic
* Choose the kinds required for problem and determined by data

## **Directed Graphs**

**Directed Graph Definition**

* Edges are a ***one-way*** connection between vertices *v* 🡪 *w*.
* Vertex *w* is **adjacent** to *v* if and only if (*v*, *w*) ∈ *E*.
  + *w* is adjacent to *v*,but *v* is not adjacent to *w*.
* In a digraph (directed graph), an edge is an ordered pair
  + Thus: (u,v) and (v,u) are not the same edge

**Degree**

* The degree of a vertex is the number of edges touching it.
* In a directed graph, each vertex has a separate

1. **In-degree**: Number of incoming edges
2. **Out-degree**: Number of outgoing edges

**Directed Graph Diagram**

* Edge drawn as arrow
* Edge can only be traversed in direction of arrow

**Example**:

* The set of edges E = {(A,B), (A,C), (A,D), (B,C), (D,C)}
  + (A,B) ∈ E
  + (D,C) ∈ E, (C,D) ∉ E
  + A node can have an edge to itself (eg (A,A) is valid)

A picture containing text, clock, watch

Description automatically generated

## **Undirected Graphs**

**Undirected Graph Definition**

* Each edge can be traversed in either direction *v* 🡪 *w* or *w* 🡪 *v*.
  + An undirected graph with edge (*v*, *w*) also has edge (*w*, *v*).
* Vertex *w* is **adjacent** to *v* if and only if (*v*, *w*) ∈ *E*.
  + *w* is adjacent to *v* and *v* is adjacent to *w*.
* **Undirected Graph Diagram**
  + Edges have no direction, so there are no arrows
  + We can traverse an edge in either direction

**Example**:

* The set E = {

(A,B), (A,C), (A,D),

(B,A), (B,D),

(C,A), (C, D),

(D,A), (D,B), (D,C),

}

* A is adjacent to B and B is adjacent to A
* If (A,B) ∈ E then (B,A) ∈ E

A picture containing clock, clipart

Description automatically generated

## **Weighted and Unweighted**

* Graphs can be classified by whether or not their edges have weights.
* **Weighted Graph**: edges have an associated cost
  + Weight typically shows cost of traversing
  + Example: weights are distances between cities

A picture containing diagram

Description automatically generated

Example of an undirected, weighted graph.

* **Unweighted graph**: edges can be thought of as having equal weight (e.g., all 0, all 1, etc.)
  + Edges simply show connections
  + Example: flight map between airports

A picture containing diagram

Description automatically generated

Example of a directed, unweighted graph.

# **Reachable, Connected, and Complete**

## **Reachable Node**

* Vertex ***a*** is *reachable* from ***b*** is a **path exists** from ***a*** to ***b***.

## **Connected Graphs**

* A graph is **connected** if every vertex is **reachable** from any other vertex.
  + Each pair of *distinct vertices* *has a path* between them.
* In a connected graph, you can get from any vertex to any other vertex by following a path.
* Notice that a connected graph does not necessarily have an edge between every pair of vertices.

### **Connected Undirected Graph**

* An **undirected** graph is **connected** if there is a path from every vertex to every other vertex.

A picture containing watch, clock

Description automatically generated A picture containing text, watch

Description automatically generated

A picture containing text, clock, watch

Description automatically generated

### **Strongly Connected Directed Graph**

* A **directed** graph is **strongly connected** if there is a directed path from every vertex to every other vertex.

A picture containing text, clock, antenna

Description automatically generated

* Strongly Connected

A picture containing text, clock

Description automatically generated

* Strongly Connected

A picture containing text, clock

Description automatically generated

* Not Strongly Connected

### **Weakly Connected Directed Graph**

* If a directed graph is not strongly connected, but the underlying graph works as an undirected graph, then the graph is said to be **weakly connected**.

A picture containing clock

Description automatically generated

* Weakly Connected

## **Disconnected Graphs**

* A **disconnected graph** is a graph in which there are **any** **two vertices that do not have a connecting path** between them.

Chart

Description automatically generated

## **Complete Graphs**

* In a **complete graph**, each pair of distinct vertices has an edge between them.
* This means that all vertices **have** **an edge with every other vertex in the graph**.
* A **complete graph is also connected**, but the converse is not true.

Shape, polygon

Description automatically generated

# **Loops and Cycles**

* **Cycle**: a path that begins and ends at the same node.
  + Example: {b, g, f, c, a} or {V, X, Y, W, U, V}
  + Example: {c, d, a} or {U, W, V, U}
* **Acyclic Graph**: A graph that does not contain any cycles
* **Loop**: A vertex has an edge with itself.
  + many graphs don’t allow loops.

A picture containing text, watch

Description automatically generated

# **Linked Lists, Trees, Graphs**

* A **linked list** is a graph with some restrictions
  + A linked list is an unweighted, directed, acyclic graph (DAG)
  + Each node has an in and out degree of at most 1.

A picture containing text, clipart

Description automatically generated

* A **binary tree** is a graph with some restriction
  + A binary tree is an unweighted, directed, acyclic graph (DAG)
  + Each node has an in-degree of at most 1, and an out-degree of at most 2.
  + There is exactly one path form the root to every other node in the tree.

A picture containing diagram

Description automatically generated