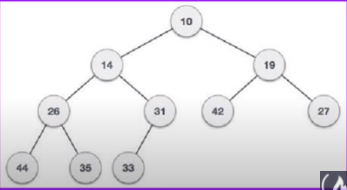
**Heaps – Introduction:**

* Think back to Binary Search Tree:
  + Each Node can have no more than 2 Children Max
  + Child to the left has a value <= the Parent Node
  + Child to the right has a value >= the Parent Node
  + No two Nodes can contain the same value

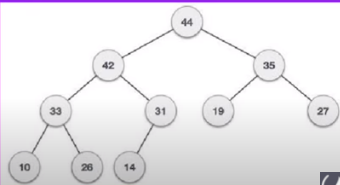
**A Heap:**

* A Heap is a special Tree where all Parent Nodes **compare** to their Child Nodes in some way by being **more** or **less** extreme
  + They will be either **greater than** or **less than** their Child Nodes
  + Whether the Parent is greater or less than its Child Nodes **determines where the data is stored**
  + It is usually dependent on the Parent Node’s value

**Min-Heaps:**

* A **Min-Heap** is a Tree where the value of the **Root Node** is less than every one of its Child Nodes
* This **must be true recursively** for all other Parent Nodes in the Tree (they must all always be less than their children).
* 

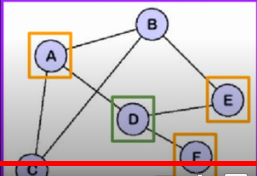
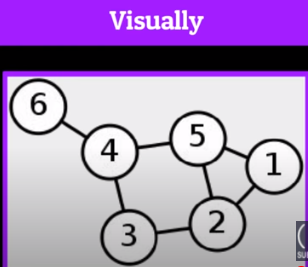
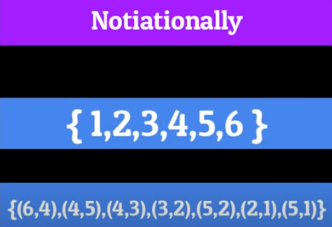
**Max-Heaps:**

* Exact opposite of Min-Heaps. Value at the root must be **greater** than the values of each of its Child Nodes
* Same **must be true recursively throughout the Tree. Example below:**
* 

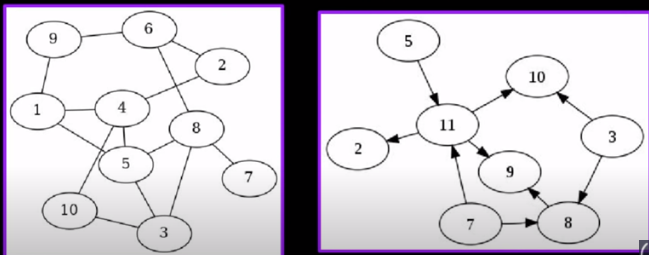
**Heaps – Implementation and Use Case:**

* Heaps are most commonly used in implementing HeapSort
  + HeapSort is a sorting algorithm that takes in a list of elements, builds them into a min or max heap, and then removes the Root Node continuously to make a sorted list.
* **Priority Queues:** 
  + An advanced Data Structure your computer uses to designate tasks and assign computer power based on how urgent the matter is

**The Graph:**

* Graphs are **pieces of information** and the **paths that run between them**
* It is a **nonlinear** Data Structure consisting of **Nodes** and **Edges**
  + There is a **finite** set of **Nodes**
  + **Nodes** are connected by the **Edges**
  + **Every Node** is connected to its surrounding Nodes, except that the final Node doesn’t branch off to any new Nodes, which makes sense…
* Graphs have **multiple starting points** and **multiple branches.**
* 
*  
* **The above two examples shows what a graph looks like when we visualize it (left) and when we write it out in notation (right)**
* In the example above, the first set of curly brackets represents **the number of every Node (or vertice) in the Graph.**
* The bottom set of tuples in curly brackets is called an **Edge Set**. It represents all of the **Node/Edge** relationships in the Graph.
* For example, 6 shares an Edge with 4 so (6, 4). 4 also shares Edges with 5 and 3, so (4, 5) and (4,3), 2 shares Edges with 3, 5, and 1 so (3, 2), (5,2), (2,1)
* If an Edge connects more than one Node/Vertice, it is said to be **adjacent** to the others. So in the example above, 5 is adjacent to 4, 2, and 1 Nodes.

**Directed vs. Undirected Graphs:**

* **Undirected Graph:**
  + A graph in which the direction you traverse the nodes **ISN’T** important
  + Usually indicated by a lack of arrows
  + Any of the above Graphs are Undirected Graphs
* **Directed Graph:**
  + A graph in which the direction you traverse the Nodes **IS** important
  + Usually indicated by arrows that point to which Nodes a certain is allowed to traverse to
  + Edges can point both ways, but don’t have to
* 
* **In the example above, the picture on the left shows an Undirected Graph, where the picture on the right shows a Directed Graph**

**Graphs – Cyclic vs. Acyclic:**

* **Cyclic Graphs:**
  + A cyclic graph is a Node that contains a **path** from **at least** one Node back to itself
  + All **Undirected Graphs** are cyclical
* **Acyclic Graphs:**
  + An acyclic graph is one that contains **no path** from any given Node back to itself
  + These can really only be **Directed Graphs** because they aren’t all connected together to start.

**Weighted Graphs:**

* The process of associating a **numerical value** with each **Edge** (cost)
* Each weight represents some property of the information you’re trying to convey.

**Types of Graphs:**

* **Undirected Cyclical Heaps with Weighted Eges** can be used through **Dijkstra’s shortest path algorithm**
* Compiles a **list** of the shortest possible paths from that source vertex to all other Nodes in the graph
* This graph and algorithm is used in Google Maps, IP Routing, and potentially even telephone services
* **Unweighted Cyclical Graphs (Directed and Undirected)** are used in the “follower” system of a majority of social media networks:

**Facebook, Snapchat, Instagram, Twitter**