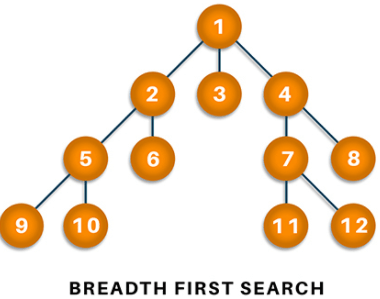
**Graph based Algorithms**

### **Breadth-First Search**

Traversing or searching is one of the most used operations that are undertaken while working on graphs. Therefore, in **breadth-first-search** (BFS), you start at a particular vertex, and the algorithm tries to visit all the neighbors at the given depth before moving on to the next level of traversal of vertices. Unlike trees, graphs may contain cyclic paths where the first and last vertices are remarkably the same always. Thus, in BFS, you need to keep note of all the track of the vertices you are visiting. To implement such an order, you use a queue data structure which First-in, First-out approach. To understand this, see the image given below.



**Algorithm**

1. Start putting anyone vertices from the graph at the back of the queue.
2. First, move the front queue item and add it to the list of the visited node.
3. Next, create nodes of the adjacent vertex of that list and add them which have not been visited yet.
4. Keep repeating steps two and three until the queue is found to be empty.

**Applications**

BFS algorithm has various applications. For example, it is used to determine the **shortest path** and **minimum spanning tree.** It is also used in web crawlers to creates web page indexes. It is also used as powering search engines on social media networks and helps to find out peer-to-peer networks in BitTorrent.

### **Depth-first search**

In depth-first-search (DFS), you start by particularly from the vertex and explore as much as you along all the branches before backtracking. In DFS, it is essential to keep note of the tracks of visited nodes, and for this, you use stack data structure.

**Algorithm**

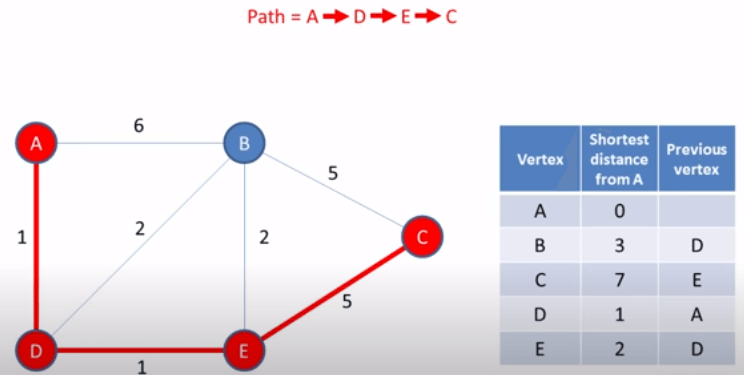
1. Start by putting one of the vertexes of the graph on the stack's top.
2. Put the top item of the stack and add it to the visited vertex list.
3. Create a list of all the adjacent nodes of the vertex and then add those nodes to the unvisited at the top of the stack.
4. Keep repeating steps 2 and 3, and the stack becomes empty.

**Applications**

DFS finds its application when it comes to finding paths between two vertices and detecting cycles. Also, **topological sorting** can be done using the DFS algorithm easily. DFS is also used for one-solution puzzles.

### **Dijkstra's shortest path algorithm**

Dijkstra's shortest path algorithm works to find the minor path from one vertex to another. The sum of the vertex should be such that their sum of weights that have been travelled should output minimum. The shortest path algorithm is a highly curated algorithm that works on the concept of receiving efficiency as much as possible. Consider the below diagram.



**Algorithm**

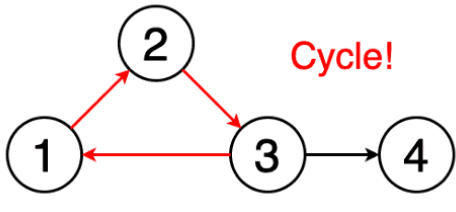
1. Set all the vertices to infinity, excluding the source vertex.
2. Push the source in the form (distance, vertex) and put it in the min-priority queue.
3. From the priority, queue pop out the minimum distant vertex from the source vertex.
4. Update the distance after popping out the minimum distant vertex and calculate the vertex distance using (vertex distance + weight < following vertex distance).
5. If you find that the visited vertex is popped, move ahead without using it.
6. Apply the steps until the priority queue is found to be empty.

**Applications**

Dijkstra's shortest path algorithm is used in finding the **distance of travel from one location to** another, like **Google Maps or Apple Maps.** In addition, it is highly used in networking to outlay min-delay path problems and abstract machines to identify choices to reach specific goals like the **number game or move to win a match**.

### **Cycle detection**

A cycle is defined as a path in graph algorithms where the first and last vertices are usually considered. For example, if you start from a vertex and travel along a random path, you might reach the exact point where you eventually started. Hence, this forms a chain or cyclic algorithm to cover along with all the nodes present on traversing. Therefore, cycle detection is based on detecting this kind of cycle. Consider the below image.

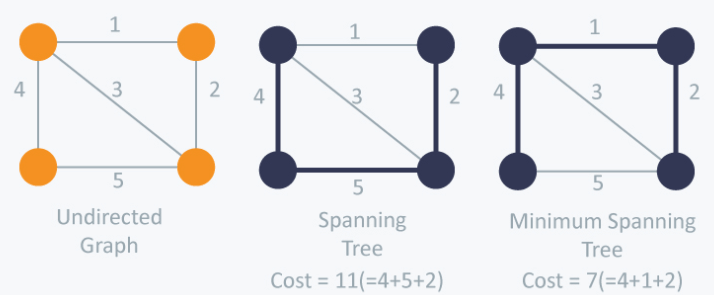


**Applications**

Cyclic algorithms are used in message-based distributed systems and large-scale cluster processing systems. It is also mainly **used to detect deadlocks in the concurrent system** and various cryptographic applications where the keys are used to manage the messages with encrypted values.

### **Minimum Spanning Tree**

A minimum spanning is defined as a subset of edges of a graph having no cycles and is well connected with all the vertices so that the minimum sum is availed through the edge weights. It solely depends on the cost of the spanning tree and the minimum span or least distance the vertex covers. There can be many minimum spanning trees depending on the edge weight and various other factors.

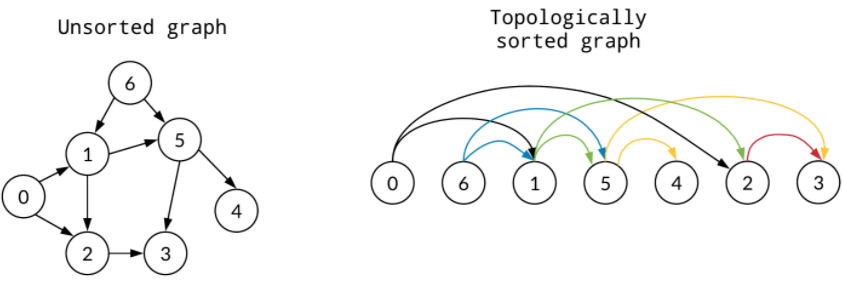


**Applications**

Minimum spanning tree finds its application in the network design and is popularly used in **traveling salesman** problems in a data structure. It can also be used to find the minimum-cost weighted perfect matching and multi-terminal minimum cut problems. MST also finds its application in the field of image and handwriting recognition and cluster analysis.

### **Topological sorting**

Topological sorting of a graph follows the algorithm of ordering the vertices linearly so that each directed graph having vertex ordering ensures that the vertex comes before it. Users can understand it more accurately by looking at the sample image given below.



**Applications**

Minimum spanning tree finds its application in the network design and is popularly used in **traveling salesman** problems in a data structure. It can also be used to find the minimum-cost weighted perfect matching and multi-terminal minimum cut problems. MST also finds its application in the field of image and handwriting recognition and cluster analysis.

### **Maximum flow**

The maximum flow algorithm is usually treated as a problem-solving algorithm where the graph is modeled like a network flow infrastructure. Hence, the maximum flow is determined by finding the path of the flow that has the **maximum flow rate.** The maximum flow rate is determined by augmenting paths which is the total flow-based out of source node equal to the flow in the sink node. Below is the illustration for the same.