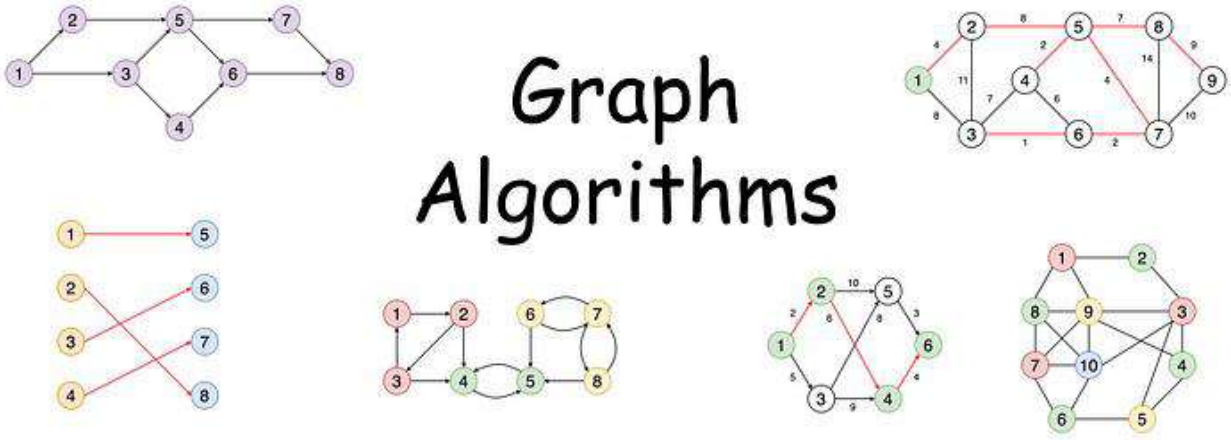
Graph Algorithms

Graphs have become a powerful means of modelling and capturing data in real-world scenarios such as social media networks, web pages and links, and locations and routes in GPS. If you have a set of objects that are related to each other, then you can represent them using a graph.



In this article, I will be briefly explaining 10 basic graph algorithms that become very useful for analysis and their applications.

Firstly, let's introduce a graph.

What is a Graph?

A **graph** consists of a finite set of **vertices** or nodes and a set of **edges** connecting these vertices. Two vertices are said to be **adjacent** if they are connected to each other by the same edge.

Some basic definitions related to graphs are given below. You can refer to Figure 1 for examples.

- **Order:** The number of vertices in the graph
- **Size:** The number of edges in the graph
- **Vertex degree:** The number of edges that are incident to a vertex
- **Isolated vertex:** A vertex that is not connected to any other vertices in the graph
- **Self-loop**: An edge from a vertex to itself
- **Directed graph:** A graph where all the edges have a direction indicating what is the start vertex and what is the end vertex
- **Undirected graph:** A graph with edges that have no direction
- **Weighted graph:** Edges of the graph has weights

- **Unweighted graph:** Edges of the graph has no weights

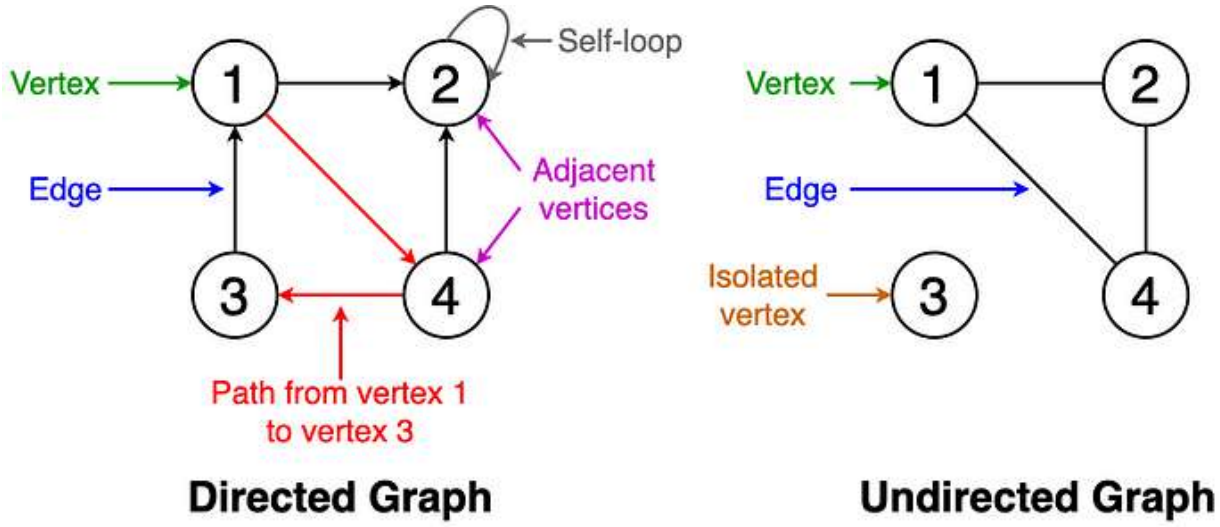


Fig 1. Visualization of Terminology of Graphs (Image by Author)

1. Breadth-first search

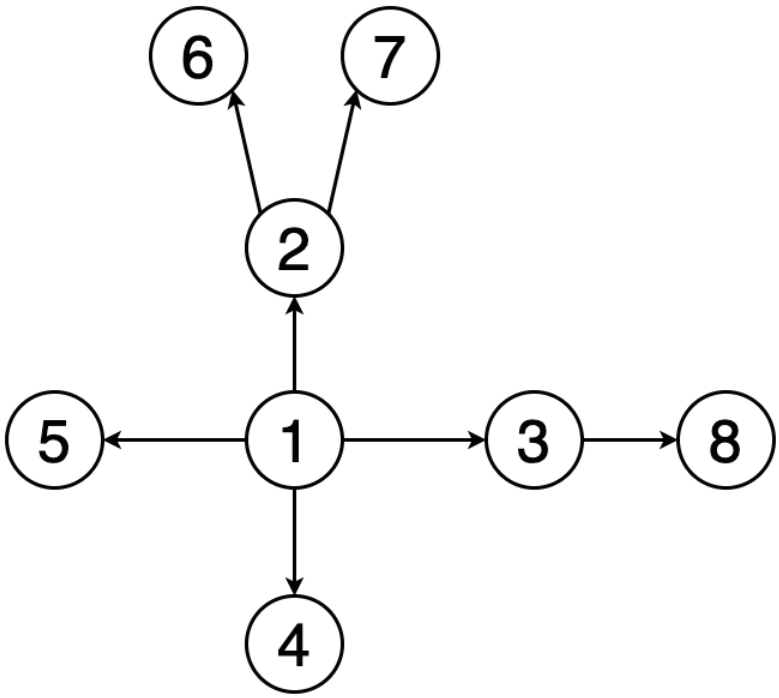


Fig 2. Animation of BFS traversal of a graph (Image by Author)

Traversing or *searching* is one of the fundamental operations which can be performed on graphs. In ***breadth-first search*** (BFS), we start at a particular vertex and explore all of its neighbours at the present depth before moving on to the vertices in the next level. Unlike trees, graphs can contain cycles (a path where the first and last vertices are

the same). Hence, we have to keep track of the visited vertices. When implementing BFS, we use a queue data structure.

Figure 2 denotes the animation of a BFS traversal of an example graph. Note how vertices are discovered (yellow) and get visited (red).

Applications

- Used to determine the shortest paths and minimum spanning trees.
- Used by search engine crawlers to build indexes of web pages.
- Used to search on social networks.
- Used to find available neighbour nodes in peer-to-peer networks such as BitTorrent.

2. Depth-first search

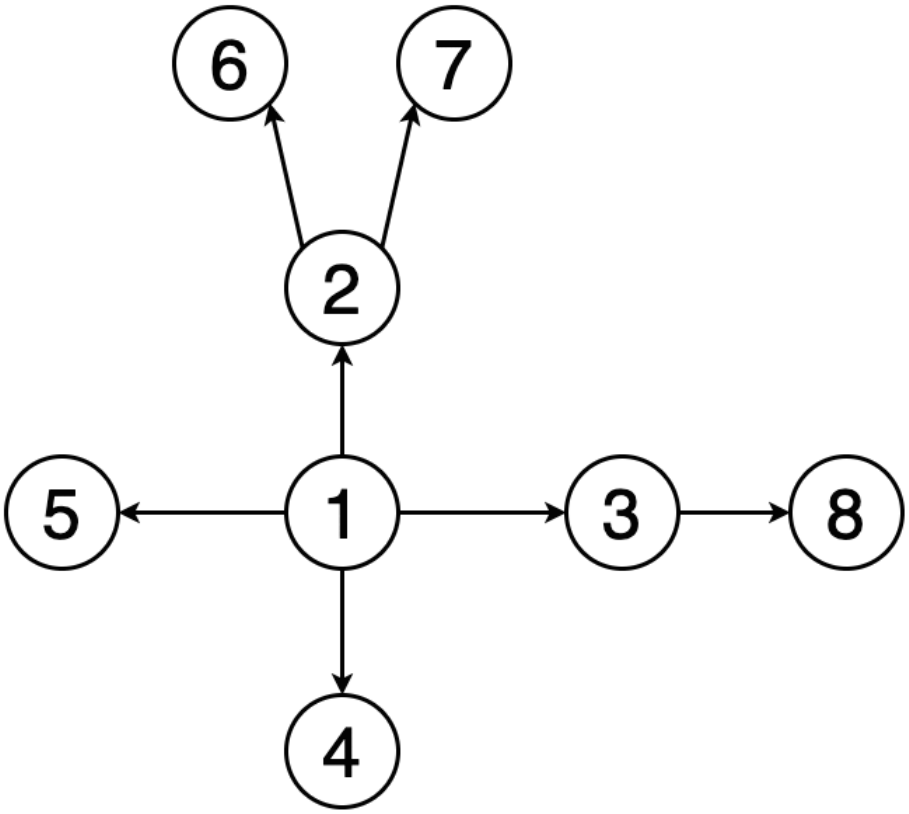


Fig 3. Animation of DFS traversal of a graph (Image by Author)

In ***depth-first search*** (DFS) we start from a particular vertex and explore as far as possible along each branch before retracing back (backtracking). In DFS also we have to keep track of the visited vertices. When implementing DFS, we use a stack data structure to support backtracking.

Figure 3 denotes the animation of a DFS traversal of the same example graph used in Figure 2. Note how it traverses to the depths and backtracks.

Applications

- Used to find a path between two vertices.
- Used to detect cycles in a graph.
- Used in topological sorting.
- Used to solve puzzles having only one solution (e.g., mazes)

3. Shortest path

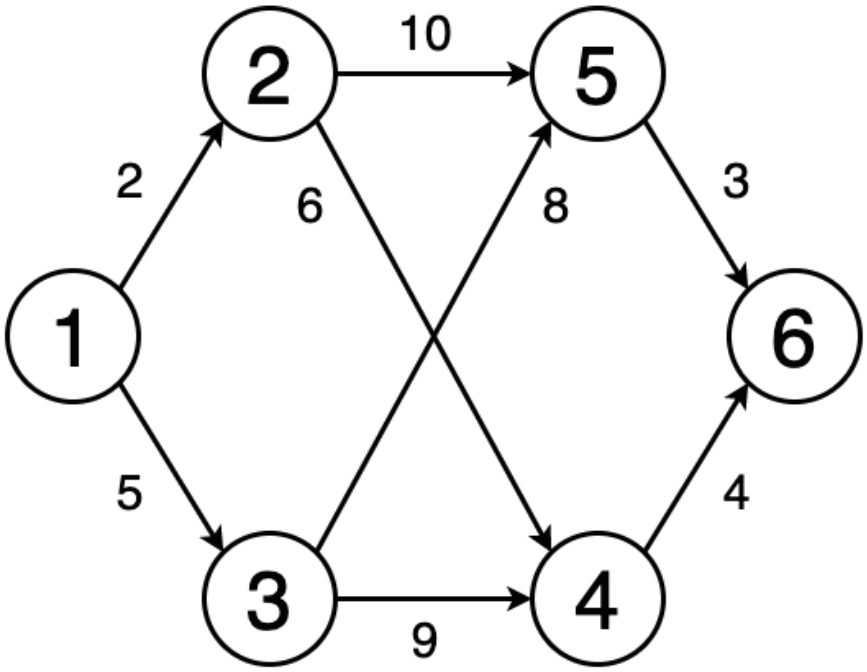


Fig 4. Animation showing the shortest path from vertex 1 to vertex 6 (Image by Author)

The ***shortest path*** from one vertex to another vertex is a path in the graph such that the sum of the weights of the edges that should be travelled is minimum.

Figure 4 shows an animation where the shortest path is determined from vertex 1 to vertex 6 in a graph.

Algorithms

1. Dijkstra's shortest path algorithm
2. Bellman–Ford algorithm

Applications

- Used to find directions to travel from one location to another in mapping software like Google maps or Apple maps.
- Used in networking to solve the min-delay path problem.
- Used in abstract machines to determine the choices to reach a certain goal state via transitioning among different states (e.g., can be used to determine the minimum possible number of moves to win a game).
-

4. Cycle detection

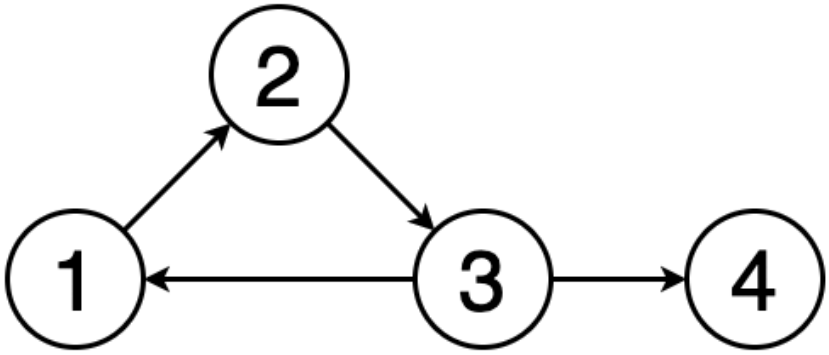


Fig 5. A cycle (Image by Author)

A *cycle* is a path in a graph where the first and last vertices are the same. If we start from one vertex, travel along a path and end up at the starting vertex, then this path is a cycle. ***Cycle detection*** is the process of detecting these cycles. Figure 5 shows an animation of traversing a cycle.

Algorithms

1. Floyd cycle detection algorithm
2. Brent's algorithm

Applications

- Used in distributed message-based algorithms.
- Used to process large-scale graphs using a distributed processing system on a cluster.

- Used to detect deadlocks in concurrent systems.
- Used in cryptographic applications to determine keys of a message that can map that message to the same encrypted value.

5. Minimum spanning tree

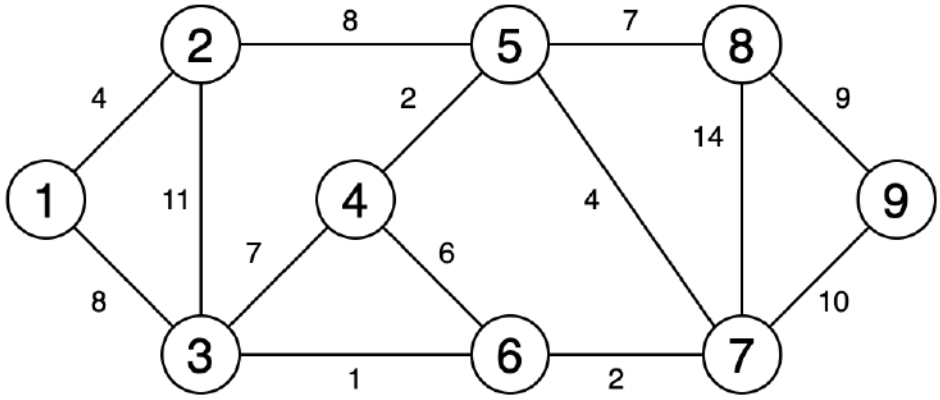


Fig 6. Animation showing a minimum spanning tree (Image by Author)

A ***minimum spanning tree*** is a subset of the edges of a graph that connects all the vertices with the minimum sum of edge weights and consists of no cycles.

Figure 6 is an animation showing the process of obtaining a minimum spanning tree.

Algorithms

1. Prim's algorithm
2. Kruskal's algorithm

Applications

- Used to construct trees for broadcasting in computer networks.
- Used in graph-based cluster analysis.
- Used in image segmentation.
- Used in regionalisation of socio-geographic areas, where regions are grouped into contiguous regions.

6. Strongly connected components

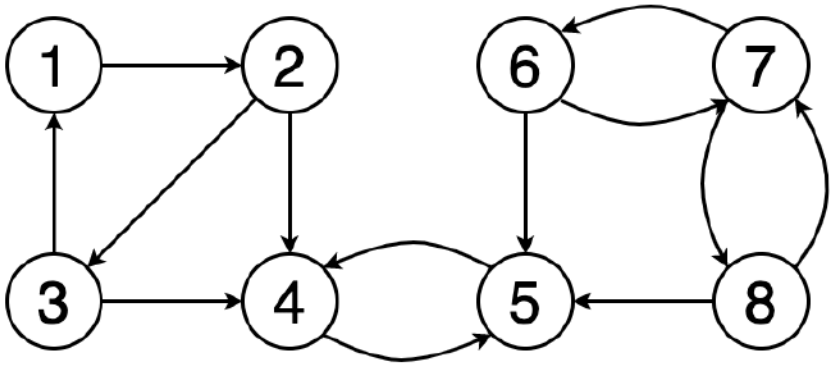


Fig 7. Strongly connected components (Image by Author)

A graph is said to be ***strongly connected*** if every vertex in the graph is reachable from every other vertex.

Figure 7 shows an example graph with three strongly connected components with vertices coloured in red, green and yellow.

Algorithms

1. Kosaraju's algorithm
2. Tarjan's strongly connected components algorithm

Applications

- Used to compute the Dulmage–Mendelsohn decomposition, which is a classification of the edges of a bipartite graph.
- Used in social networks to find a group of people who are strongly connected and make recommendations based on common interests.

7. Topological sorting

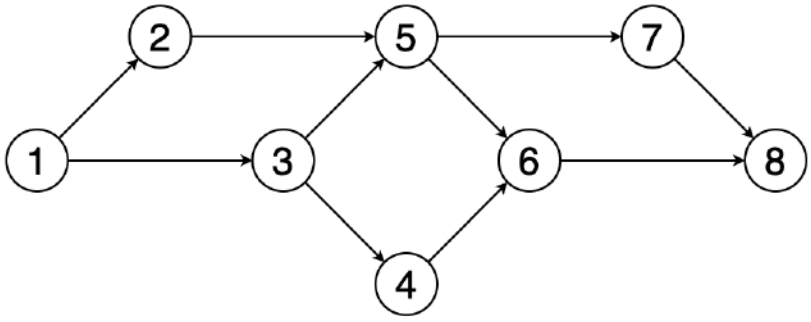


Fig 8. A topological ordering of vertices in a graph (Image by Author)

Topological sorting of a graph is a linear ordering of its vertices so that for each directed edge (u, v) in the ordering, vertex u comes before v.

Figure 8 shows an example of a topological ordering of vertices (1, 2, 3, 5, 4, 6, 7, 8). You can see that vertex 5 should come after vertices 2 and 3. Similarly, vertex 6 should come after vertices 4 and 5.

Algorithms

1. Kahn's algorithm
2. The algorithm based on depth-first search

Applications

- Used in instruction scheduling.
- Used in data serialisation.
- Used to determine the order of compilation tasks to perform in makefiles.
- Used to resolve symbol dependencies in linkers.

8. Graph colouring

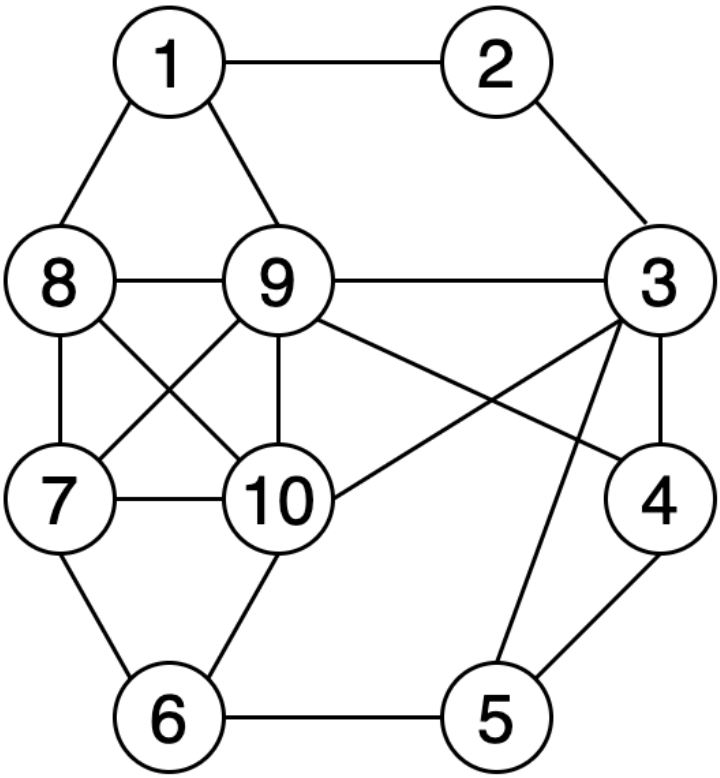


Fig 9. Vertex colouring (Image by Author)

Graph colouring assigns colours to elements of a graph while ensuring certain conditions. *Vertex colouring* is the most commonly used graph colouring technique. In vertex colouring, we try to colour the vertices of a graph using *k* colours and any two adjacent vertices should not have the same colour. Other colouring techniques include *edge colouring* and *face colouring*.

The *chromatic number* of a graph is the smallest number of colours needed to colour the graph.

Figure 9 shows the vertex colouring of an example graph using 4 colours.

Algorithms

1. Algorithms using breadth-first search or depth-first search
2. Greedy colouring

Applications

- Used to schedule timetable.
- Used to assign mobile radio frequencies.
- Used to model and solve games such as Sudoku.
- Used to check if a graph is bipartite.
- Used to colour geographical maps of countries or states where adjacent countries or states have different colours.

9. Maximum flow

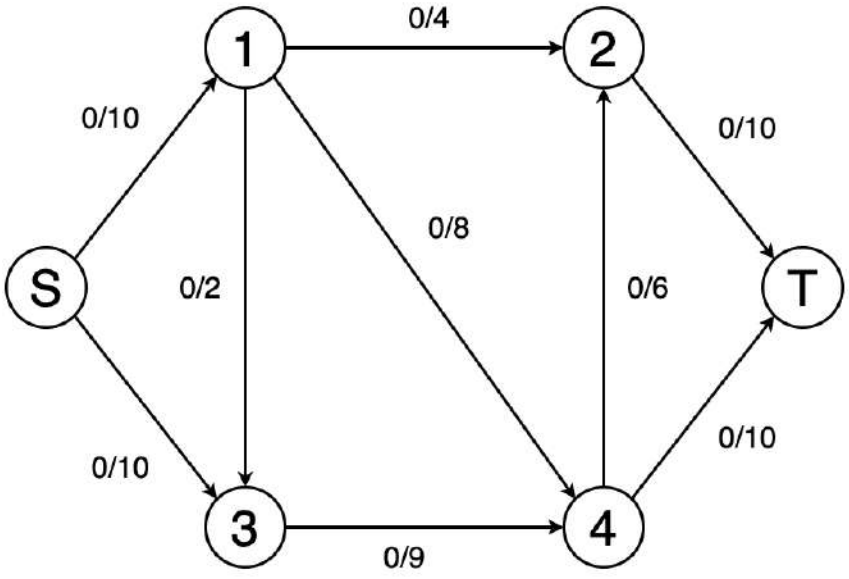


Fig 10. Determining the maximum flow (Image by Author)

We can model a graph as a flow network with edge weights as flow capacities. In the ***maximum flow*** problem, we have to find a flow path that can obtain the maximum possible flow rate.

Figure 10 shows an animated example of determining the maximum flow of a network and determining the final flow value.

Algorithms

1. Ford-Fulkerson algorithm
2. Edmonds–Karp algorithm
3. Dinic's algorithm

Applications

- Used in airline scheduling to schedule flight crews.
- Used in image segmentation to find the background and the foreground in an image.
- Used to eliminate baseball teams that cannot win enough games to catch up to the current leader in their division.

Source: https://towardsdatascience.com/10-graph-algorithms-visually-explained-e57faa1336f3