Back to Explore

Favorite

Detailed Explanation of

# Graph



## Introduction

`Graph` is probably the data structure that has the closest resemblance to our daily life. There are many types of graphs describing the relationships in real life. For instance, our friend circle is a huge "graph".

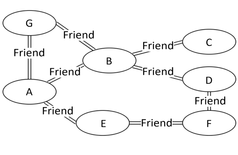


Figure 1. An example of a undirected graph.

In Figure 1 above, we can see that person G, B, and E are all direct friends of A, while person C, D, and F are indirect friends of A. This example is a social graph of friendship. So, what is the "graph" data structure?

## Types of "graphs"

There are many types of "graphs". In this Explore Card, we will introduce three types of graphs: **undirected graphs, directed graphs, and weighted graphs.**

### Undirected graphs

The edges between any two vertices in an "undirected graph" do not have a direction, indicating a two-way relationship.

Figure 1 is an example of an undirected graph.

### Directed graphs

The edges between any two vertices in a "directed graph" graph are directional.

Figure 2 is an example of a directed graph.

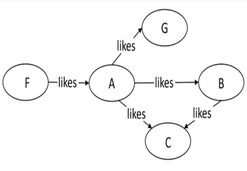


Figure 2. An example of a directed graph.

### Weighted graphs

Each edge in a "weighted graph" has an associated weight. The weight can be of any metric, such as time, distance, size, etc. The most commonly seen "weighted map" in our daily life might be a city map. In Figure 3, each edge is marked with the distance, which can be regarded as the weight of that edge.

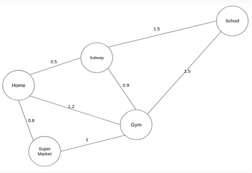


Figure 3. An example of a weighted graph.

## The Definition of "graph" and Terminologies

"Graph" is a non-linear data structure consisting of vertices and edges. There are a lot of terminologies to describe a graph. If you encounter an unfamiliar term in the following Explore Card, you may look up the definition below.

- Vertex: In Figure 1, nodes such as A, B, and C are called vertices of the graph.
- Edge: The connection between two vertices are the edges of the graph. In Figure 1, the connection between person A and B is an edge of the graph.
- Path: the sequence of vertices to go through from one vertex to another. In Figure 1, a path from A to C is [A, B, C], or [A, G, B, C], or [A, E, F, D, B, C].

**Note**: there can be multiple paths between two vertices.

- Path Length: the number of edges in a path. In Figure 1, the path lengths from person A to C are 2, 3, and 5, respectively.
- Cycle: a path where the starting point and endpoint are the same vertex. In Figure 1, [A, B, D, F, E] forms a cycle. Similarly, [A, G, B] forms another cycle.
- Negative Weight Cycle: In a "weighted graph", if the sum of the weights of all edges of a cycle is a negative value, it is a negative weight cycle. In Figure 4, the sum of weights is -3.
- Connectivity: if there exists at least one path between two vertices, these two vertices are connected. In Figure 1, A and C are connected because there is at least one path connecting them.
- Degree of a Vertex: the term "degree" applies to unweighted graphs. The degree of a vertex is the number of edges connecting the vertex. In Figure 1, the degree of vertex A is 3 because three edges are connecting it.
- In-Degree: "in-degree" is a concept in directed graphs. If the in-degree of a vertex is d, there are d directional edges incident to the vertex. In Figure 2, A's indegree is 1, i.e., the edge from F to A.
- Out-Degree: "out-degree" is a concept in directed graphs. If the out-degree of a vertex is d, there are d edges incident from the vertex. In Figure 2, A's outdegree is 3, i,e, the edges A to B, A to C, and A to G.

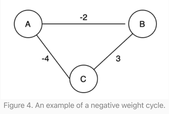


Figure 4. An example of a negative weight cycle.

After understanding the basics of "graph", let's start our journey on learning data structures and algorithms related to "graph".

### Disjoint Set

- Overview of Disjoint Set
- Quick Find - Disjoint Set
- Quick Union - Disjoint Set
- Union by Rank - Disjoint Set
- Path Compression Optimizatio...
- Optimized "disjoint set" with P...
- Summary of the "disjoint set" d...
- Number of Provinces
- LeetCode 547 - Number of Pro...
- Graph Valid Tree
- Number of Connected Compon...
- The Earliest Moment When Eve...
- Smallest String With Swaps
- Evaluate Division
- Optimize Water Distribution in ...

### The Depth First Search Algorithm in Graph

- Overview of Depth-First Searc...
- Traversing all Vertices – Depth...
- Traversing all paths between t...
- Find if Path Exists in Graph
- LeetCode 1971 - Find if Path E...
- All Paths From Source to Target
- LeetCode 797 - All Paths From ...
- Clone Graph
- Reconstruct Itinerary
- All Paths from Source Lead to ...

### The Breadth First Search Algorithm in Graph

- Overview of Breadth-First Sear...
- Traversing all Vertices - Breadt...
- Shortest Path Between Two Ve...
- Find if Path Exists in Graph
- LeetCode 1971 - Find if Path E...
- All Paths From Source to Target
- LeetCode 797 - All Paths From ...
- Populating Next Right Pointers ...
- Shortest Path in Binary Matrix
- N-ary Tree Level Order Traversal
- Rotting Oranges

### Algorithms to Construct Minimum Spanning Tree

- Overview of Minimum Spannin...
- Cut Property
- Kruskal's Algorithm
- Min Cost to Connect All Points
- LeetCode 1584 - Min Cost to C...
- Prim's Algorithm
- Min Cost to Connect All Points
- LeetCode 1584 - Min Cost to C...

### Single Source Shortest Path Algorithm

- Overview of Single Source Sho...
- Dijkstra's Algorithm
- Network Delay Time
- Bellman Ford Algorithm
- Improved Bellman-Ford Algorit...
- Cheapest Flights Within K Stops
- LeetCode 787 - Cheapest Fligh...
- Path With Minimum Effort

### Kahn's Algorithm for Topological Sorting

- Overview of Kahn's Algorithm
- Course Schedule II
- LeetCode 210 - Course Sched...
- Alien Dictionary
- Minimum Height Trees
- Parallel Courses