Implementation graph

A **graph** is a non-linear data structure that consists of nodes and edges. Formally, a graph can be defined as an ordered set G(V, E) where V represents the vertices and E represents the set of edges that are used to connect these vertices.

**Graphs may be directed or undirected graphs**. A directed graph is a set of vertices/nodes connected by edges, with each node having a direction associated with it. Edges are represented by arrows pointing to the direction in which the graph can be traversed. If there is an edge between vertex 0 and 1 with an arrow pointing towards 1, then the graph can be [traversed](https://www.codingninjas.com/blog/2020/09/25/graph-traversal-techniques-in-dfs-bfs/) (also see [Traversal](https://www.codingninjas.com/studio/library/traversal)) from 0 to 1 and not from 1 to 0. In contrast, an undirected graph can be traversed in either direction. The edges are bidirectional. If there is an edge between vertex 0 and 1, then the graph can be traversed from 0 to 1 and from 1 to 0.

Some graphs have weights associated with edges between two vertices these graph called wighted graph.

Implementation of the Graph can be done by using either an **adjacency list** or an **adjacency matrix**. Each of the two representations has its pros and cons; the choice of a particular graph representation depends on the requirements.

**Advantages of Adjacency Matrix Representation**:

1. Insertion and deletion of an edge can be done in O(1) time
2. We can easily determine if two edges are adjacent to each other in constant time complexity

Disadvantages of Adjacency Matrix Representation:

1. The memory consumption of adjacency matrix representation is of the order of O(V^2), where V is the number of vertices in the graph, which is way too high as compared to Adjacency Lists
2. Traversal of a graph requires O(V^2) time complexity

A graph can also be represented in the form of an adjacency list. An adjacency list represents a graph as an array of linked lists, wherein the index of the array represents a vertex, and each element of the linked list represents other vertices that form an edge with the vertex.

### Advantages of Adjacency List representation:

1. The representation is quite easy to follow and implement. It clearly shows the adjacent nodes of a particular node**.**
2. For graphs with a small-to-moderate number of edges, the Adjacency list consumes less memory of the order of [O(V + E), where V is the number of vertices and E is the number of edges] as compared to the Adjacency matrix. Therefore it is a preferred choice for a graph with a small-to-moderate number of nodes**.**

### Disadvantages of Adjacency List representation:

1. It takes linear time to determine whether there is an edge from vertex ‘i’ to ‘j’ as compared to the constant time in the case of the adjacency matrix.
2. Also, in the case of graphs with a high number of vertices and edges, the memory consumption O(V^2) is comparable to the Adjacency Matrix. So the adjacency matrix is a preferred choice in such cases.

### Traversal of a graph means visiting all the vertices and edges of the graph. Graph traversal can be done in one of the following ways:

1. Breadth-First Search

### Depth First Search