Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

**Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.**

**Graphs are also used in social networks like linkedIn, facebook. For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale. See this for more applications of graph.**

**Representations:**

**Adjacency Matrix Representation:**

**adjacencymatrix**

Now, this is adjacency list representation.

**Adjacency List:**listadjacency

**Adjacency Matrix Representation:  
  
Pros:** Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O(1).

**Cons:** Consumes more space O(V2). Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is O(V2) time.

**Adjacency List Representation:**

**Pros:** Saves space O(|V|+|E|) . In the worst case, there can be C(V, 2) number of edges in a graph thus consuming O(V2) space. Adding a vertex is easier.

**Cons:** Queries like whether there is an edge from vertex u to vertex v are not efficient and can be done O(V).