**Adjacency List**

An adjacency list represents a graph as an array of linked lists. The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.

For example, we have a graph below.

|  |
| --- |
| A graph |
| An undirected graph |

We can represent this graph in the form of a linked list on a computer as shown below.

|  |
| --- |
| Linked list representation of the graph |
| Linked list representation of the graph |

Here, **0**, **1**, **2**, **3** are the vertices and each of them forms a linked list with all of its adjacent vertices. For instance, vertex 1 has two adjacent vertices 0 and 2. Therefore, 1 is linked with 0 and 2 in the figure above.

## Pros of Adjacency List

* An adjacency list is efficient in terms of storage because we only need to store the values for the edges. For a sparse graph with millions of vertices and edges, this can mean a lot of saved space.
* It also helps to find all the vertices adjacent to a vertex easily.

## Cons of Adjacency List

* Finding the adjacent list is not quicker than the adjacency matrix because all the connected nodes must be first explored to find them.

## **Adjacency List Java**

We use Java Collections to store the Array of Linked Lists.

class Graph{

private int numVertices;

private LinkedList<integer> adjLists[];

}

The type of LinkedList is determined by what data you want to store in it. For a labeled graph, you could store a dictionary instead of an Integer.

## Applications of Adjacency List

* It is faster to use adjacency lists for graphs having less number of edges.