|  |  |  |
| --- | --- | --- |
| Comparing | Adjacency Matrix | Adjacency List |
| How Do You Do It? | 1. Create a matrix A of size NxN and initialise it with zero.  2. Iterate over each given edge of the form (u,v) and assign 1 to A[u][v]. Also, If graph is undirected then assign 1 to A[v][u]. | 1. Create an array A of size N and type of array must be list of vertices. Intially each list is empty so each array element is initialise with empty list.  2. Iterate each given edge of the form (u,v) and append v to the uth list of array A. Also, If graph is undirected append u to the vth list of array A. |
| Use Memory | Worst | Best |
| Usage | It is recommended that we should use adjacency matrix for representing Dense Graphs | It is recommended that we should use adjacency list for representing Sparse Graphs |
| Use space | O( N \* N) Worst | O(N + M) Best |
| Application | Adjacency matrix is used where information about each and every possible edge is required for the proper working of an algorithm like :- Floyd-Warshall Algorithm where shortest path from each vertex to each every other vertex is calculated (if it exists). It can also be used in DFS (Depth First Search) and BFS (Breadth First Search) but list is more efficient there. Sometimes it is also used in network flows. | Adjacency List is a space efficient method for graph representation and can replace adjacency matrix almost everywhere if algorithm doesn't require it explicitly. It is used in places like: BFS, DFS, Dijkstra's Algorithm etc. |

# Task 2 comparing between : Adjacency matrix and Adjacency list

Task 3:

Depth First Search: is an algorithm for searching a graph or tree data structure. The algorithm starts at the root (top) node of a tree and goes as far as it can down a given branch (path), then backtracks until it finds an unexplored path, and then explores it. The algorithm does this until the entire graph has been explored. Many problems in computer science can be thought of in terms of graphs. For example, analyzing networks, mapping routes, scheduling, and finding spanning trees are graph problems. To analyze these problems, graph-search algorithms like depth-first search are useful.

Breadth First Search: is a method for exploring a tree or graph. In a BFS, you first explore all the nodes one step away, then all the nodes two steps away, etc. Breadth-first search is like throwing a stone in the center of a pond. The nodes you explore ripple out from .