1. **Adjacency List:** An Adjacency list is an array consisting of the address of all the linkedlist. The first node of the linked list represents the vertex and the remaining lists connected to this node represents the vertices to which this node is connected. This representation can also be used to represent a weighted graph. The linked list can slightly be changed to even store the weight of the edge.
2. **Adjacency Matrix**: Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

Adjacency Matrix

- Uses O(n^2) memory
- It is fast to lookup and check for presence or absence of a specific edge
- between any two nodes O(1)
- It is slow to iterate over all edges
- It is slow to add/delete a node; a complex operation O(n^2)
- It is fast to add a new edge O(1)

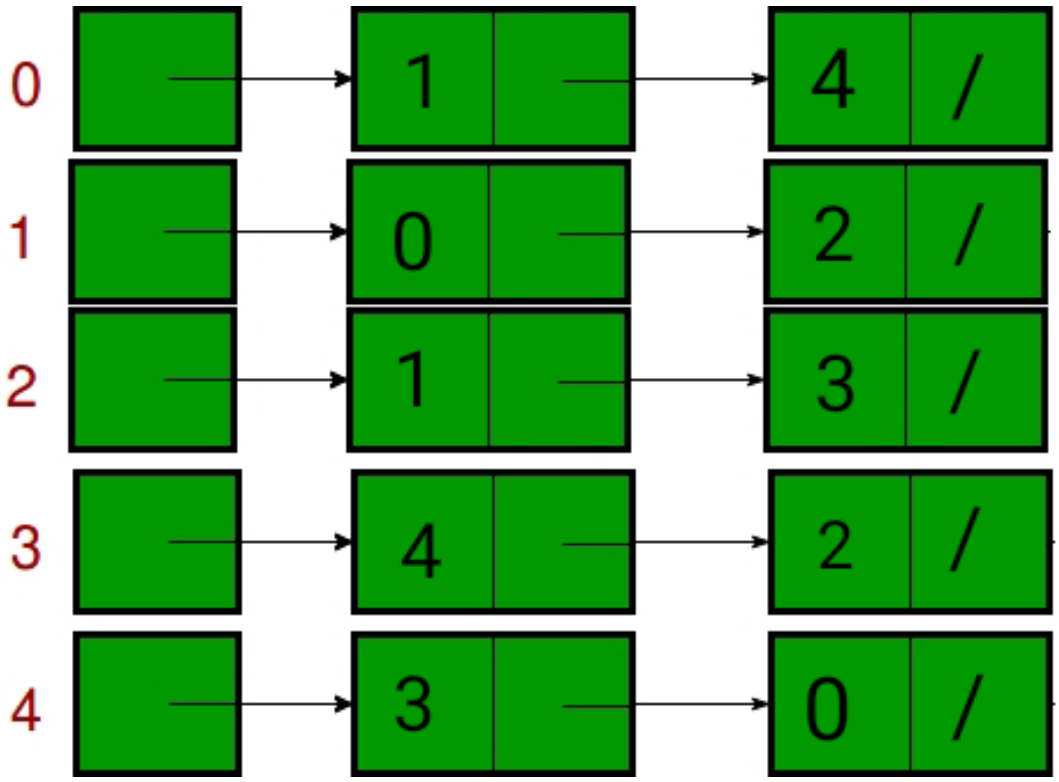
Adjacency List

- Memory usage depends more on the number of edges (and less on the number of nodes),
- which might save a lot of memory if the adjacency matrix is sparse
- Finding the presence or absence of specific edge between any two nodes
- is slightly slower than with the matrix O(k); where k is the number of neighbors nodes
- It is fast to iterate over all edges because you can access any node neighbors directly
- It is fast to add/delete a node; easier than the matrix representation
- It fast to add a new edge O(1)

Memory

- An adjacency matrix occupies $n^2/8$ byte space (one bit per entry).
- An adjacency list occupies 8e space, where e is the number of edges (32bit computer)

Adjacency List



Adjacency Matrix