Graph Representation - Adjacency List

\*In adjacency matrix representation we store details about which nodes are connected as well as which are not connected.

For a social network with a billion(10^9) users:

Row-size = 10^9

If I have 1000 friends:

No of 1s  = 1000 🡪1KB

No of 0s = 10^9 – 1000 🡪 1GB

So let’s see other ways of storing the connection details

Connection of node A

Instead of representing node A as an array such that index represents endpoint of an edge and value at that particular index represents whether we have an edge ending there or not. we can simply keep a list of all the nodes to which we are connected. This is the list or set of nodes that A is connected. We can represent this list either using indices or actual names for the node. Let’s just use indices because names can be long and may consume more memory. You can always look at the names of the node from vertex list using indices in constant time.

In a machine we can store these set of nodes which basically is a set of integers in something as simple as an array and this array as you can see is a different arrangement from our previous array . In our earlier arrangement index was representing index of one node in the graph and value was representing if there was a connection to that node or not. Here index doesn’t represent anything and values are the actual indices of the node to which we are connected.

Now instead of using an array here to store this set of integers we can also use a linked list and tree or binary search tree.

So there are a bunch of ways in which we can store connection of a node. We can do all insertion deletion in logn

Look at the image here



