Graph

A data Structure $G(V, E)$

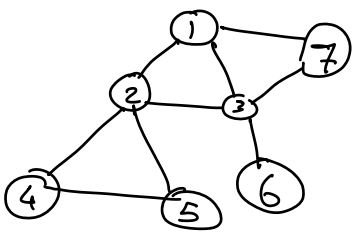
V: vertices

every edge $e \in E$ is in form (v_i, v_j) where $v_i, v_j \in V$

- directed
  - Twitter following
- Undirected
  - facebook friends

- weighted
  - edges have weights
  - map edges
- Unweighted
  - facebook friends

$G(V, E)$



$V = \{1, 2, \ldots 7\}$

$E = \{(1,2), \ldots, (3,7)\}$

degree of a node:

edges incident to the node

$deg(3) = 4$, $deg(6) = 1$

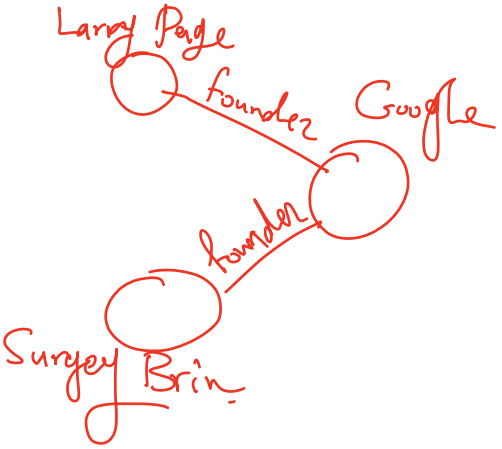
weighted Graphs

↳ edges / vertices have Lables / properties

Knowledge Graphs

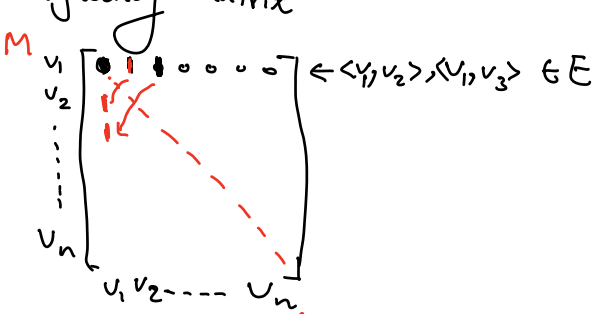
Vertices: objects

edges: relation b/w obj°



Graph Representation

Adjacency Matrix



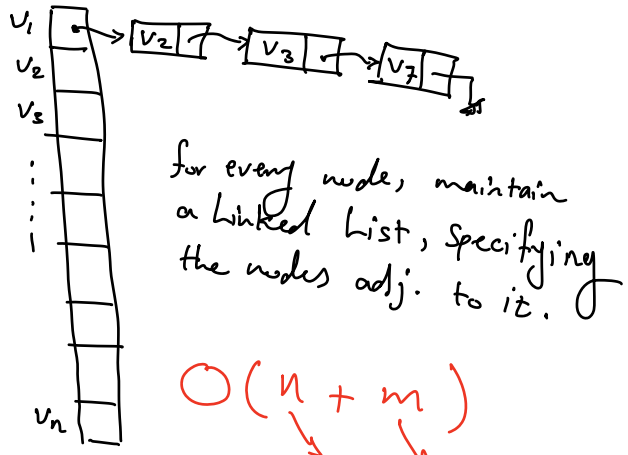
$$M[i,j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Observation: if the Graph is not directed ⇒ Storing the upper-Triangle of the Matrix is enough

Requires $O(n^2)$ Space

Checking the existance of an edge $O(1)$

Adj. List



for every node, maintain a Linked List, specifying the nodes adj. to it.

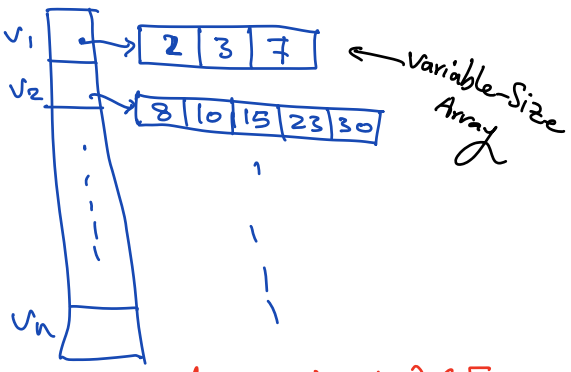
$O(n+m)$

- n: # vertices
- m: # edges

- It Saves Space
- $O(n)$ to check the existance of an edge
  - $O(\text{max degree of nodes})$

Access $O(\log n)$

Storage $O(n+m)$



does $(v_2, v_{17}) \in E$

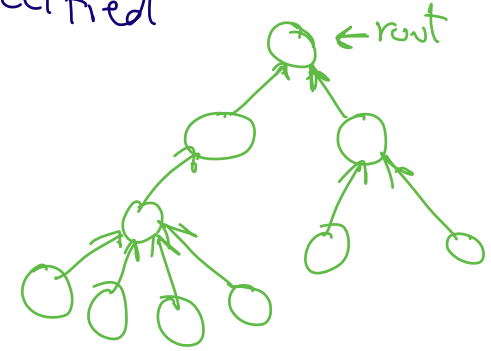
Path: a Sequence of vertices $\langle v_{i_1}, v_{i_2}, \dots, v_{i_k} \rangle$ where $\forall j \in [1, k-1]$, $\langle v_{i_j}, v_{i_{j+1}} \rangle \in E$

Simple Path: a path where no vertex is repeated more than once.

Cycle: a path where $v_{i_1} = v_{i_k}$

Tree: a graph with no Cycle

Rooted Tree: a Tree where a node is specified as root and from every node in the tree the direction to root is Specified



Connectivity in graph

- S-t Connectivity:
  if there exists a path from node s to node t
- S-t Shortest path:
  the path with min length from s to t
- Connected Component:
  a subgraph G', where $\exists v_i, v_j \in G'$, v_i is connected to v_j.

Graph Traversal

- Breadth first Search (BFS)
- Depth first Search (DFS)