Tuesday, March 19, 2024 10:52 AM

Graph Algorithms

Graph $G = (V, E)$

V: set of vertices

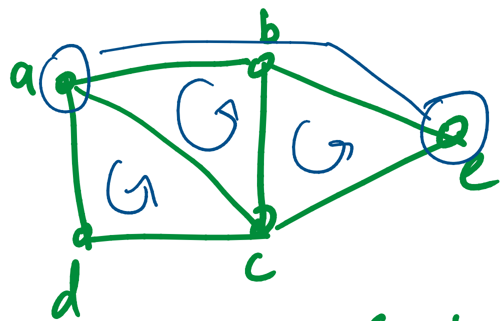
E: set of edges.

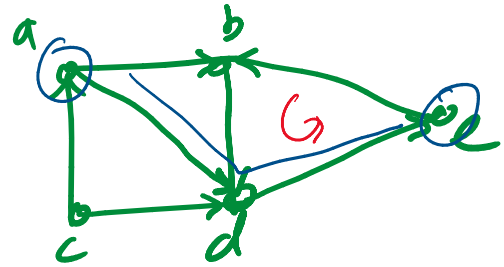
$|V| = n$

$|E| = m$

$(n-1) \leq m \leq n^2$

eg.





$V = \{a, b, c, d, e\}$

$E = \{(a,b), (c,a), (b,d), (a,d), (d,e), (e,b), (c,d)\}$

$E = \{ab, ac, ad, bc, dc, ce, be\}$

$ab \to \{a, b\}$

Appl'n: facebook graph, road n/w, internet

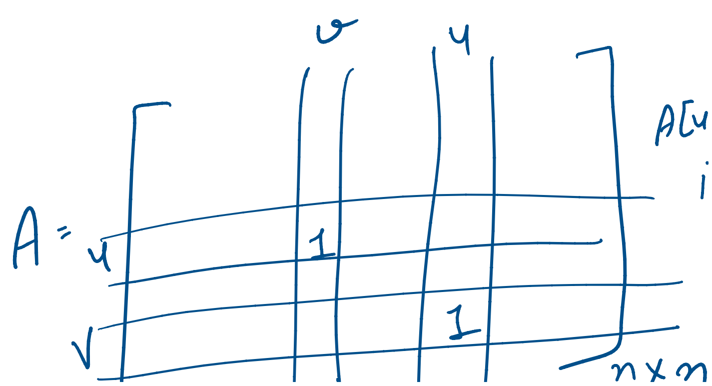
☆ Basic Concept: path, connected, cycles.

☆ Representation:

- Adjacency Matrix

(if G undirected

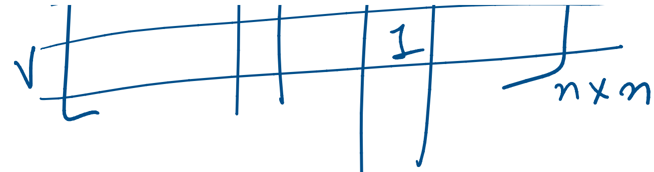
tric matrix)



$A[u,v] = 1$ if $(u,v) \in E$

$= 0$ o.w.

(if G undirected
A is symmetric matrix)



space $= O(n^2)$ Look up time $O(1)$.

- Adjacency list

$Adj(u) = \{v \mid (u,v) \in E\}$

a: b, d
b: d
c: d, a
d
e
⋮

Look up time $O(n)$.

$$\text{space} : O\left(\sum_{u \in V} |Adj(u)|\right)$$

$$= O(m+n)$$

(if sparse graph $m << n^2$)

☆ Basic que:

- if ∃ a path from s to t, or betⁿ s & t
- if G is connected?
- Vertices reachable from s?

⋮

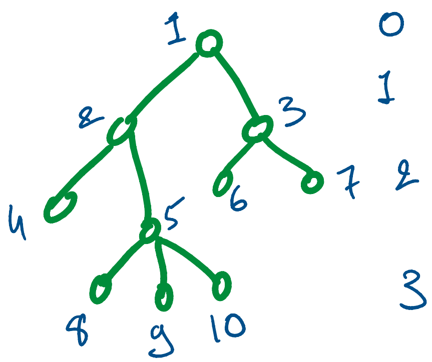
☆ Basic Search Algo:

Breadth First search (BFS)

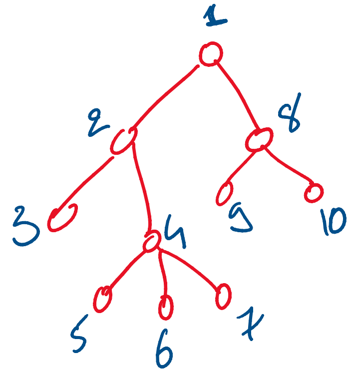
Depth First search (DFS).

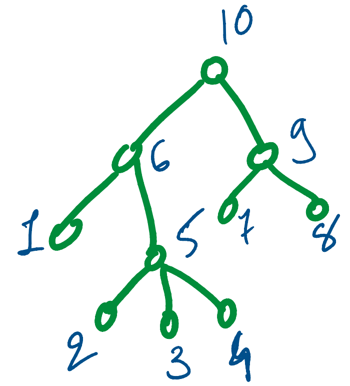
Eg. Tree

BFS



DFS

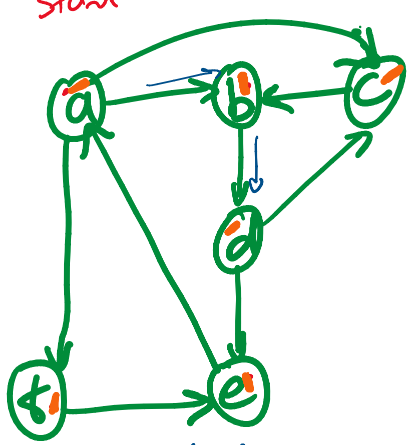




Discover order.
= Preorder traversal

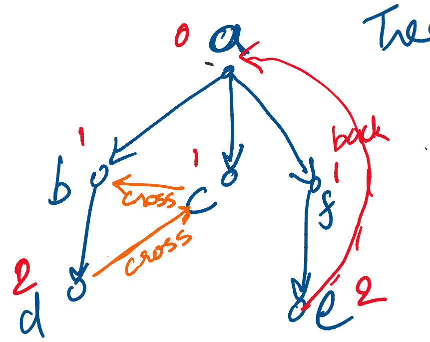
Finish order
= post order.

★ Graph (Extension).



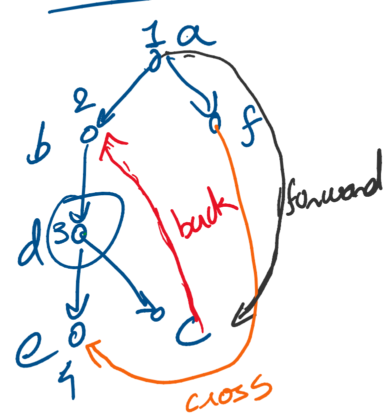
Q: ~~a b c f d e~~

BFS



Tree-edges

DFS



No forward edges
in BFS tree.

★ Non-tree edges:

- Back edges: edge from a node to one of it's ancestors.
- Forward edges: edge " " " " descendents.
- Cross edges: all other non-tree edges.

★ Implementation: BFS(G, s)

// idea 1: Mark visited vertices.

// idea 2: Use a data structure Q.
queue

$O(n) \leftarrow$ 1) for $u \in V$, do unmark u.

2) Insert s in Q. Mark s. level$[s] = 0$

3). while $Q \neq \emptyset$ {

4) remove a head vertex u from Q.

5) for each $v \in \underline{Adj(u)}$ do {

6) if v is unmarked.

7) insert v in Q at the end. Mark v.

parent$[v] = u$. level$[v]$ = level$[u]+1$

Run time: steps 5-7 $O(|Adj(u)|)$

Total time $O\left(\sum_{u \in V} |Adj(u)|\right) + O(n)$

$= O(m+n)$

global time = 1

★ DFS(G, u) {

// similar, with different data structure. stack

or Recursion.

discovered$[u]$ = time ++.

ok ...

1) Mark u. discovered [u] = time ++.
2) for $v \in Adj(u)$ do {
3) if v is unmarked.
4) DFS (G, v)
Parent [v] = u
5) }
6) Finished [u] = time ++

Q1: Shortest path distance from s to t.