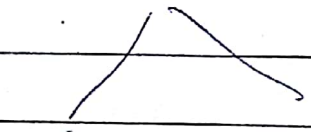
28/3/18

GRAPHS

$G(V, E)$

→ Complete graph : direct edge b/w any 2 vertices :



directed — undirected

edges: $n(n-1)$ — ${}^{n}C_2 = \frac{n(n-1)}{2}$

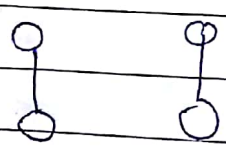
→ If (v_0, v_1) is an edge in an undirected graph
- v_0 & v_1 are adjacent
- The edge (v_0, v_1) is incident on v_0 & v_1

→ self-edge : self-loops
multigraph : more than 1 edges b/w a pair of vertices

→ Sub-graph : any subset of graph

→ Simple path : all ~~path~~ vertices, except possibly 1st & last are distinct.

→ Unconnected :



→ Connected component of an undirected graph is a maximal connected subgraph

→ Degree (vertex) : no. of edges incident on that vertex

↳ Directed

in-degree — v → head

out-degree — v → tail

Teacher's Signature

Graph Representation :

1) Adjacency Matrix :

if no self loop ⇒ all diagonal elements = 0

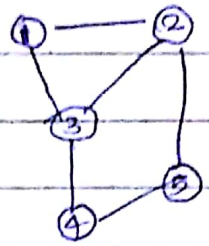
Undirected ⇒ $m[i][j] = m[j][i]$

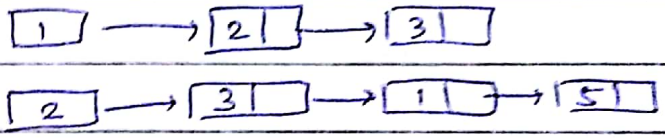
complexity : initialize with 0 ⟹ n^2

& fill info. from set of edges & vertices

⇒ $O(n^2)$

2) Adjacency list :





$O(2|E|) + O(|V|)$ $\quad |E| = 6$, $|V| = 5$

→ whether to choose 1) or 2) : depends on #V & #E

↳ if $|V|^2 >> |E|$ ⇒ use 2) → tough to implement

Abstract Data Type

1) Adj - Mtr - G

Insert node ⇒ add 1 col & 1 row

Adj-Mtr-G' = $\begin{bmatrix} \text{Adj\_Mtr\_G} & \begin{matrix}0\\0\\0\end{matrix} \\ \begin{matrix}0&0&0&0&0\end{matrix} & 0 \end{bmatrix}$

Teacher's Signature

Breadth First Search

$d[v]$: shortest distance from source S to vertex v.

$\pi[v]$: vertex which is predecessor to v if shortest path is followed

→ For each vertex : we use one variable : color

White : not discovered vertex

Gray : vertex is discovered

Black : finished
if all vertices adjacent to it have been discovered.

→ Also require queue

```
BFS(G, s)
  for each vertex u in V[G] - {s}
      do color[u] ← white          } initialization
         d[u] ← ∞
  π[u] ← nil
  color[s] ← gray                  (discovered source)
  d[s] ← 0
  π[s] ← nil
  Q ← ∅                            } Moment a vertex is discovered, put it into queue
  enqueue(Q, s)
  while Q ≠ ∅
      do u ← dequeue(Q)
         for each v in Adj[u]
             do if color[v] = white
                then color[v] ← gray
```

· for each — —
do —

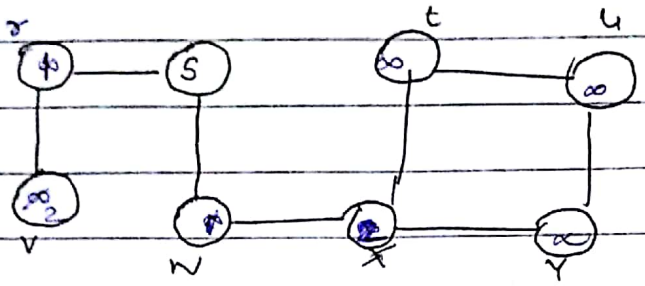
$d[v] \leftarrow d[u] + 1$

$\pi[v] \leftarrow u$

enqueue (Q, v)

color[u] ← black ;

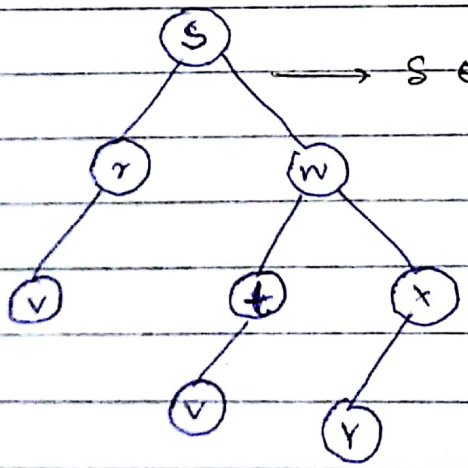
Ex



Q : s w r x y t y u

0 1 1 2 2 3 3 4

BFS :



level 0 → S energy is required to pass an edge (let)

level 1

level 2

level 3 — node farthest from source is at last level

→ If give energy = 3 $\delta \epsilon$ ⇒ can reach to each node
(Max energy required = 3 $\delta \epsilon$)

App^n of BFS

→ diameter of graph (D)

$S(u,v)$ ← shortest path distance from u to v.

$$D = \max\left(S(u,v)\right)$$

Here, $D = 5$

How to find D?

For 1 s → we can get most distant (at last level)

- Do same for each source & find max among all.

In last of code write :

return max [d]

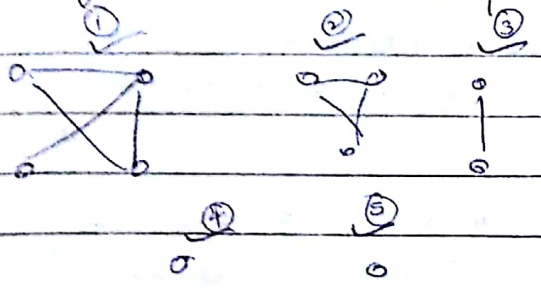
for all $v \in V$

$D[v] \leftarrow BFS(v, G)$

Diameter = max [D]

DATE : / /
PAGE NO. :

→ to find connected component



No of connected components here = 5

validity : check whether vertex is discovered or not

Choose any vertex whose color is still white → run on all vertices

can have counter how many times BFS is running

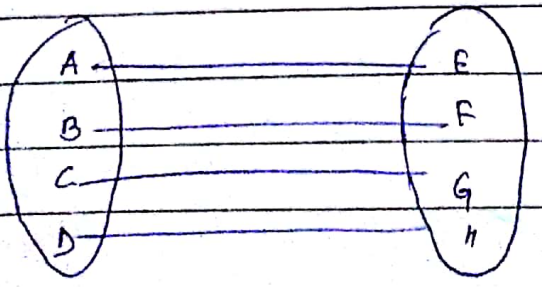
value of counter = no. of connected components

Keep on doing till all nodes are black.

Logic :

```
for v in V[G]
    if ( color[v] == white)
        BFS (v, G)
        counter ++ ;
```

→ to check whether a graph is bi partite or not ?



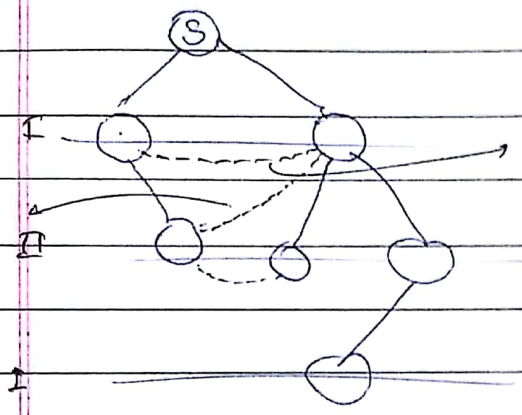
end of edges are divided into 2 categories

⇓

bipartite

(if edge b/w A & B ⇒ not bi partite)

Teacher's Signature.....



length of cycle → 4

if ∃ an edge b/w 2 vertices at same level ⇒ cycle

length = 3 (odd)

if vertices at diff. level → may / no cycle : len : even

cycle len : odd ⇒ not bipartite

cycle len : even ⇒ bipartite