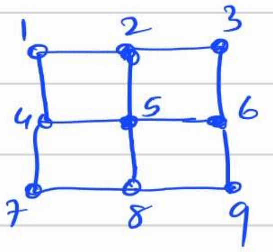
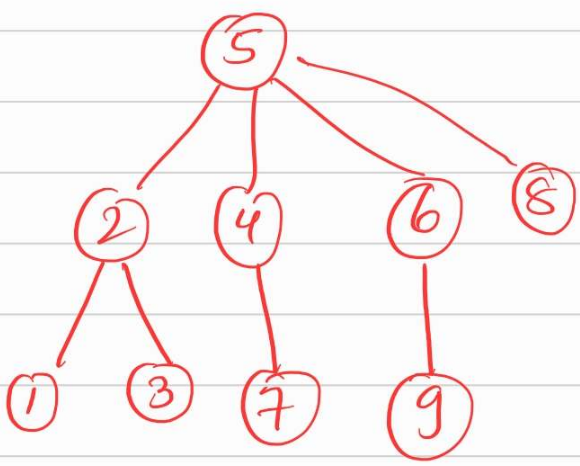
02/03/2022

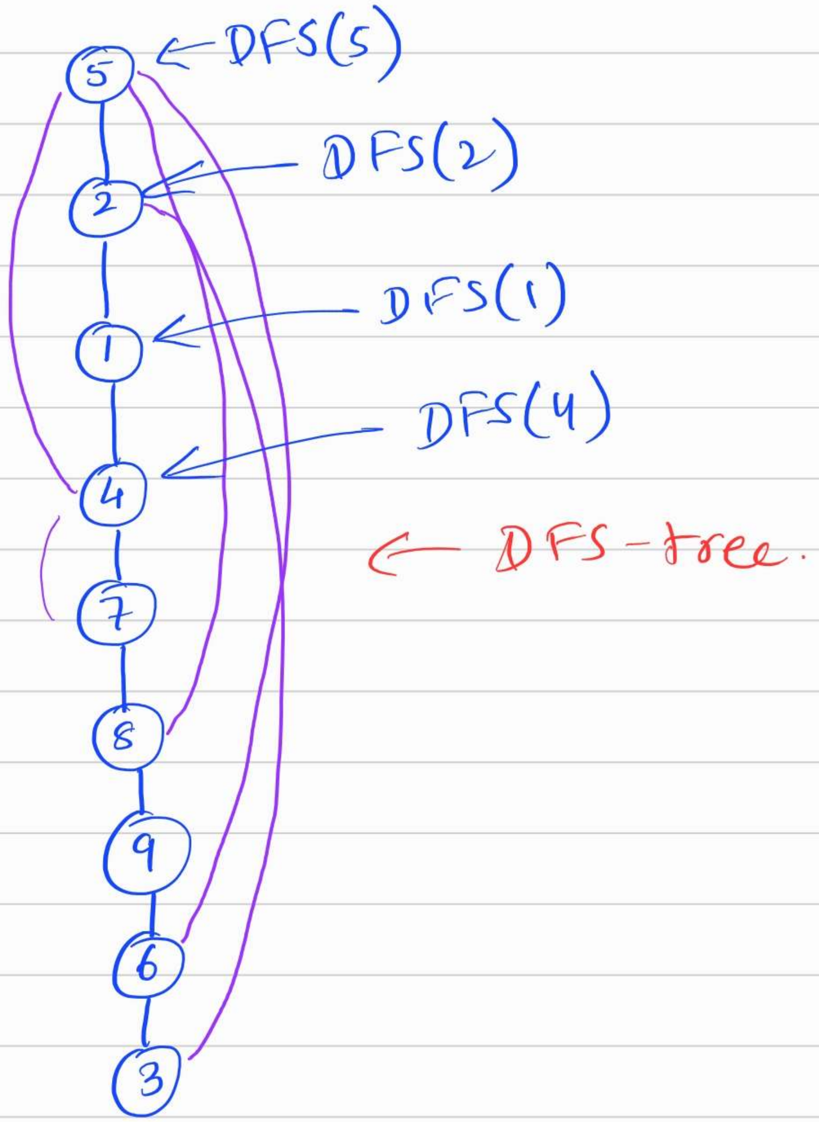
DFS (s)

Mark S as "seen"
For each edge (s, v) incident on s
  if v is not marked "seen" then
    Recursively call DFS(v)
  End if
End for.





BFS - tree



Fact:- Let T be a DFS - tree and (x, y) be an edge in G that doesn't belong to T. Then either x is an ancestor of y.

or y is an ancestor of x.

Proof:- WLOG, DFS(x) was executed before DFS(y).

At the invocation of DFS(x) y is not marked as "seen".

Also DFS(y) is not called just after DFS(x).

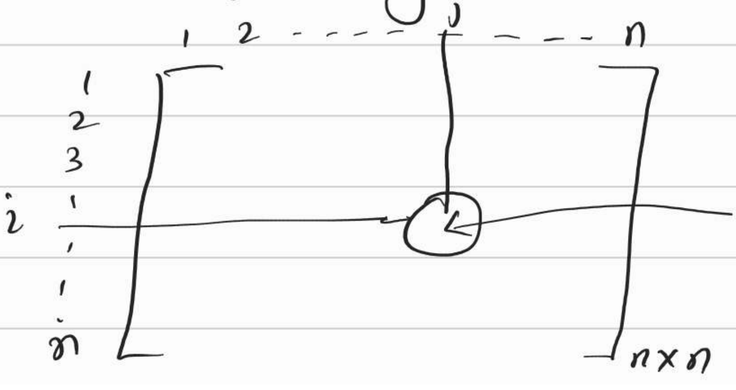
Then (x, y) will be considered after you invoke DFS(y).

Furthermore this invocation is before the end of the recursive call DFS(x). ∎

Runtime:

Representing a graph:

(1) Adjacency Matrix (2) Adjacency list.



this entry (i,j) is 0 or 1 depending on whether (i,j) is present or not

Adjanceny matrix is implemented as an Array.

Q: Is Adjacency matrix a symmetric matrix? Yes.

Q: How much space does it take?

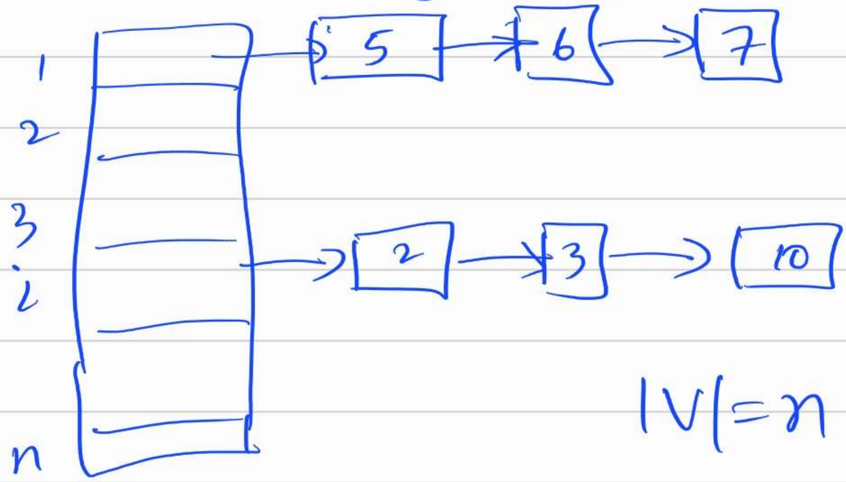
$O(n^2)$

Q: How much time does it take to figure out if an edge (i, j) is present? $O(1)$ time.

Q: How much time does it take to find all neighbors of a given node i? $O(n)$ time.

2) Adjacency list.

→ Array of lists.



$|V| = n, \ |E| = m$

Q: How much space does it take?

each edge is represented twice.

$\langle i, j \rangle \quad 2m = O(m)$

Q: How much time does it take to figure out if an edge is present?

O(deg of that vertex)

degree of a vertex = # of neighbors of that vertex

Q: How much time does it take to find all neighbours of a given vertex?

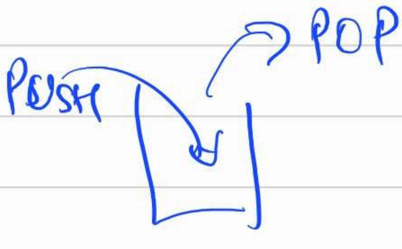
O(degree)

<u>Standard input representation of a graph</u>

:= Adjacency list.

Queue (FIFO) $\longrightarrow$ BFS

Stacks (LIFO) $\longrightarrow$ DFS.



Implementing BFS: $O(m+n)$ time.

Linear-time.

BFS(s):

Discovered(s) = True and Discovered(v) = False $\forall v \neq s$.

Intialize $L(0) = \{s\}$.

Set the layer count i to 0.

Set the BFS tree $T = \emptyset$.

While $L(i)$ is not empty.

    Intialize $L(i+1) = \emptyset$.

    For each node $u \in L(i)$

        Consider edge (u,v) incident on u.

        If Discovered(v) = False then

            Discovered(v) = True.

            Add v to $L(i+1)$

            Add (u,v) to T.

        End if

    End for.

    Increment the layer count i to $i+1$

End While.

<u>Fact</u> :- The above algorithm takes $O(m+n)$.

Implementing DFS:

Define Array Parent $[1, \ldots, n]$

DFS(s) :

Initialize $R = \emptyset$.

Intialize a Stack S with vertex s in it

While S is not empty.

Take a node u from S.

if Explored $[u]$ = false then

Set Explored (u) = True.

$R = R \cup \{u\}$

For each edge (u, v) incident on u

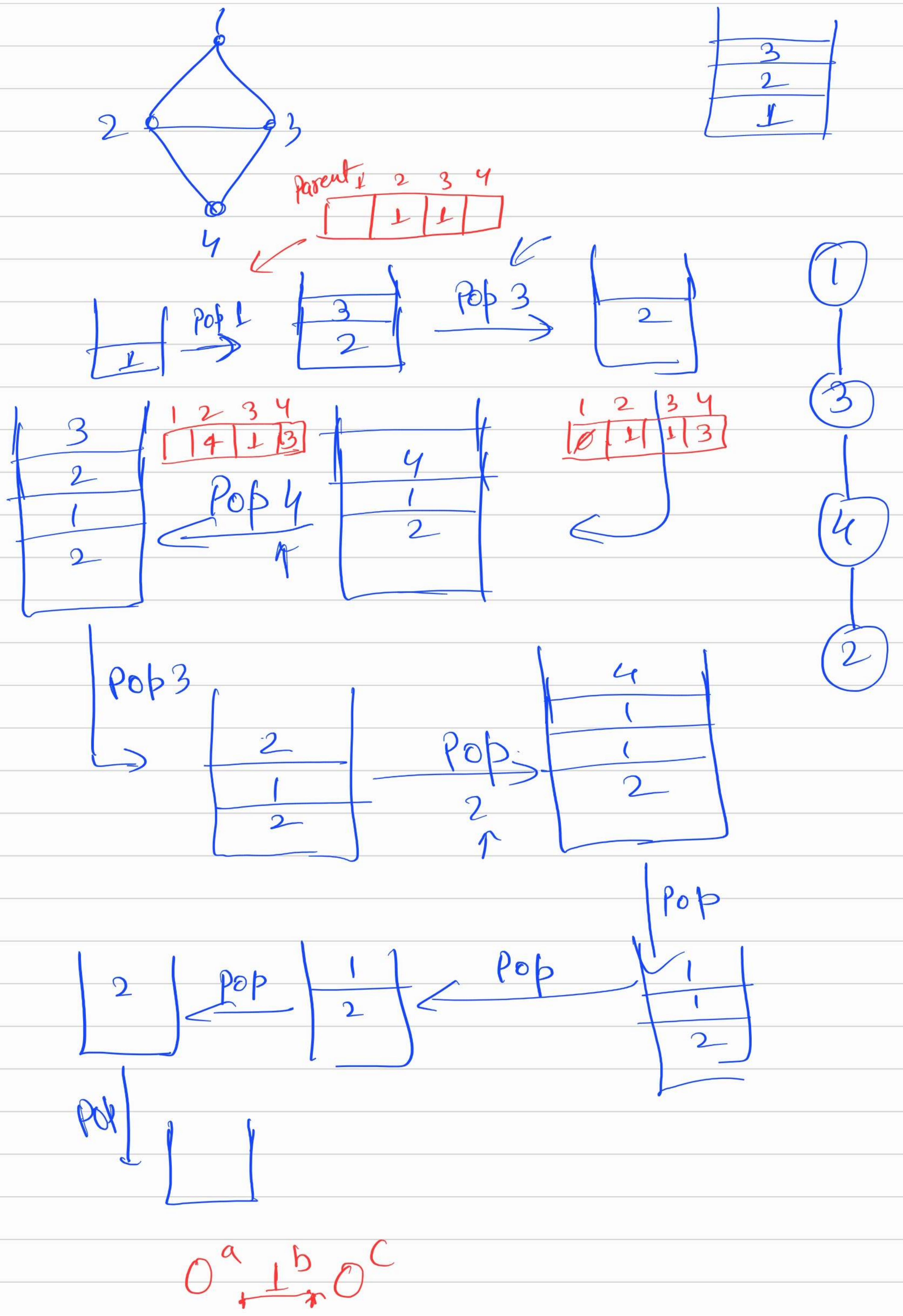
Add v to S ; Set Parent $[v] = u$ when $v \neq s$. and Explored $[v]$ = false.

End for

End if

End while.

Output R.


Parent
1 2 3 4
Pop 1
Pop 3
Pop 4
Pop 3
Pop
Pop
Pop
pop
Pop