AAD - Assignment - 1

NAGA MANOHAR
2021101128

(1) Given,

$$T_1(n) = a.T_1\left(\frac{n}{b}\right) + b.n$$

$$T_2(n) = b.T_2\left(\frac{n}{a}\right) + a.n$$

$a \geq b$ & $\boxed{T_1(1) = T_2(1) = 1}$ — (*)

we first find the expression for $T_1(n)$

$$T_1(n) = a.T_1\left(\frac{n}{b}\right) + b.n$$

$$\Rightarrow T_1\left(\frac{n}{b}\right) = a.T_1\left(\frac{n}{b^2}\right) + b.\left(\frac{n}{b}\right) \quad -(1)$$

$$T_1\left(\frac{n}{b^2}\right) = a.T_1\left(\frac{n}{b^3}\right) + b.\left(\frac{n}{b^2}\right) \quad -(2)$$

$$\Rightarrow T_1(n) = a^2.T_1\left(\frac{n}{b^2}\right) + bn\left(1+\frac{a}{b}\right) \quad (\because (1))$$

$$\Rightarrow T_1(n) = a^3.T_1\left(\frac{n}{b^3}\right) + bn\left(1+\frac{a}{b}+\frac{a^2}{b^2}\right) \quad (\because (2))$$

⋮ k times

$$\Rightarrow T_1(n) = a^k.T_1\left(\frac{n}{b^k}\right) + bn\left(1+\frac{a}{b}+\left(\frac{a}{b}\right)^2+\dots+\left(\frac{a}{b}\right)^k\right)$$

GP with $\begin{pmatrix} a = 1 \\ r = \frac{a}{b} \end{pmatrix}$ a, ar, a

$$= a^k.T_1\left(\frac{n}{b^k}\right) + bn.\left(\frac{\left(\frac{a}{b}\right)^k - 1}{\left(\frac{a}{b}\right) - 1}\right)$$

$$\frac{a(r^n - 1)}{r - 1}$$

$$\boxed{T_1(n) = a^k.T_1\left(\frac{n}{b^k}\right) + bn.\left(\frac{\left(\frac{a}{b}\right)^k - 1}{\left(\frac{a}{b}\right) - 1}\right)}$$

Now

let $\frac{n}{b^k} = 1$ as we know $T_1(1) = 1$

$\Rightarrow n = b^k$

$\Rightarrow \boxed{\log_b n = k}$

$\Rightarrow T_1(n) = a^{\log_b n} \cdot T_1\left(\frac{n}{n}\right) + bn \cdot \left(\frac{\left(\frac{a}{b}\right)^{\log_b n} - 1}{\left(\frac{a}{b}\right) - 1}\right)$

$$\boxed{T_1(n) = a^{\log_b n} + bn \cdot \left(\frac{\left(\frac{a}{b}\right)^{\log_b n} - 1}{\left(\frac{a}{b}\right) - 1}\right)}$$

Interchange a with b to get $T_2(n)$

$\Rightarrow T_2(n) = b^{\log_a n} + an \cdot \left(\frac{\left(\frac{b}{a}\right)^{\log_a n} - 1}{\left(\frac{b}{a}\right) - 1}\right)$

Now we ~~fir~~st prove that

$a^{\log_b n} \geq b^{\log_a n}$ $(\because a \geq b$ given$)$

$a \geq b$

$\Rightarrow \log a \geq \log b$ $\Rightarrow \frac{\log a}{\log b} \geq \frac{\log b}{\log a}$ $\left(\because a \geq \frac{1}{a}\right)$

$\Rightarrow \frac{\log a}{\log b} \geq 1$

$\Rightarrow \frac{\log a}{\log b} \cdot \log n \geq \frac{\log b}{\log a} \cdot \log n$

$\Rightarrow \log a \cdot \left(\frac{\log n}{\log b}\right) \geq \log b \left(\frac{\log n}{\log a}\right)$

$\Rightarrow \log a (\log_b n) \geq \log b \cdot \log_a n$

$(\because n \log a = \log a^n)$

$\Rightarrow (\log_b n) \cdot \log a \geq (\log_a n) \cdot \log b$

$\Rightarrow \log a^{\log_b n} \geq \log b^{\log_a n}$

$\Rightarrow \boxed{a^{\log_b n} \geq b^{\log_a n}}$ — (a)

Now we prove that

$$bn\left(\frac{\left(\frac{a}{b}\right)^{\log_b n}-1}{\left(\frac{a}{b}\right)-1}\right) \geq an\left(\frac{\left(\frac{b}{a}\right)^{\log_a n}-1}{\left(\frac{b}{a}\right)-1}\right)$$

Here $a \geq b \Rightarrow \boxed{\frac{a}{b} \geq 1} \Rightarrow \boxed{\frac{b}{a} \leq 1}$

Now $\frac{a}{b} \geq 1 \Rightarrow$ as $n \to \infty$ $(\log_b n \to \infty)$ GP sum tends to $\to \infty$ (diverges)

$\frac{b}{a} \leq 1 \Rightarrow$ as $n \to \infty$ $\log_a n \to \infty \Rightarrow$ GP sum $\to$ tends to some constant

Hence 1^{st} GP sum $\geq 2^{nd}$ GP sum as n grows large

Thus $\boxed{bn\left(\frac{\left(\frac{a}{b}\right)^{\log_b n}-1}{\left(\frac{a}{b}\right)-1}\right) \geq an\left(\frac{\left(\frac{b}{a}\right)^{\log_a n}-1}{\left(\frac{b}{a}\right)-1}\right)}$ — (C_2)

From (C_1) & (C_2)

$$\boxed{T_1(n) \geq T_2(n)}$$

Hence $\boxed{T_1(n) \geq T_2(n)}$ As n grows larger $(n \to \infty)$

Q2 Detect a Cycle In an Undirected Graph:-

we can do this by DFS and we need:-

- Adjacency List / Matrix (to store edges)
- is_visited array - make note of nodes that are visited

~~Part~~

1. We First Run a loop on each node in the adjacency list and

   Check if the node is_visited then continue

   else if the node is NOT visited the call DFS (take that node as a parameter)

2. In the DFS Function -

   - First make that node visited

Here let node => at

i => neighbours of node at

- Now run a loop on the neighbours of node and

Check if ~~node~~ i is visited

if NOT visited then call DFS (with i as parameter as node) (Recursively)

else Check if

i is visited and i is parent of at

if yes then return true - that cycle exists

else continue the loop

Now

if atleast one DFS calls returns TRUE

then $\exists$ a cycle in the Graph

else $\nexists$ any cycle in the Graph

use flag to return answer

(Q3)

Given a binary tree, $n =$ No. of nodes in the Tree

RTP:

statement → Number of nodes with 2 children + 1 = No. of Leaf nodes.

Proof By Induction:

1) For $n=1$ No. of nodes wit 2 child $= 0$

No. of Leaf nodes $= 1$

∴ The statement is True for $\boxed{n=1}$

2) Now Assume that the statement is True for $\boxed{n=k}$

we have to prove that the statement is True for $\boxed{n=k+1}$

For

$n = k+1$

There are 2 cases to increase the number of nodes as it is a binary Tree Each node can have atmost 2 childs

Now

① Add a node as a child to a Leaf node (node with a child)

then - No. of Leafs don't change

- No. of nodes with 2-children don't change

hence, the statement holds True (as $n=k$)

② Add a node to a node with 1 child

- No. of Leaf nodes increases by 1
- No. of nodes with 2 child increases by 1

Thus

$$\text{No. of leaf's} + 1 = \left(\begin{array}{c}\text{No. of node}\\ \text{wt 2 child}\end{array} + 1\right) + 1$$

Hence,

The statement holds True ~~f (n=k+1)~~

for ($n = k+1$)

(∵ we cannot add a node to node with 2 child)

Hence, proved by Induction that the statement is True

Q4

Given, a Graph $G = (V, E)$

- make a DFS Tree from vertex $u \in V$
- make a BFS Tree from vertex $u \in V$

RTP: If DFS Tree = BFS Tree = $\boxed{T}$

then prove that $\boxed{G = T}$

Here there are 2 conditions to prove

① If Graph G is a <u>Tree itself</u> then, as there is a unique/distinct path from every node to every other node both DFS and BFS will return the same Tree as <u>G</u>

Hence, if DFS Tree = BFS Tree & G is a Tree

then $\boxed{G = T}$

② Now ~~If~~ Graph G is <u>NOT a Tree</u> and Assume there is a cycle in Graph G.

Then, let nodes in the cycle be $a_1, a_2, \ldots, a_m, a_1$

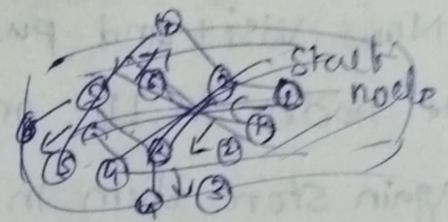
~~BFS~~

Now $a_1, a_2, \ldots a_m, a_1$

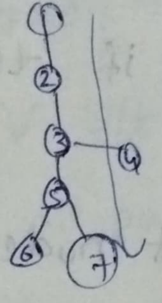
DFS:

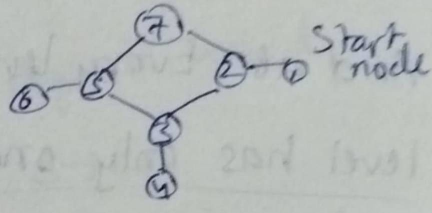
whenever we visit a node in the cycle we will all the nodes of the cycle in a single path (even if they have other neighbours) as shown below.

let the Graph (G) be



DFS Tree →





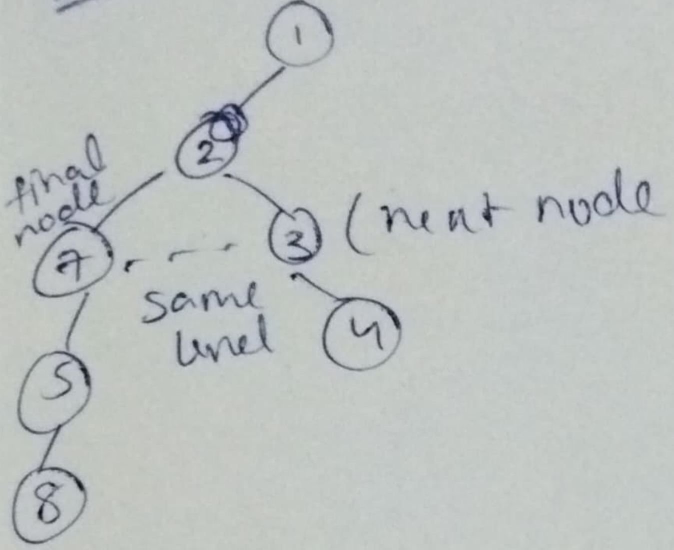
Here ③, ⑤ have other neighbours still there is a straight path from ① to ⑦.

BFS:

when we visit a node in the cycle (1st node)

the node next to it (to be visited) and the last node in the cycle are in the same level as they are neighbours of (1st node)

BFS Tree



$\therefore$ DFS Tree $\neq$ BFS Tree

Hence our Assumption is wrong and DFS Tree = BFS Tree iff Graph(G) is a Tree

But from (*) condition (1) if G is a Tree then $\boxed{G = T = \text{BFS Tree} = \text{DFS Tree}}$

Hence, proved.

Q5

Given a Graph (G) with n nodes

- n is even

RTP: If every node of G has degree atleast $n/2$, then G is connected

proof By Contradiction:

Let the Graph (G) is not connected with each node having degree $\geq \frac{n}{2}$

$\Rightarrow$ It should have atleast 2 nodes (node1, node2) s/t there is no Edge between them and $\frac{n}{2}$ nodes that this nodes connect to are disjoint (otherwise there is a path from (node1 $\to$ node2) $\Rightarrow$ connected)

Thus Total No. of nodes

$= \underset{\text{node1}}{(1)} + \underset{\text{neighbours of node 1}}{\left(\frac{n}{2}\right)} + \underset{\text{node2}}{(1)} + \underset{\text{neighbours of node2}}{\left(\frac{n}{2}\right)}$

$= 1 + \frac{n}{2} + 1 + \frac{n}{2} = \boxed{n+2 \geq n}$

This is a Contradiction!

$\therefore$ G is connected

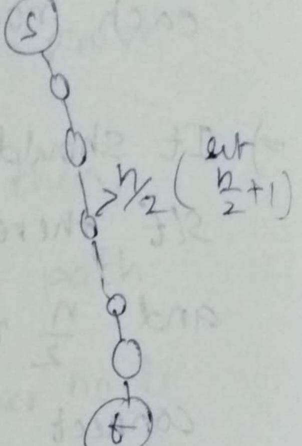
(Q6)

Given an $\underline{n \text{ node}}$ undirected graph $G = (V, E)$

2 nodes $\underline{s \& t}$ s/t distance between them is $> \frac{n}{2}$

$\underline{\text{RTP}}$: $\exists v$ (node) s/t deleting v would destroy all paths from $s \to t$. and $v \neq s, v \neq t$

distance $(s \to t) > \frac{n}{2}$



let distance be $= \frac{n}{2} + 1$

$\Rightarrow$ There are $\frac{n}{2}$ in the PATH between $s \& t$.

- we cannot build one more such path directly joining $s \& t$ without using these $\frac{n}{2}$ nodes

as we have remaining $= n - \frac{n}{2} - 2$ (s, t node)

$= \frac{n}{2} - 2$ nodes

which can give a distance of max

$\boxed{\frac{n}{2} - 1 < \frac{n}{2} + 1}$

Thus, there is only single path with distance $> \frac{n}{2}$ with all distinct nodes.

But we should have atleast 2 nodes at each level from s to t for

v not to exist.

But there is only one ~~one~~ single path from $s \to t$ with distance $\geq \frac{n}{2}$ with all distinct nodes

$\Rightarrow$ If there are 2 nodes at each level

then Total No. of nodes $\geq \frac{n}{2} \cdot 2 = n + 2$
$+ 2 (s, t)$
$= \boxed{n + 2 > n}$

is a Contradiction!

$\therefore \exists v \in V$ s/t on deleting v would destroy all paths from $s \to t$.

As For v to exist we need atleast 2 nodes at each level. (in BFS call)

Algorithm to find v :-

- we use the idea the if BFS is done then at some level only one-node will be there according to the contradiction

① START BFS from node's s and push all its neighbours to the queue and store them in Level0 (L0) - ~~a~~ array (2d-array)

② Now visit and push all the neighbours of nodes in L0 onto queue again store them in L1 - array of 2d-array

③ And at Every level check if that level has only one node
if yes return that node as v

④ As $\exists v$, we gaurantee that we get that (v) node at some point in the traversal

⑤ As BFS runs in $O(n+m)$ this algorithm's Time complexity $= O(n+m)$ only