**BFS in Binary Tree**

## **Problem-1**

<https://leetcode.com/problems/binary-tree-level-order-traversal/description/>

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

vector<vector<int>> levelOrder(TreeNode\* root) {

vector<vector<int>> ans;

queue<TreeNode\*> q;

if(root!=NULL)

{

q.push(root);

}

while(!q.empty()){

//\*\*\* this is the Most Important Line, with this "len" size we can Track the "Level" of the BFS... otherwise, we won't be able to determine how many element belongs to SAME Level\*\*\*

int len = q.size(); //This Line is Most Important

vector<int> tempVec;

for(int i=0;i<len;i++)

{

TreeNode\* node = q.front();

q.pop();

tempVec.push\_back(node->val);

if(node->left!=NULL){

q.push(node->left);

}

if(node->right!=NULL){

q.push(node->right);

}

}

ans.push\_back(tempVec);

}

return ans;

}

};

## **Problem-2**

<https://leetcode.com/problems/average-of-levels-in-binary-tree>

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

vector<double> averageOfLevels(TreeNode\* root) {

queue<TreeNode\*> q;

q.push(root);

vector<double> ans;

while(!q.empty()){

//\*\*\* this is the Most Important Line, with this "n" size we can Track the "Level" of the BFS... otherwise, we won't be able to determine how many element belongs to SAME Level\*\*\*

int n = q.size(); //This Line is Most Important

long int sum=0; // "long" is Required otherwise, Sum will Overflow because of Testcase

for(int i=0;i<n;i++) {

TreeNode\* current = q.front();

q.pop();

sum+=current->val;

if(current->left!=NULL){

q.push(current->left);

}

if(current->right!=NULL){

q.push(current->right);

}

}

ans.push\_back((double)sum/n);

}

return ans;

}

};