PRINTING TOP-VIEW ELEMENT OF A BINARY TREE

Mera Code

```
#include<iostream>
#include<queue>
#include<unordered_map>
#include <climits>
using namespace std;

class TreeNode{
  public:
  int val;
  TreeNode* right;
  TreeNode* left;
  TreeNode(int val){
    this→val = val;
    right = NULL;
    left = NULL;
  }
};

void Top_View(TreeNode* root){
  unordered_map<int ,int> m;
  queue<pair<TreeNode*,int>> q;
  pair<TreeNode*,int> r;
  r.first = root;
  r.second = 0;//level of the root
  q.push(r);
  //lets initiate the bfs
  while(q.size()>0){
```

```
    TreeNode* temp = q.front().first;//this is the TreeNode
    int temp_val = temp->val;
    int level = q.front().second;
    q.pop();
    if(m.find(level) == m.end()){
      m[level] = temp_val;
    }
    if(temp -> left != NULL){
      pair<TreeNode*,int> r;
      r.first = temp->left;
      r.second = level - 1;
      q.push(r);
    }
    if(temp -> right != NULL){
      pair<TreeNode*,int> r;
      r.first = temp->right;
      r.second = level + 1;
      q.push(r);
    }
  }
  int max_lvl = INT_MIN;
  int min_lvl = INT_MAX;
  for(auto x : m){
    max_lvl = max(x.first,max_lvl);
    min_lvl = min(x.first,min_lvl);
  }
  for(int i=min_lvl; i<=max_lvl; i++){
    cout<<m[i]<<" ";
  }
  cout<<endl;
}
int main(){
  TreeNode* a = new TreeNode(1);
  TreeNode* b = new TreeNode(2);
  TreeNode* c = new TreeNode(3);
  TreeNode* d = new TreeNode(4);
```

```
  TreeNode* e = new TreeNode(5);
  a -> left = b;
  a -> right = c;
  b -> left = d;
  b -> right = e;
  Top_View(a);
  return 0;
}
```

Ab iska mera breakdown

Step 1 – Mera Intention

Mujhe Binary Tree ka **Top View** print karna hai.

Top View matlab har vertical line me jo sabse upar ka node hai, wahi output me aayega.

Step 2 – Mera Plan

- Main **BFS** use karunga taaki level-by-level traverse ho.
- Har node ke saath main uska **horizontal level** (root = 0, left = -1, right = +1) store karunga.
- Main ek **map** use karunga jo first time aane wale horizontal level ka node store kare.
- Traversal complete hone ke baad, min horizontal level se max horizontal level tak print kar dunga.

Step 3 – BFS Execution

- Main ek queue banata hoon `(TreeNode*, level)` type ki.
- Root ko `(root, 0)` push karta hoon.
- Jab tak queue khali nahi hoti:
  - Front node nikalta hoon.

- Agar uska horizontal level pehle se map me nahi hai → store karta hoon.
- Uska left child `(level - 1)` pe push karta hoon.
- Uska right child `(level + 1)` pe push karta hoon.

Step 4 – Min/Max Levels

- Map iterate karke min aur max horizontal level nikalta hoon.
- Min se max tak loop chala ke map ka value print kar deta hoon.

Dry Run Example

Tree:

```
    1
   /   \
  2     3
 / \
4   5
```

Initial State:

```
Queue: [(1, 0)]
Map: {}
```

BFS Steps:

1. Pop (1, 0) → m[0] = 1

   Push (2, -1), (3, 1)

   Queue: [(2, -1), (3, 1)]

   Map: { 0: 1 }

2. Pop (2, -1) → m[-1] = 2

   Push (4, -2), (5, 0)

   Queue: [(3, 1), (4, -2), (5, 0)]

Map: { 0: 1, -1: 2 }

3. Pop (3, 1) → m[1] = 3

   Queue: [(4, -2), (5, 0)]

   Map: { 0: 1, -1: 2, 1: 3 }

4. Pop (4, -2) → m[-2] = 4

   Queue: [(5, 0)]

   Map: { 0: 1, -1: 2, 1: 3, -2: 4 }

5. Pop (5, 0) → level 0 already exists in map → skip

   Queue: []

Min/Max Levels:

```
min_lvl = -2
max_lvl = 1
```

Output Order:

```
m[-2] → 4
m[-1] → 2
m[ 0] → 1
m[ 1] → 3
Final Output: 4 2 1 3
```
