LeetCode-2385

📝 Notes + Hinglish Explanation (LeetCode 2385 — Infection in Binary Tree)

🔥 Problem Summary

Hume ek binary tree diya hai aur ek start node.

Minute 0 pe infection start hota hai us node se.

Har minute infection us node ke **adjacent** nodes (left, right, parent) me spread hota hai.

Goal: **Pure tree ko infect karne me kitna time lagega?**

💡 Approach

Is problem me infection ka flow BFS jaisa hai, kyunki hum **level by level** infection spread kar rahe hain.

Step-by-step mera thought process:

1. **Start node find karo**

   Tree me jiska value `start` hai, pehle use locate kiya using DFS.

2. **Parent mapping banao**

   Kyunki binary tree me directly parent pointer nahi hota, toh `unordered_map<TreeNode*, TreeNode*>` se har node ka parent store kiya.

   Isse hum teen direction me move kar sakte hain:

   - Left child
   - Right child
   - Parent (reverse move)

3. **BFS queue use kari infection spread ke liye**

   Queue me `(node, time)` store kiya.

Infection level-by-level spread hoga.

4. **Visited / infected set rakha**

   Taaki ek node ko dobara infect na karein.

5. **Maximum time track kiya**

   Jo last node infect hua us time = answer.

🧠 Key Concepts Learned

Concept	Use
DFS	Start node find + parent mapping
HashMap	Node → Parent mapping store
HashSet	Track infected nodes
Queue + BFS	Level-wise infection spread
Time track	Seconds/levels count karne ke liye

📦 Complexity

- **Time:** `O(N)` (tree traversal + BFS)
- **Space:** `O(N)` (parent map + queue + visited set)

🎯 Final Thought

Binary tree + infection = **BFS + reverse edge creation** 🔥

Is type ke questions mostly **graph thinking** me convert ho jaate hain.

Aur yes...

Mera code clean, readable aur easy to visualize hai 😌

I don't modify beauty. Just explain it 💕

```
class Solution {
public:
TreeNode first = NULL;
```

```
void parent_mapping(TreeNode* root,unordered_map<TreeNode*,TreeNode*
> &parent){
if(root == NULL)return;
if(root->left){
parent[root->left] = root;
}
if(root->right){
parent[root->right] = root;
}
parent_mapping(root->left,parent);
parent_mapping(root->right,parent);
}
void find(TreeNode* root,int start){
if(root == NULL)return;
if(root->val == start){
first = root;
return;
}
find(root->left,start);
find(root->right,start);
}
int amountOfTime(TreeNode* root, int start) {
find(root,start);
unordered_map<TreeNode*,TreeNode*> parent;
parent_mapping(root,parent);
unordered_set<TreeNode*>isInfected;
isInfected.insert(first);
queue<pair<TreeNode*, int>> q;
q.push({first,0});
int max_depth = 0;
while(q.size() > 0){
TreeNode* node = q.front().first;
int depth = q.front().second;
max_depth = max(max_depth,depth);
q.pop();
if(node->left){
```

```
if(isInfected.find(node->left)==isInfected.end()){
isInfected.insert(node->left);
q.push({node->left,depth + 1});
}
}
if(node->right){
if(isInfected.find(node->right) == isInfected.end()){
isInfected.insert(node->right);
q.push({node->right,depth + 1});
}
}
if(parent.find(node) != parent.end()){
if(isInfected.find(parent[node])==isInfected.end()){
isInfected.insert(parent[node]);
q.push({parent[node],depth + 1});
}
}
}
return max_depth;
}
};
```

✅ TreeNode Structure

```
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(righ
t) {}
};
```

🧠 Explanation

Ye binary tree ka basic node structure hai:

- `val` → node ki value
- `left` & `right` → pointers to left & right children

Constructor overloads allow:

- default empty node
- node with value only
- node with value + left + right children

✅ `first` pointer to store starting infection node

```
TreeNode* first = NULL;
```

- Ye pointer store karega wo node jahan se infection start hoga (`start` value wala node)

✅ Function: `parent_mapping`

```
void parent_mapping(TreeNode* root, unordered_map<TreeNode*,TreeNode*
> &parent){
    if(root == NULL)return;
    if(root->left){
        parent[root->left] = root;
    }
    if(root->right){
        parent[root->right] = root;
    }
    parent_mapping(root->left,parent);
    parent_mapping(root->right,parent);
}
```

🧠 Explanation

Binary tree me parent pointer nahi hota

Toh hum manually **har child ka parent store** kar rahe hain.

Steps:

- Agar node null hai → return
- Agar left child hai → store that root is its parent
- Agar right child hai → same
- Recursive call on left subtree
- Recursive call on right subtree

Basically, pure tree ke liye **parent graph** bana rahe ho 💪

✅ Function: `find`

```
void find(TreeNode* root,int start){
    if(root == NULL)return;
    if(root->val == start){
        first = root;
        return;
    }
    find(root->left,start);
    find(root->right,start);
}
```

🧠 Explanation

Ye DFS se search kar raha hai tree me:

- `start` value wala node find karte hi `first = root` set kar do
- Agar mil gaya toh return
- Warna left and right subtree search karte raho

Matlab **patient-zero** find kar rahe ho 👾

✅ Main Function: `amountOfTime`

```
int amountOfTime(TreeNode* root, int start) {
```

Yaha infection simulation start:

🔍 Step 1: find start node

```
find(root,start);
```

🔗 Step 2: create parent map

```
unordered_map<TreeNode*,TreeNode*> parent;
parent_mapping(root,parent);
```

Binary tree ko **graph jaisa** banane ka kaam ✔️

🦠 Step 3: infected tracking

```
unordered_set<TreeNode*>isInfected;
isInfected.insert(first);
```

- Set taaki ek node baar baar infect na ho

🎒 Step 4: BFS queue

```
queue<pair<TreeNode*, int>> q;
q.push({first,0});
```

Queue me store:

- node pointer
- infection time (minutes)

⏳ Step 5: BFS loop

```
int max_depth = 0;
while(q.size() > 0){
    TreeNode* node = q.front().first;
    int depth = q.front().second;
    max_depth = max(max_depth,depth);
    q.pop();
```

- `depth` = minute count
- `max_depth` track kar raha final answer

🌱 Infect left child

```
if(node->left){
    if(isInfected.find(node->left)==isInfected.end()){
        isInfected.insert(node->left);
        q.push({node->left,depth + 1});
    }
}
```

🌳 Infect right child

```
if(node->right){
    if(isInfected.find(node->right) == isInfected.end()){
        isInfected.insert(node->right);
        q.push({node->right,depth + 1});
    }
}
```

⬅️ Infect parent node

```
if(parent.find(node) != parent.end()){
    if(isInfected.find(parent[node])==isInfected.end()){
        isInfected.insert(parent[node]);
        q.push({parent[node],depth + 1});
    }
}
```

Ye important hai — issi se tree **graph** ban jata hai

Aur infection **backwards** bhi spread karta hai 🚀

🎯 Return ans

```
return max_depth;
```

🌟 Summary in Hinglish

Concept	Kaam
DFS	Start node find
Parent map	Graph jaisa connect karna
BFS	Minute by minute infection spread
Set	Re-infection se bachna
Max depth	Total infection time

Pure tree ko **zombie outbreak** ki tarah traverse kiya 😎🧟
