Binary Tree BFS Type Questions

1. //Binary tree level traversal (LC102)

Create queue
add first element to the queue
while loop:(until queue is empty)
forloop:for the length of queue
1. pop the first element of the queue
2. update output:
3. add children in the queue

```
public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList<>();
        Queue<TreeNode> q = new LinkedList<>();
        if(root==null)
            return result;

        q.add(root);
        while(!q.isEmpty()){
            List<Integer> curList = new ArrayList<>();
            int lengthQueue = q.size();

            //empty the queue for the level
```

Binary Tree BFS Type Questions (cont)

```
            // and insert the child of the elements in the queue
            for(int i =0; i<1-engthQueue;i++){
                TreeNode curr = q.poll();
                curList.add(cur-r.val);
                if(curr.left != null)
                    q.add(cur-r.left);
                if(curr.right != null)
                    q.add(cur-r.right);
            }
            result.add(curList);
        }

        return result;
    }
```

Invert Binary Tree(BFS)

```
public TreeNode invertTree(TreeNode root) {
    if (root == null) return null;
    Queue<TreeNode> queue = new LinkedList<TreeNode>();
    queue.add(root);
    while (!queue.isEmpty()) {
        TreeNode current = queue.poll();
        TreeNode temp = curren-t.left;
        current.left = curren-t.right;
        current.right = temp;
        if (current.left != null) queue.add(current.left);
        if (current.right != null) queue.add(current.right);
    }
    return root;
}
```

Binary Tree BFS Type Questions

2. Shortest path in binary maze

code

Pseduo code:

1. We start from the source cell and calls BFS procedure.

2. We maintain a queue to store the coordinates of the matrix and initialize it with the source cell.

3. We also maintain a Boolean array visited of same size as our input matrix and initialize all its elements to false.

We LOOP till queue is not empty

Dequeue front cell from the queue

Return if the destination coordinates have reached.

For each of its four adjacent cells, if the value is 1 and they are not visited yet, we enqueue it in the queue and also mark them as visited.

Subset

```
LC78
public List<List<Integer>>
subsets(int[] nums) {
    List<List<Integer>> list =
new ArrayList<>();
    Arrays.sort(nums);
    backtrack(list, new ArrayL-
ist<>(), nums, 0);
    return list;
}
private void backtrack(List<Lis-
t<Integer>> list , List<Integer>
tempList, int [] nums, int
start){
    list.add(new ArrayList<>(-
tempList));
    for(int i = start; i <
nums.length; i++){
        tempList.add(nums[i]);
        backtrack(list,
tempList, nums, i + 1);
        tempList.remove(tempL-
ist.size() - 1);
    }
}
```

Subsets, Permutations, Combination Sum, Palindrome Partitioning