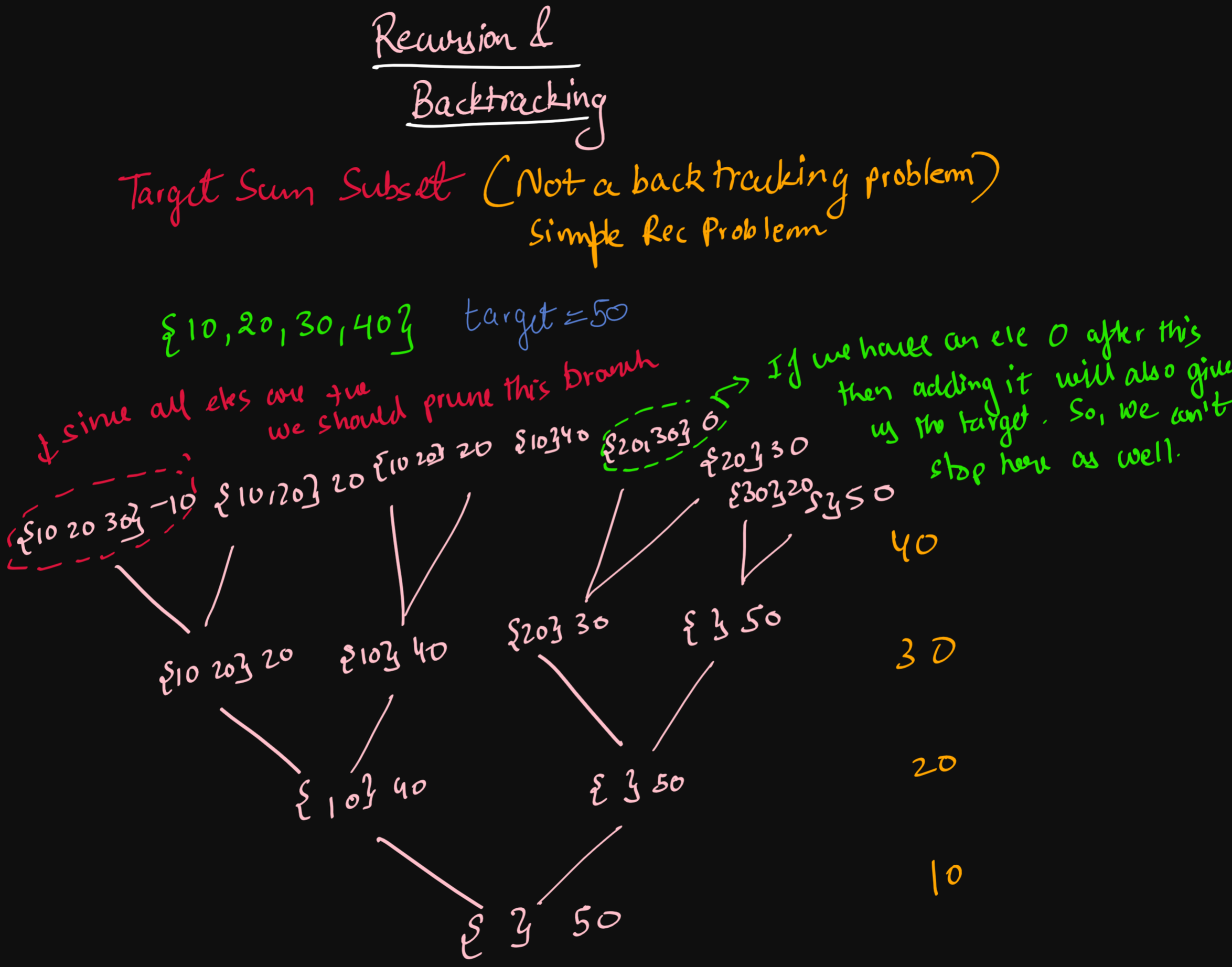
Recursion & Backtracking

Target Sum Subset (Not a backtracking problem)

Simple Rec Problem

{10, 20, 30, 40} target = 50



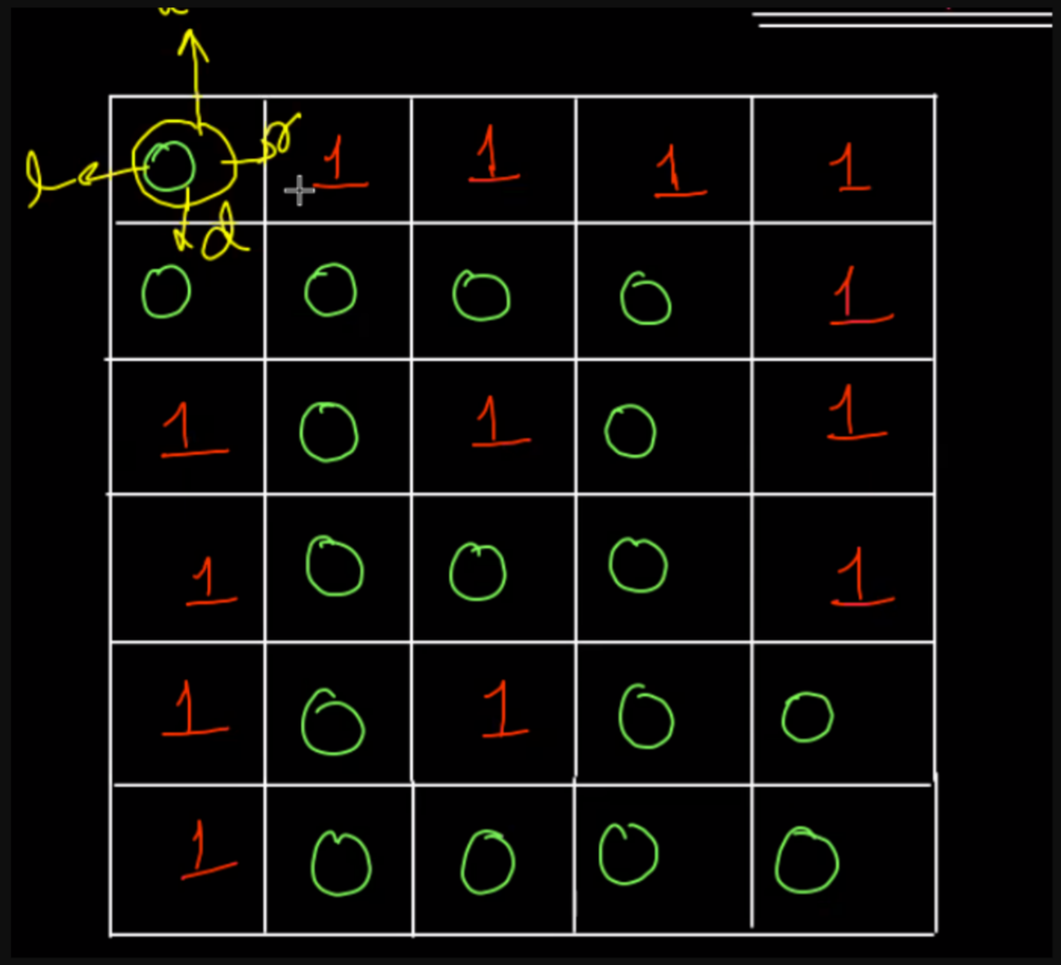
Pruning is very imp in this ques.

→ Negative base case & pruning is diff

Negative base case	Pruning
↓	↓
If we don't take care of this then stack overflow will occur as recursion will keep on going	If we don't take care of this then recursion is eventually going to stop but a lot of extra calls might be made.

```java
public static void printTargetSumSubsets(int[] arr, int idx, String set, int remTarget, int tar) {

    if(idx == arr.length) {
        if(remTarget == 0) {
            System.out.println(set + ".");
        }

        return;
    }

    //pruning
    if(remTarget < 0) {
        return;
    }

    printTargetSumSubsets(arr,idx+1,set + arr[idx] + ", ",remTarget - arr[idx],tar);
    printTargetSumSubsets(arr,idx+1,set,remTarget,tar);
}
```

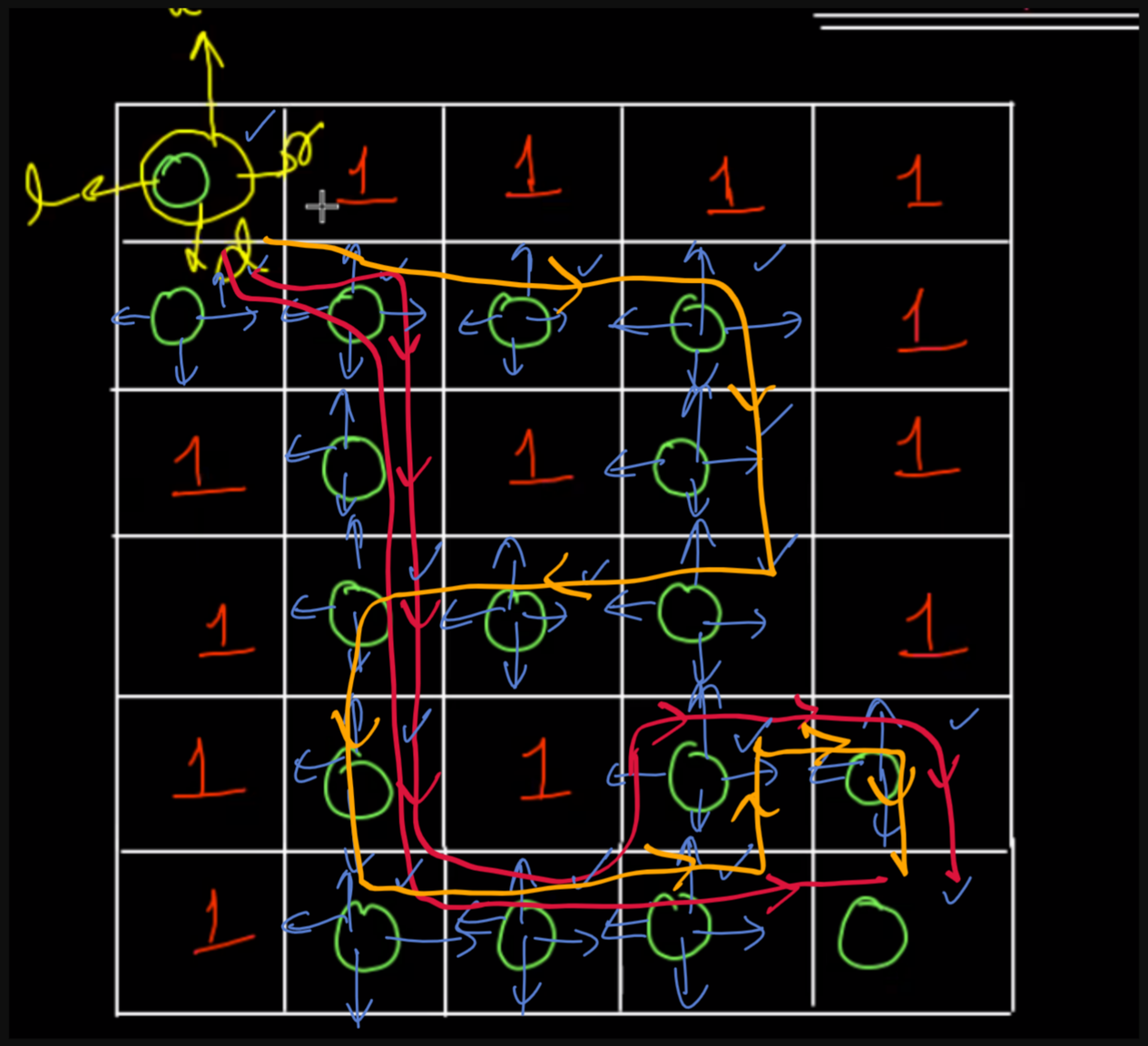
flood fill



↓ This code will cause stack overflow as we end up on visiting same node in an endless cycle.

```java
// asf -> answer so far
public static void floodfill(int[][] maze, int sr, int sc, String asf) {
    if(sr > dr || sc > dc || sr < 0 || sc < 0 || maze[sr][sc] == 1){
        // negative base case
        return;
    }

    if(sr == dr && sc == dc){
        // positive base case
        System.out.println(psf);
        return;
    }

    floodfill(sr - 1, sc, dr, dc, psf + "t"); // top
    floodfill(sr, sc - 1, dr, dc, psf + "l"); // left
    floodfill(sr + 1, sc, dr, dc, psf + "d"); // down
    floodfill(sr, sc + 1, dr, dc, psf + "r"); // right

}
```

So, we need backtracking now.



tldr → order of calls

```java
public static void floodfill(int[][] maze, int sr, int sc, String asf,boolean[][] visited) {

    if(sr < 0 || sc < 0 || sr >= maze.length || sc>= maze[0].length || maze[sr][sc] == 1 || visited[sr][sc] == true) {
        return;
    }
    else if(sr == maze.length - 1 && sc == maze[0].length - 1) {
        System.out.println(asf);
        return;
    }


    visited[sr][sc] = true;
    floodfill(maze,sr-1,sc,asf + "t",visited); //top
    floodfill(maze,sr,sc-1,asf + "l",visited); //left
    floodfill(maze,sr+1,sc,asf + "d",visited); //down
    floodfill(maze,sr,sc+1,asf + "r",visited); //right
    visited[sr][sc] = false;    → backtrack
}
```