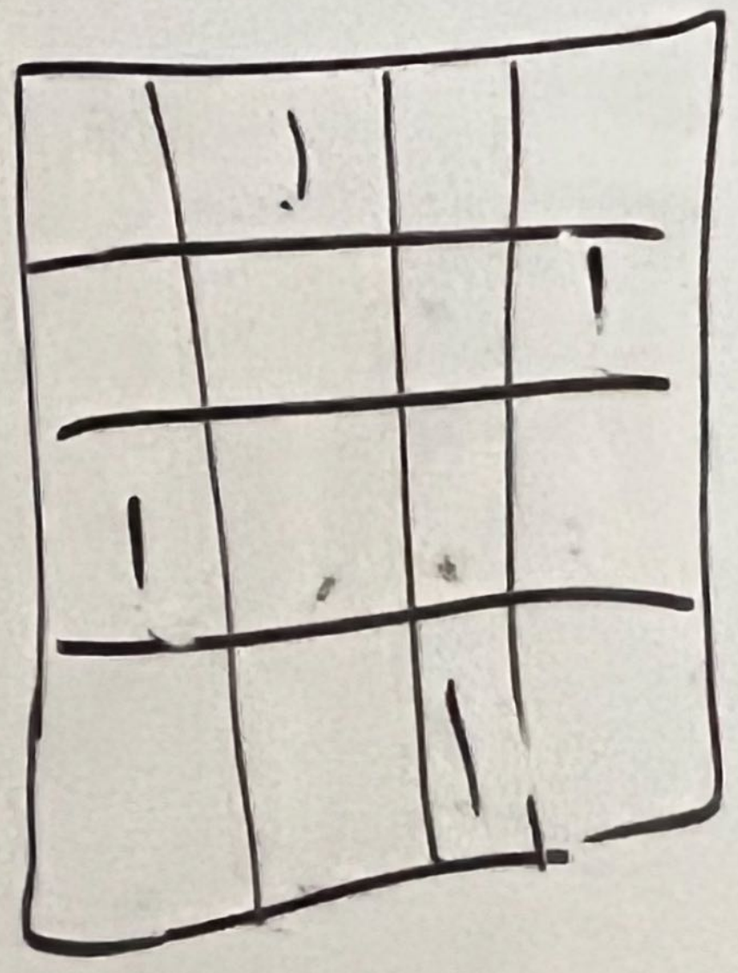
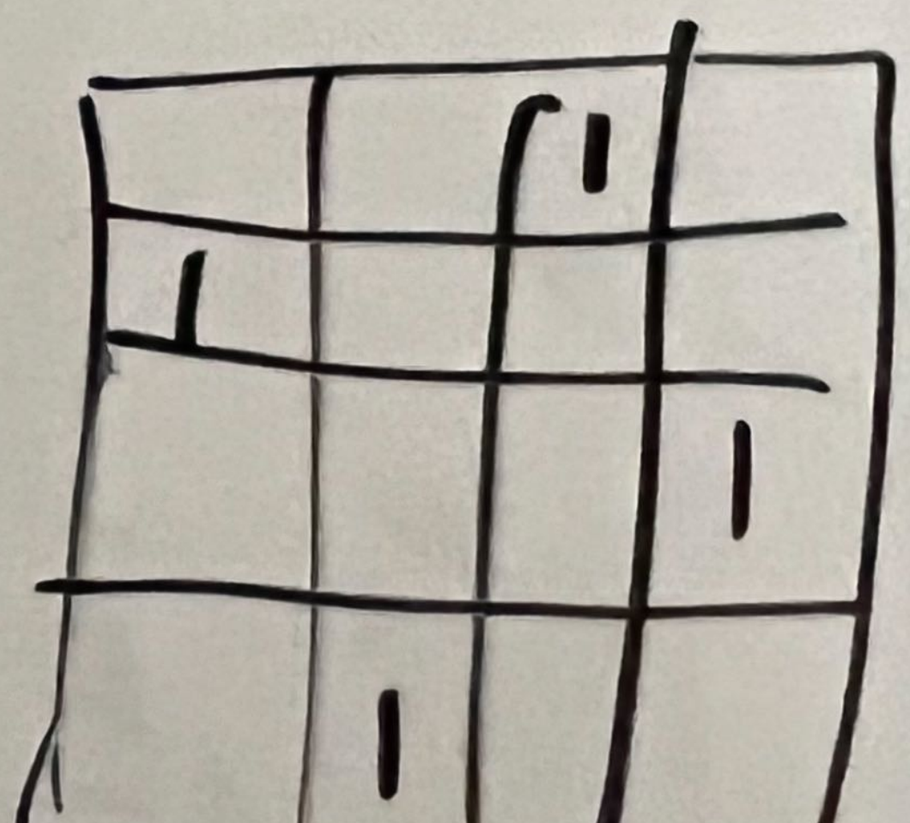
- Design DS
- Hashmap
- Binary Search
- Array
- Two pointers
- Dynamic Programming
- Heaps
- Monotonic Stack
- Linked list
- Trees
- BFS
- DFS
- Backtracking
- Graph





- Tries
- Design (LRU Cache, Autocomplete, Logging Rate)
- Sliding Window
- DP2

Combination Sum

[2,3,6,7]

T = 7

// action

// Recurse

// backtrack

Path*

[10, 7, 2, 5, 8, 13, 4]

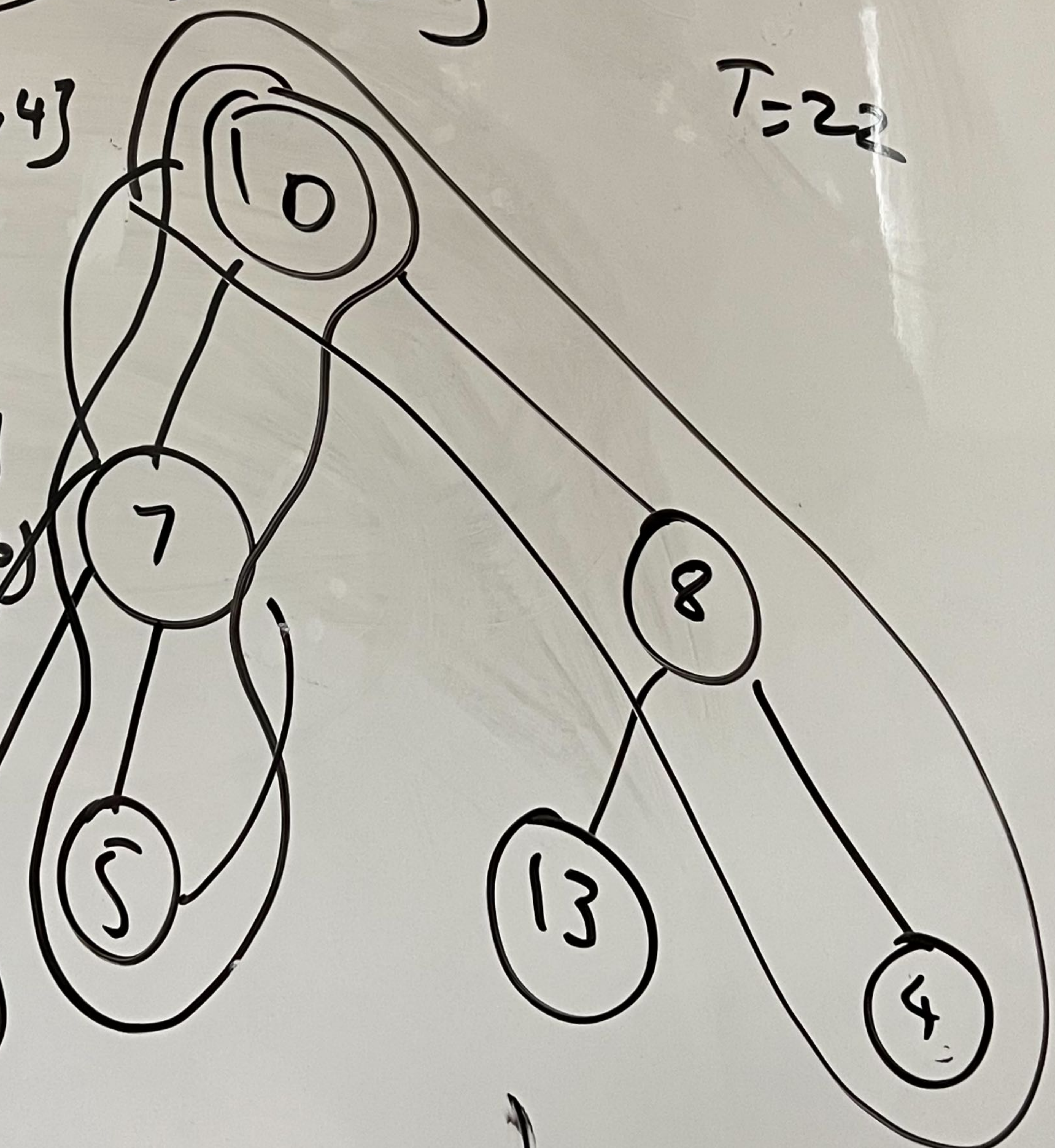
[[ ] , [ ]]

[10, 8, 4]

[10, 7, 5]

```
if curr == T:
    temp = Path[:]
    result.append(temp)
```

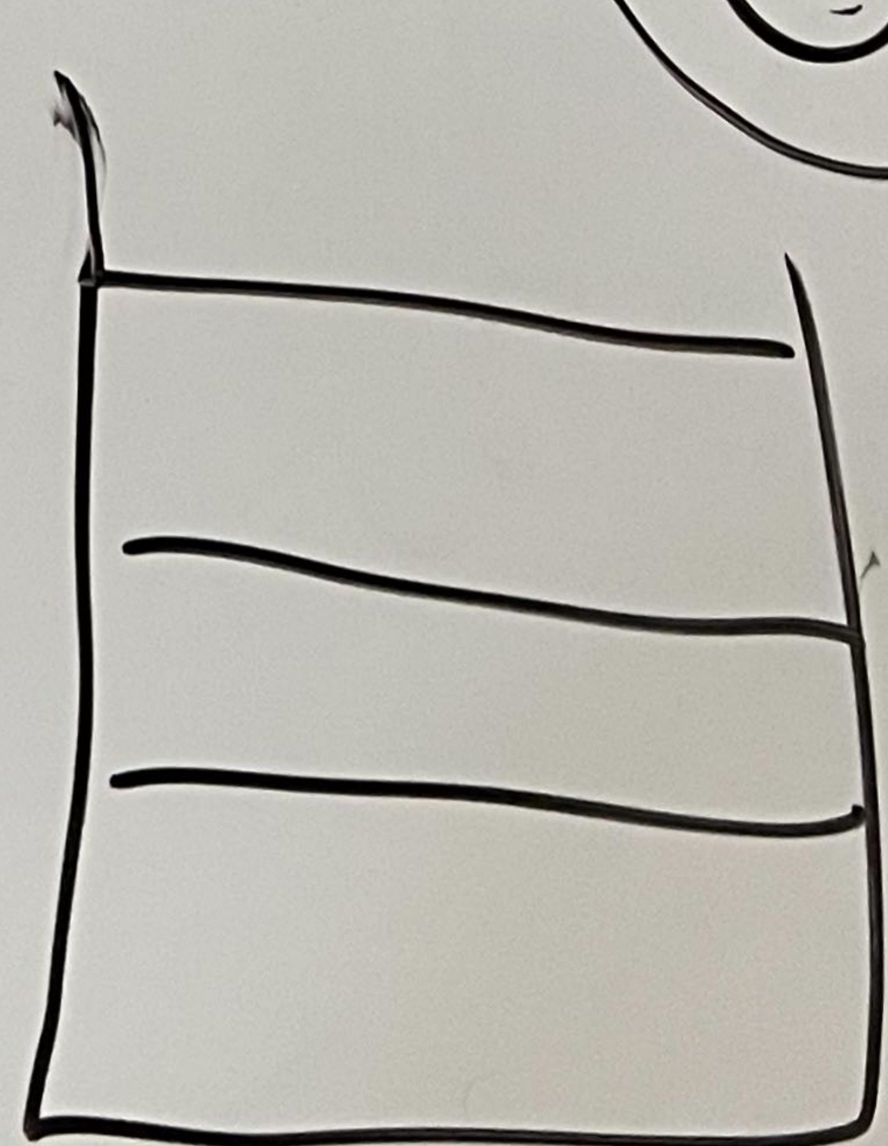
T=22

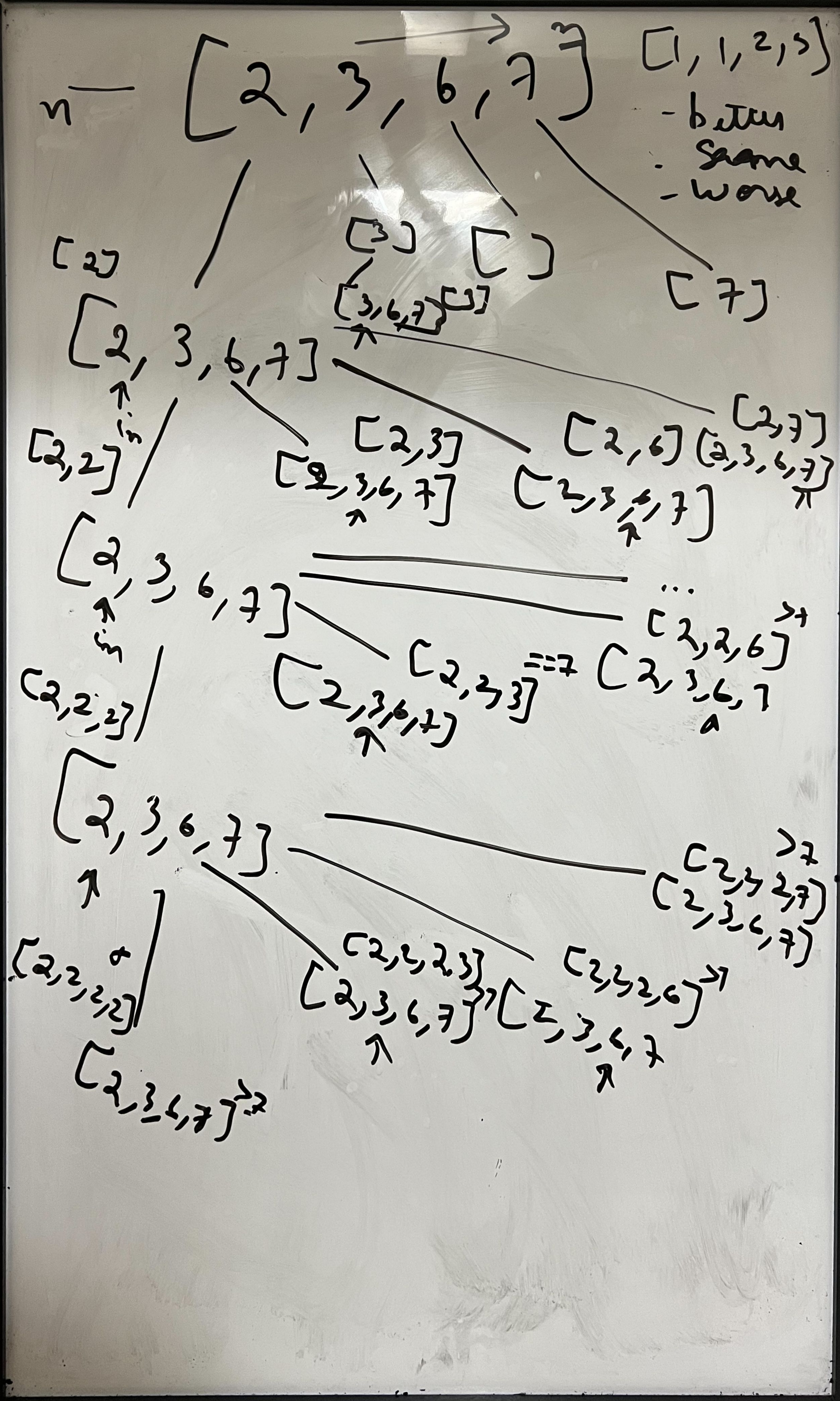


// backtrack

.pop()

Going back to previous state




n
[2, 3, 6, 7]
[1, 1, 2, 3]
- better
- same
- worse
[2]
[3]
[ ]
[7]
[3, 6, 7]
[2, 3, 6, 7]
[2, 2]
[2, 3]
[2, 3, 6, 7]
[2, 6]
[2, 3, 6, 7]
[2, 7]
[2, 3, 6, 7]
[2, 3, 6, 7]
...
[2, 2, 6] >7
[2, 2, 2]
[2, 2, 3] ==7
[2, 3, 6, 7]
[2, 3, 6, 7]
[2, 3, 6, 7]
[2, 2, 2, 2]
[2, 2, 2, 3] >7
[2, 3, 6, 7]
[2, 2, 2, 6] >7
[2, 3, 6, 7]
[2, 2, 2, 7] >7
[2, 3, 6, 7]
[2, 3, 6, 7] >7

Combination Sum recur(target, i, path)

TC: $2^{T/min}$

SC: T/min