Batch — 60 % ——— 70 %

over the long weekend

High Achievers

| Nov23_PSP_24Jan |
| --- |
| VIDYA CHAITANYA |
| Vijay V A |
| Vigneshwaran K |
| Tushar Desarda |
| Kevin Theodore E |
| Mayur Hadawale |
| Sai Sharath |
| Prabhakar |
| Pranadarth S |
| Suraj Devraye |
| Shaurya Srivastava |
| Harshil Dabhoya |
| Yash Malviya |
| Varun |
| Panduranga |
| Shashi |

| Nov23_PSP_24Jan |
| --- |
| RAMESH R S |
| Pratham Singh |
| Rahul Kotha |
| Manjunatha I |
| Pushkar Deshpande |
| SIJU SAMSON |
| Rajeev Singh Khati |
| Sarat Patel |
| Mateen |
| ALLEN GEOSHAN M |
| M.veronica |
| Manikandan M |
| Jyotiranjan Bej |
| Akanksha Narayan Shinde |
| Poornima Patil |
| sowmya |

Quiz 1

For Int N = 7,

Output = 111

```
int magicfun (Int N = 7) {
    Pf (N==0) return 0;
    else return magicfun (N/2) * 10 + (N%2);
}
```

11 * 10 + (7%2)

return 11

```
int magicfun (Int N = 3) {
    Pf (N==0) return 0;
    else return magicfun (N/2) * 10 + (N%2);
}
```

return 1

```
int magicfun (Int N = 1) {
    Pf (N==0) return 0;
    else return magicfun (N/2) * 10 + (N%2);
}
```

return 0

```
int magicfun (Int N = 0) {
    Pf (N==0) return 0;
    else return magicfun (N/2) * 10 + (N%2);
}
```

## Quiz 2

What is the time complexity of above code ?

$$N \;>\; \frac{N}{2} \longrightarrow \frac{N}{4} \longrightarrow \cdots \longrightarrow 0$$

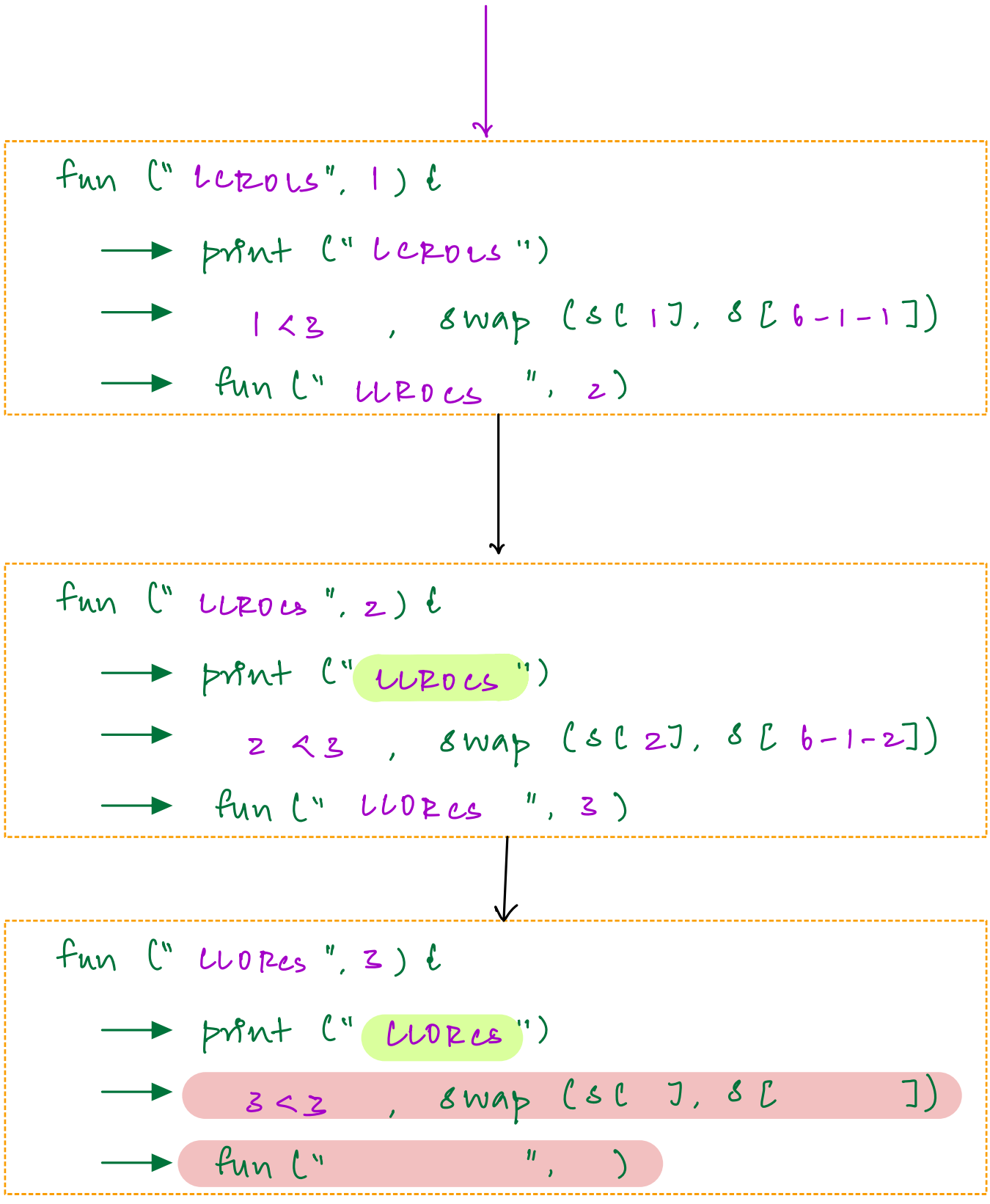
Ans : $O(\log n)$

## Quiz 3

```
void fun (char s[], int x) {
    print (s) ;
    char temp;
    if (x < s.length / 2) {
        swap (s[x], s[s.length - 1 - x]);
        fun (s, x+1);
    }
} fun ("SCROLL", 0);
```

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| S | C | R | O | L | L |

```
fun ("SCROLL", 0) {
    → print ("SCROLL")
    → 0 < 3 , swap (s[0], s[6-1-0])
    → fun ("LCROLS", 1)
```




Quiz 4

what is the time complexity of above code ?

# function calls * T.C (individual call)

= $n/2$ * $O(1)$

= $O(n)$

## Question 1

Given an array with distinct integers. Print all subsets using recursion.

| | Subarray | Subsets |
|---|---|---|
| Define | Contiguous part of array | elements, present in the array not necessarily contiguous |
| # count | $n * (n+1) / 2$ | $2 \times 2 \times 2$ ... $n$ times $= 2^n$ possible subsets |

array :

| 4 | 1 | 9 | 2 | 3 | -1 | 6 | 11 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| | Subarray | Subsets |
|---|---|---|
| 1 9 2 | ✓ | ✓ |
| 1 3 11 | ✗ | ✓ |
| 4 1 2 -1 | ✗ | ✓ |
| -2 1 2 -1 | ✗ | ✗ |

Example

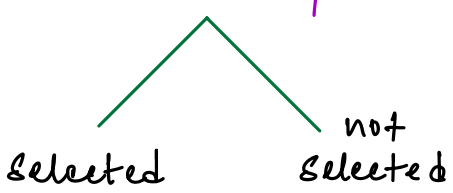
Input : [1, 2, 3]

Output : {}, {1}, {2}, {3}

{1,2} {1,3} {2,3}

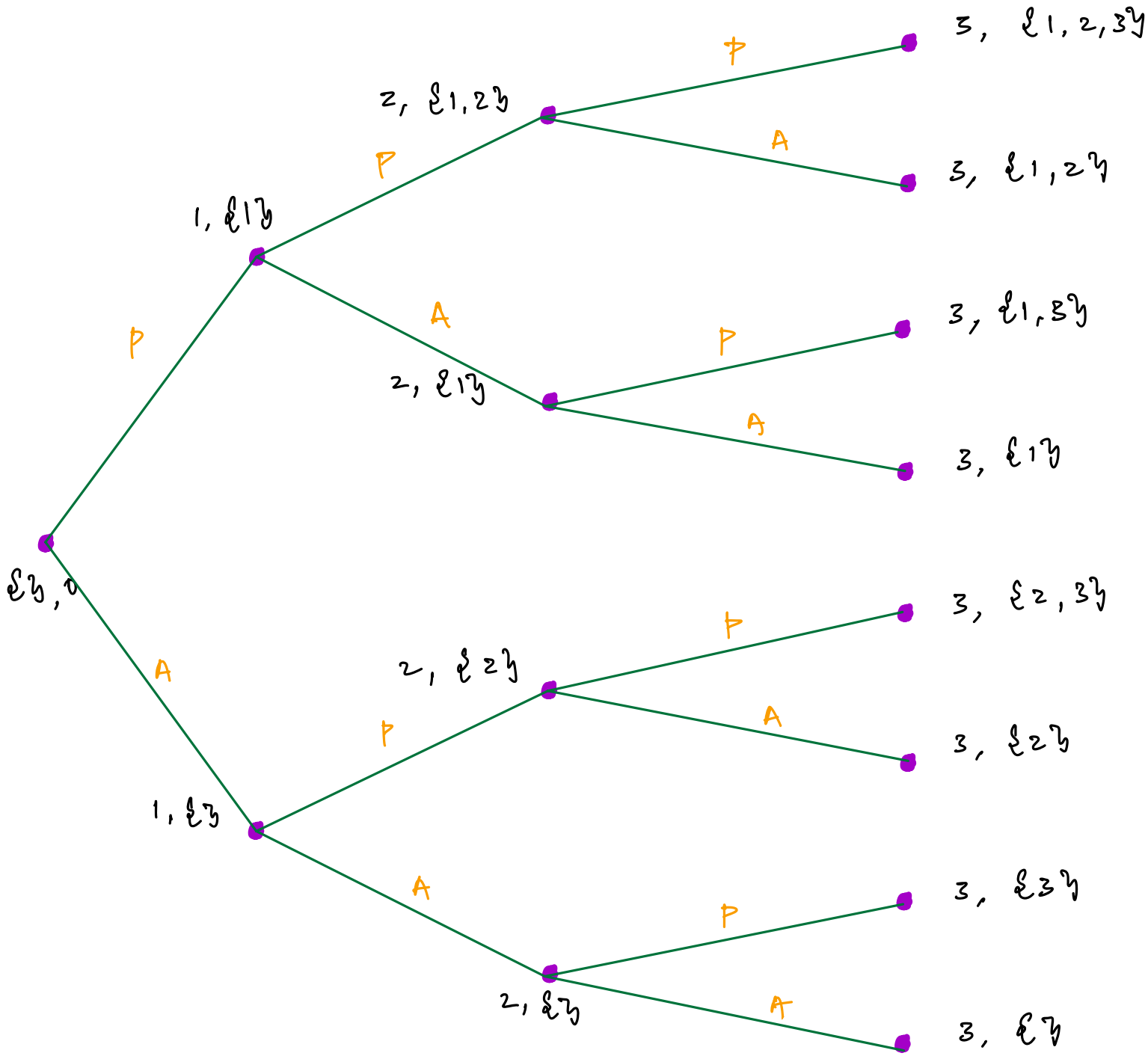
{1,2,3}

each element has 2 options




---

idx, curr list, A A = [1, 2, 3]



# pseudo code

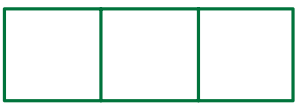
```
List <List> ans ; // Initialize
void subset ( List <Int> A, List <Int> curr, Int idx){
    // Base condition
    if (idx == A.length) {
        ans.push ( copy (curr)) ; return
    }
    // for every element
    // choice 1: consider
    curr.add (A[idx])
    subsets (A, curr, idx+1)
    // choice 2: don't consider
    curr.remove_back ()
    subsets (A, curr, idx+1)
}
```

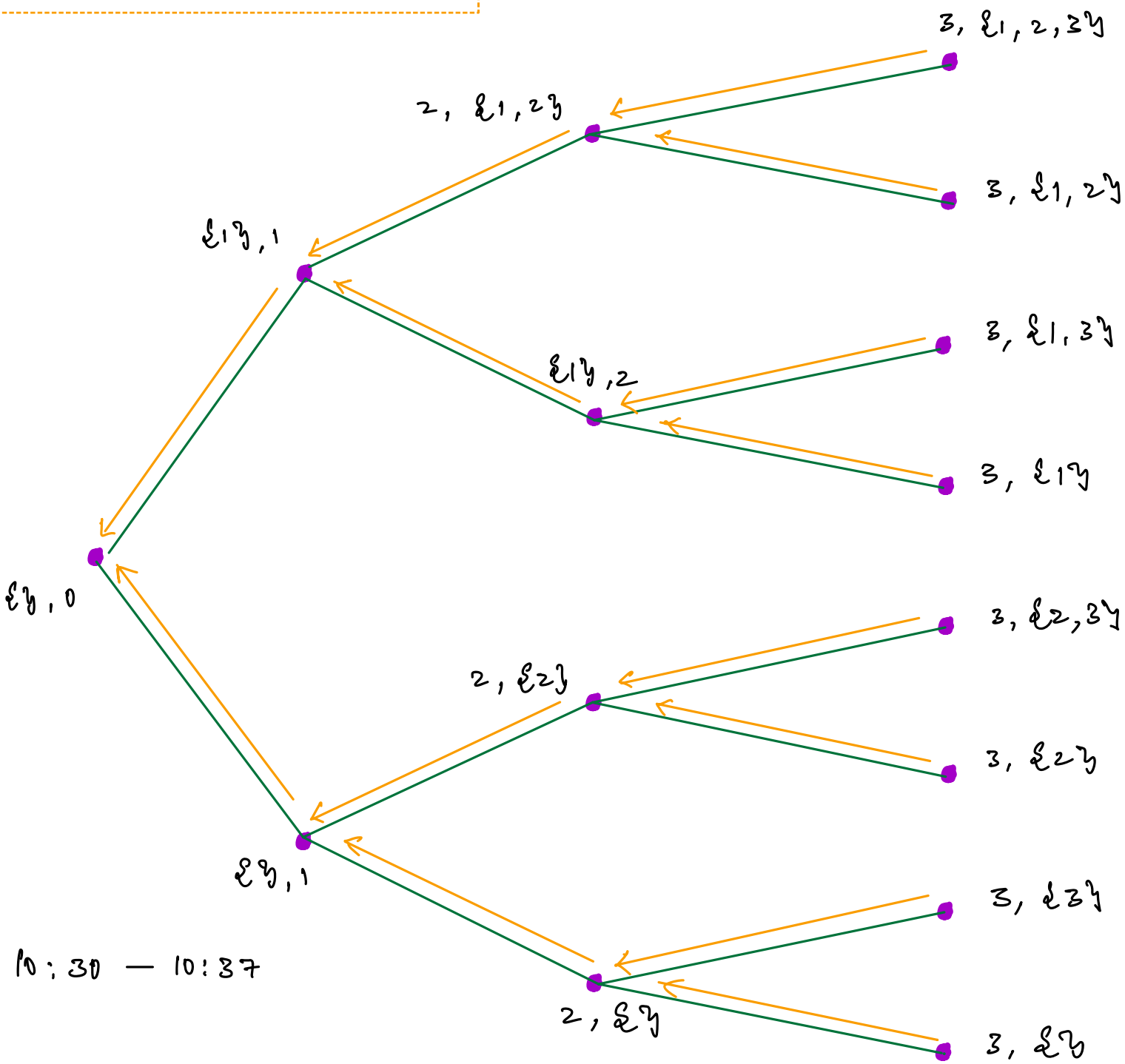
## Dry Run

A : {1, 2, 3} , curr = {} , idx = 0

```
void subsets (A, curr, idx) {
    if (idx == A.length)
        ans.add (curr); return;
    curr.add (A[idx]);
    subsets (A, curr, idx + 1);
    curr.remove_back ();
    subsets (A, curr, idx + 1);
}
```

curr






10 : 30 — 10 : 37

**Permutations** : ways to arrange an array

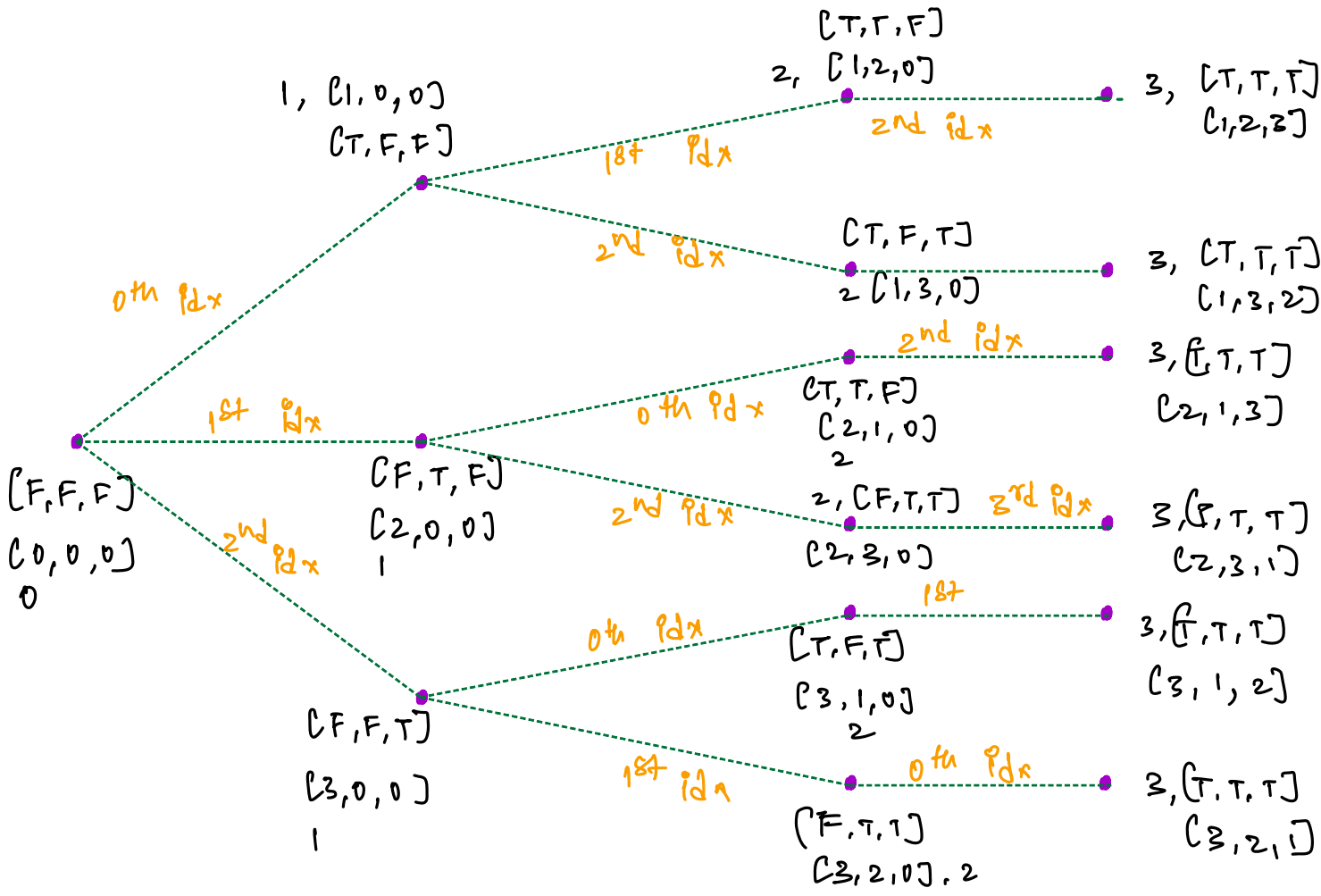
A = [1, 2, 3]

## Permutations of A

| | | |
|---|---|---|
| [1,2,3] | [1,3,2] | [2,1,3] |
| [2,3,1] | [3,1,2] | [3,2,1] |

Given an array A[]. Print all permutations of A[] // only given distinct elements //

pos, used[] // boolean, perm // current permutation



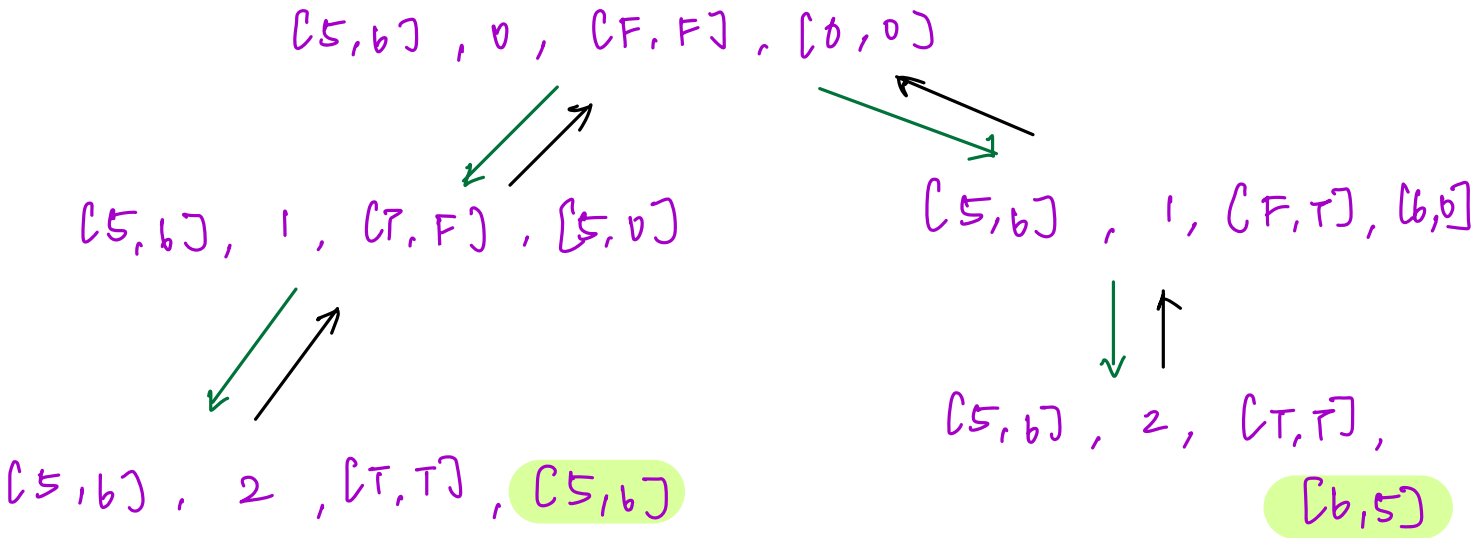
# pseudo code

```
// printing permutation

void permutation (A[], pos, used[], perm[]){
    // Base Condition
    if (pos == a.length)
        print (perm); return;
    // Identify elements already in array
    for (i= 0; i< n; i++) {
        // check used is false
        if (! used[i]) {
            // set to position
            used[i] = true;
            perm[pos] = A[i]
            perm (A, pos +1, used, perm);
            // unset the position
            used[i] = False;
            perm[pos] = 0;
        }
    }
}
```

S.C = O(n)

T.C = n * n!


[5,6] , 0 , [F, F] , [0,0]
[5,6] , 1 , [T, F] , [5,0]
[5,6] , 2 , [T,T] , [5,6]
[5,6] , 1 , [F,T] , [6,0]
[5,6] , 2 , [T,T] ,
[6,5]