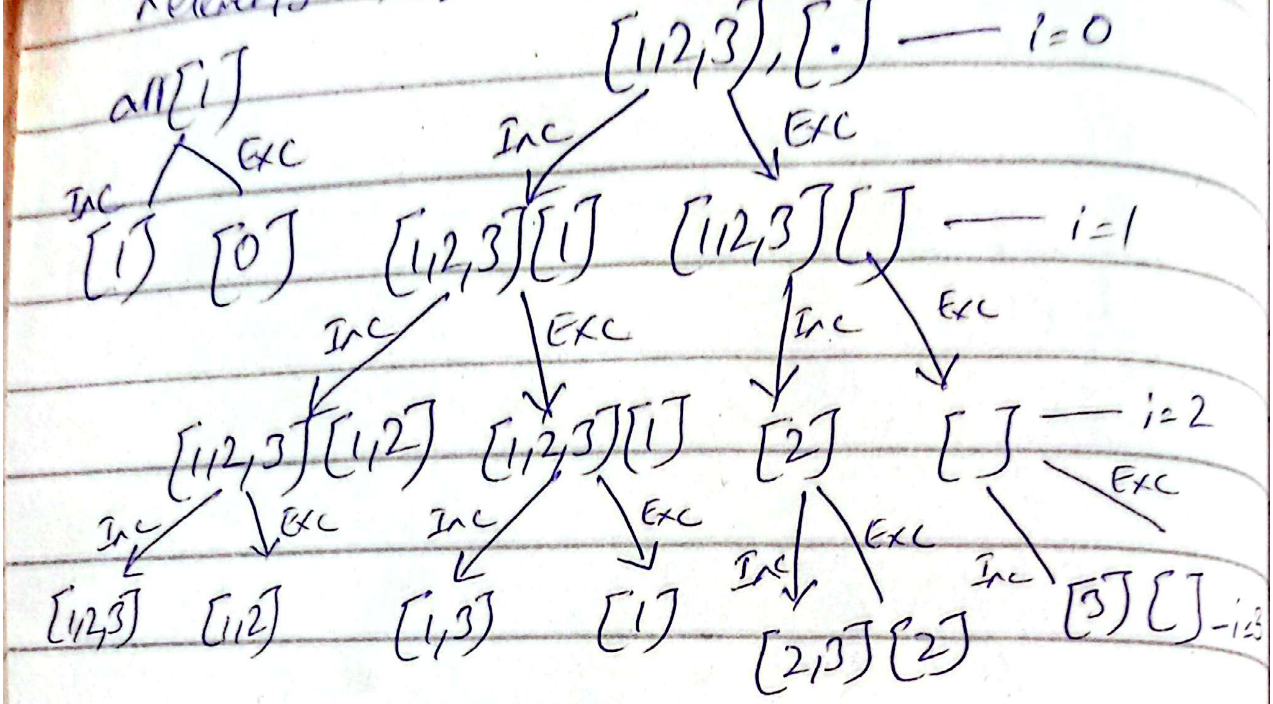
Backtracking with Recursion

Print All Subsets (Subgroups)

arr = {1,2,3} — $2^3 = 8$

n elements → 2^n



```
void PS(arr[], ans[], i){
  if(i==arr.size()) → prints the answer
                       return;
    ans.pushback(arr[i])  ]
    PS(arr, ans, i+1)     ] Inc
      ans.popback() — Backtracking
    PS(arr, ans, i+1)
}
```

T.C = total calls * w.D in each call

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

$\boxed{T.C = O(2^n * n)}$

Subsets II

Array might include duplicate values.
Using previous logic will give the subsets
but it will give repeating subsets.

① Sort the array
② calculate subsets

```
void PS(arr, ans, i, allsubs){
if(i==n) allsubsets(ans)     ] Base
                return       ] case

      ans.pushback([arr[i]])
          PS(arr, ans, i+1, allsubsets)
            ans.popback()
skipping    { idx = i+1
duplicate   { while(([idx<n && A[idx]==A[idx-1])
values      {       idx++
                PS(arr, ans, idx, allsubs)
```

$$\boxed{T.C = O(2^n * n)}$$

CamScanner