Backtracking ⇒ finding all solution by exploring all potential candidates.

1) Print all n digit numbers which can be formed with {1, 2}

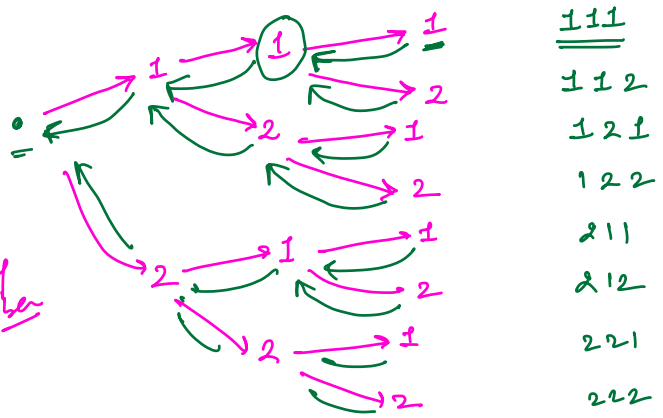
N = 3

1 1 1
1 1 2
1 2 1
1 2 2
2 1 1
2 1 2
2 2 1
2 2 2

1/2 1/2 1/2

1 ✗ 2 dig with {1,2}

2



111
112
121
122
211
212
221
222

parameter

i, n, ans[n] ⇒ making the number

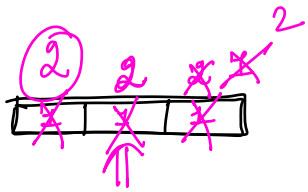
i ⇒ place you are filling

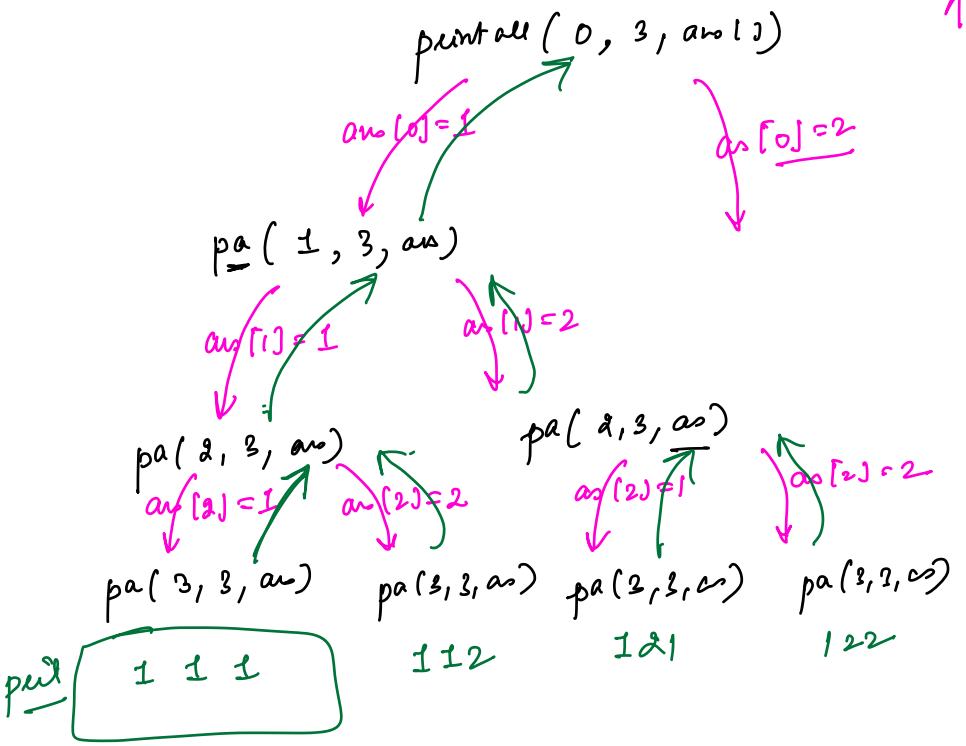
n ⇒ no of digits

0

```
void printall ( int i, int n, int ans[] )
{
    if (i==n) {   for (int j=0; j<n; j++) cout<<ans[j];  return;

    ans[i] = 1;
    printall ( i+1, n, ans);
    ans[i] = 2;
    printall ( i+1, n, ans);
}
```





$T(n) = T(n-1) + T(n-1) + 1$ $\quad T(0) = n$

T.C: $2^n * n$

S.C: $O(n)$

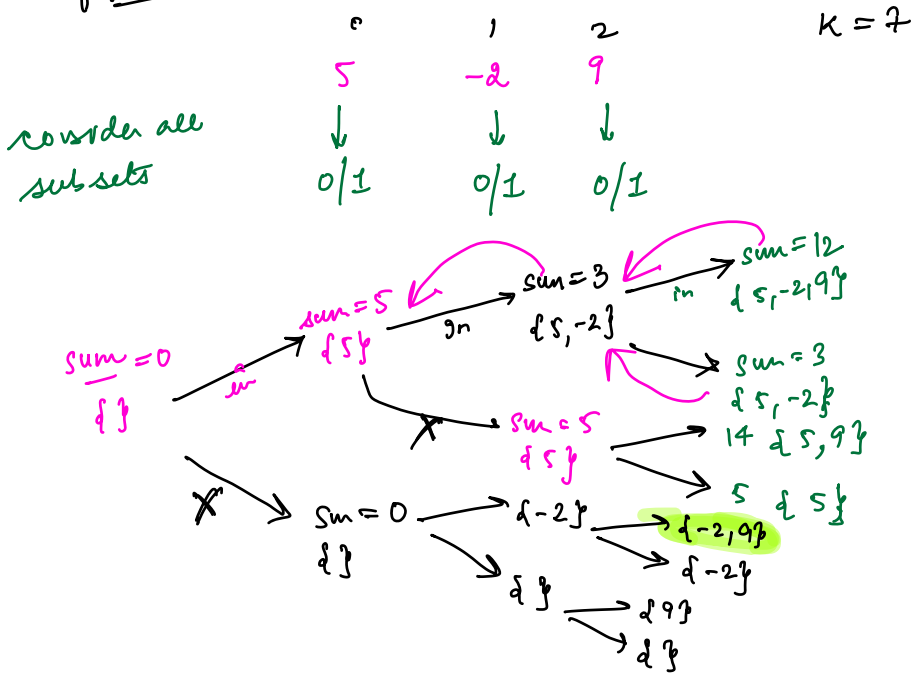
$T(N) = 2T(N-1) + 1$ $\quad TC: O(2^n)$

```
void printall ( int i, int n, int ans[])
{
        if (i==n) {     for (int j=0; j<n; j++) return;
                            print (ans[j]);

        for (x=1; x<=5; x++)                    { 1, 2, 3, 4, 5}
        { ans[i]=x;
            printall ( i+1, n, ans);
        }
}
```

unique elements

- Find count of subsets which has sum = K
  array of N int



params

i, n, sum, arr[], K

```
int sub( int i, int n, int K, int sum, int arr[])
{
    if( i==n) { if ( sum==k) return 1; }
              else return 0;

    sum += arr[i];
    x = sub( i+1, n, k, sum, arr);
    sum -= arr[i];
    y = sub( i+1, n, k, sum, arr);
    return x+y;
}
```

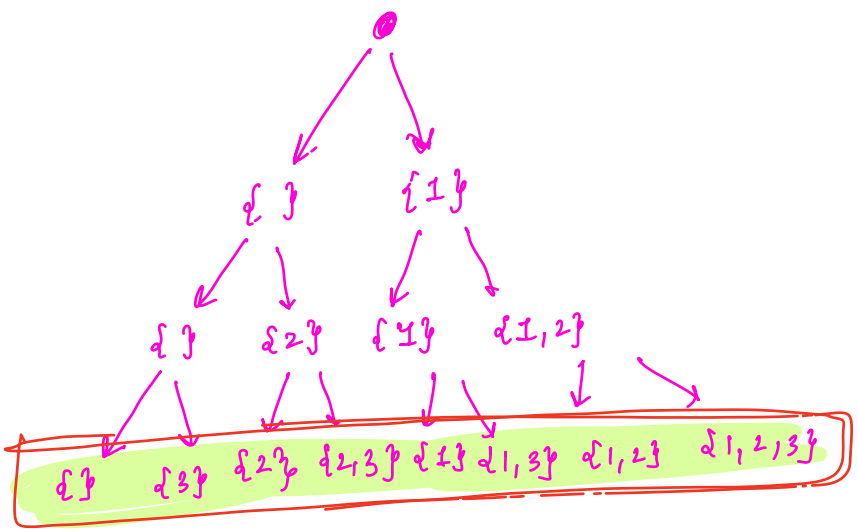
generate all the subsets

list of lists

```
int galsub( int i, int n, int k, List<List<Integer>> finallist, int arr[])
{
    if( i == n) { finallist.insert(list); }

    list.insert( arr[i]);
    sub( i+1, n, k, sum, arr);
    list.pop_back()
    sub( i+1, n, k, sum, arr);
    .
}
```

$T(n) = T(n-1) + T(n-1) + 1$



unique integers, generate all permutations!

rearrangement

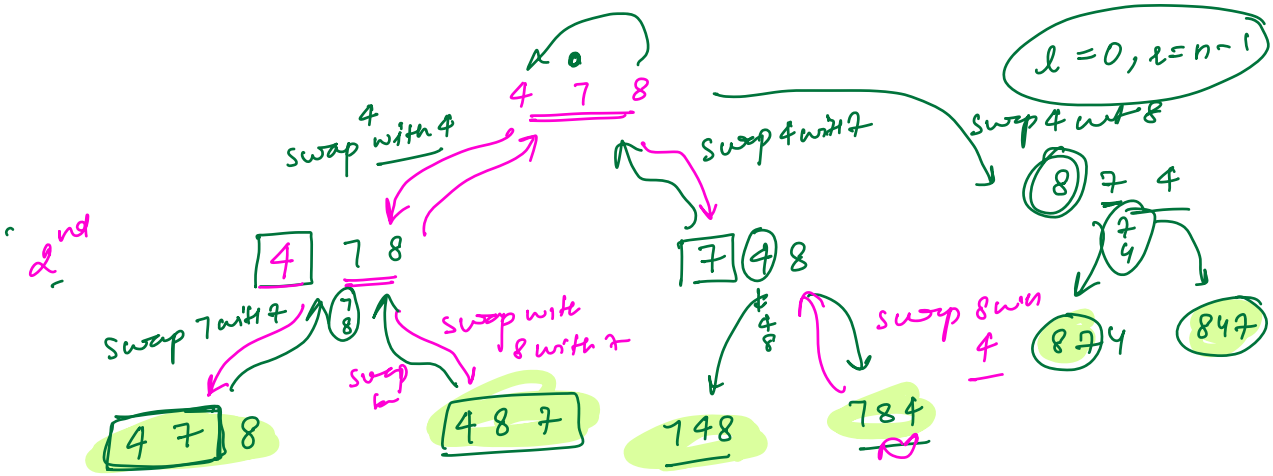
4 7 8

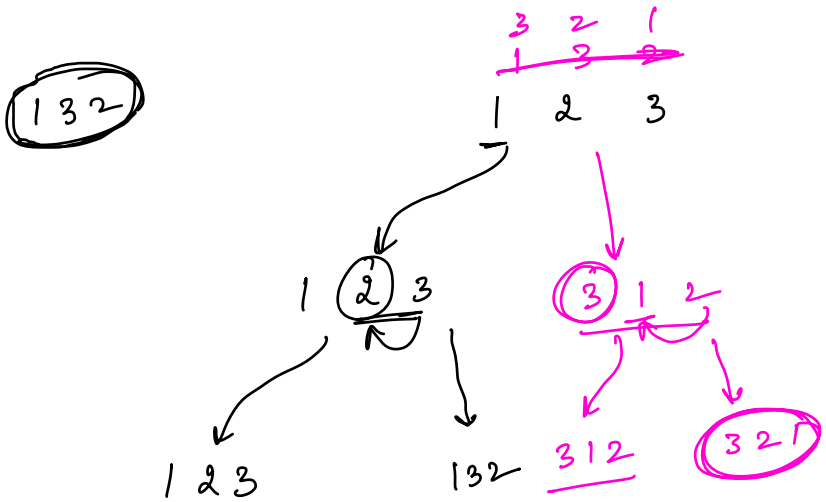
3! = 6

n!

4 7 8
4 8 7
7 4 8
7 8 4
8 4 7
8 7 4

3 2 1





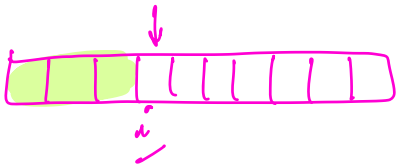
arr, i, n

list of list

```
void getallper( int i, int n, arr[] )
{
    if( i == n) { finalas.push(arr); return; }

    for( l = i; l < n; l++)
    {
        swap( arr[i], arr[l]);
        getallper( i+1, n, arr);
        swap( arr[l], arr[i]);
    }

}
```



T.C :- $n! * n$

```
function( ... ) {
    // base condition;
    Try all possibilities {
        [ ]  do
        Recursive call
        [ ]  undo
        . . . . .
    }
}
```

{4, 7, 8}