Backtracking: finding all solutions by exploring all possible candidates.

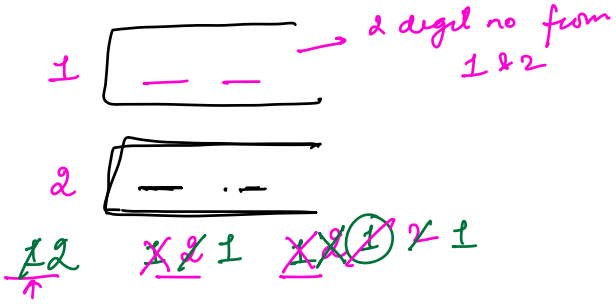
(1) Find all n digit numbers which can be created by 1 or 2.

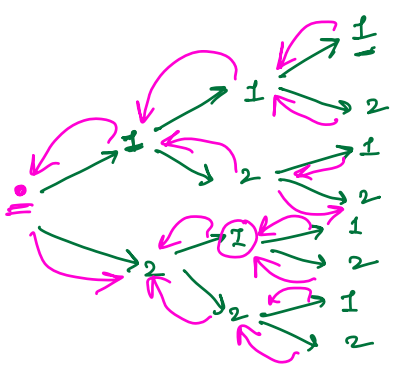
$\Rightarrow 2^n$ $\Leftarrow N = 3$

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

parameters → i, n, ans[n]

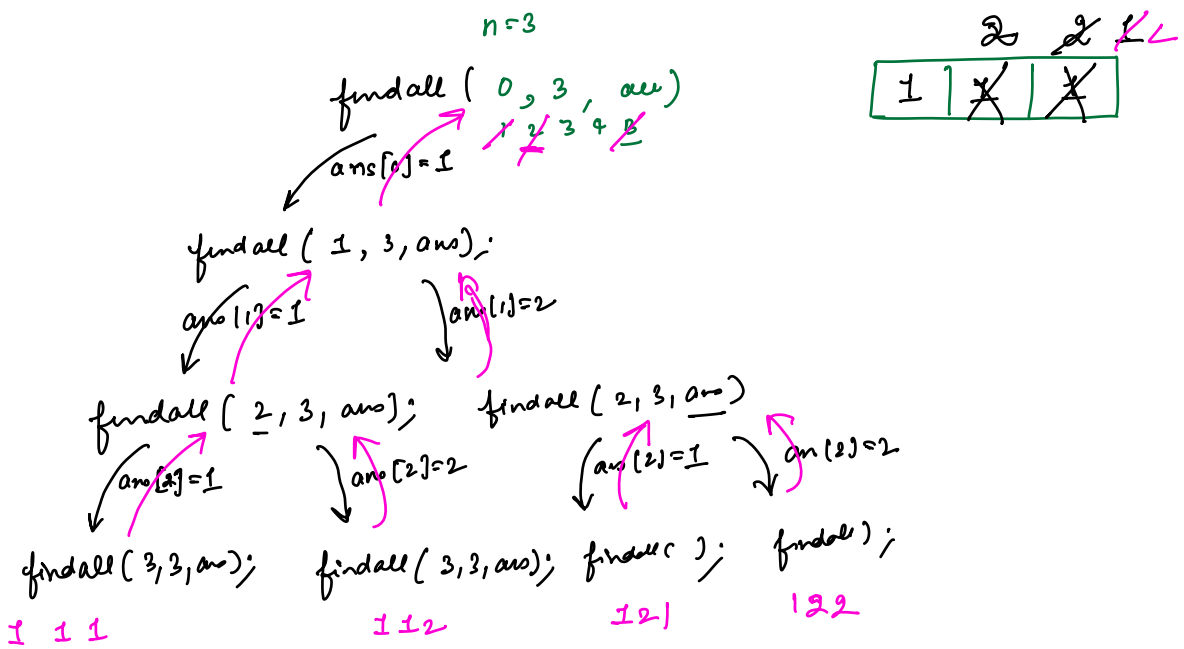
1/2 1/2 1/2





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

```
findall ( int i, int n, int ans[])
{
    if( i==n) { print(ans); return; }   (2)

    ans[i] = 1;                          (1)
    findall( i+1, n, ans);               (2)
    ans[i]= 2;                           (3)
    findall ( i+1, n, ans);              (4)
}
```



T.C: 2^n

S.C: $O(n)$

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

$T(n) = 2T(n-1) + 1$

$= 2^n$

```
fundall ( int i, int n, int ans[ ])
{
        if( i==n) {  print(ans); return; }

        for( int j=1; j<5;j++)
    { ans[i]=j;
       fundall( i+1, n, ans);
    }
}
```

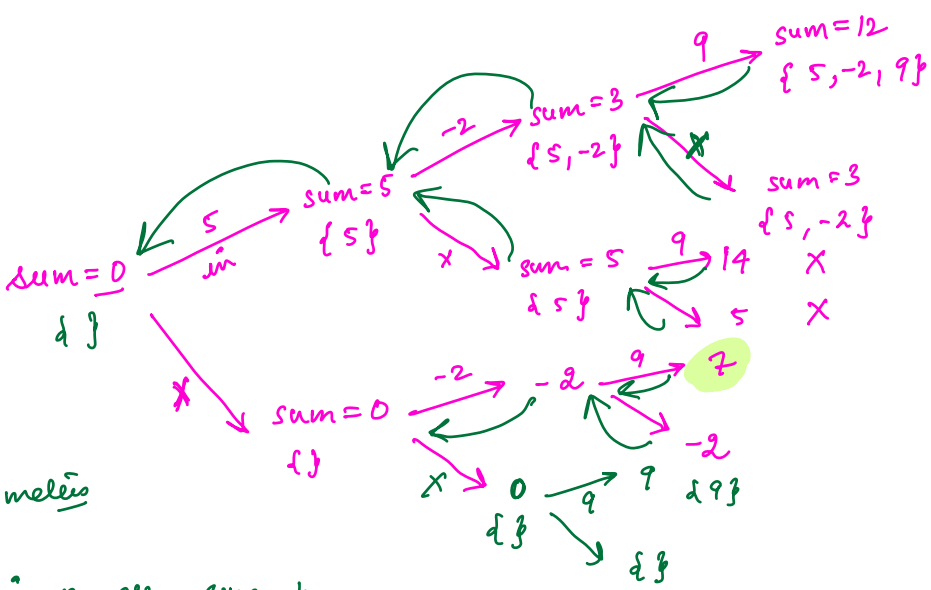
- array of size N
  → count no of subsets with sum = k.

2^n

{ 5 −2 9 }
0/1 0/1 0/1

generate all subsets

k = 7



12 == 7 ✗

3 ≠ 7

parameters

i, n, arr, sum, k

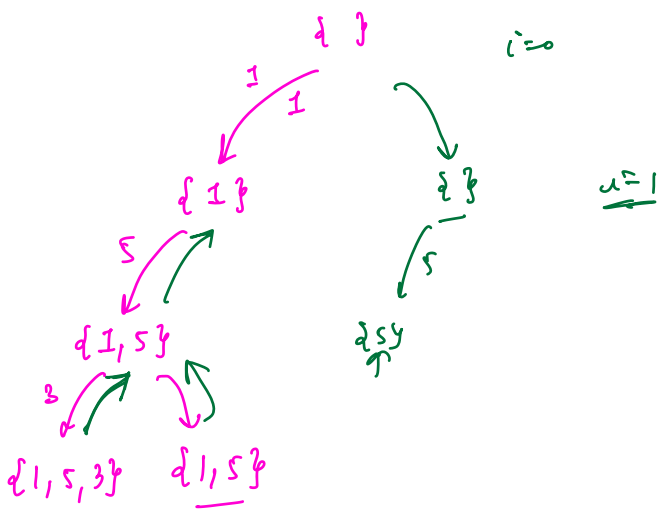
```
int findsum( int i, int n, int arr[], int sum, int k)
{
    if( i==n){ if(sum==k) return 1; }
               else return 0;

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

find all subsets if elements are unique

list of list ⇒ final list

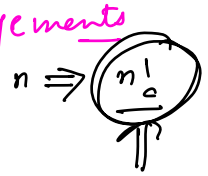
```
void genallsub( int i, int n, int arr[], list)
{
    if( i==n){  finallist.insert(list);
                return }

    list.insert( arr[i]);
    genallsub( i+1, n, arr, list);
    list.removefromend( );
    genallsub( i+1, n, arr, list);

}
```

1 5 3



0/ unique integers, find all permutations

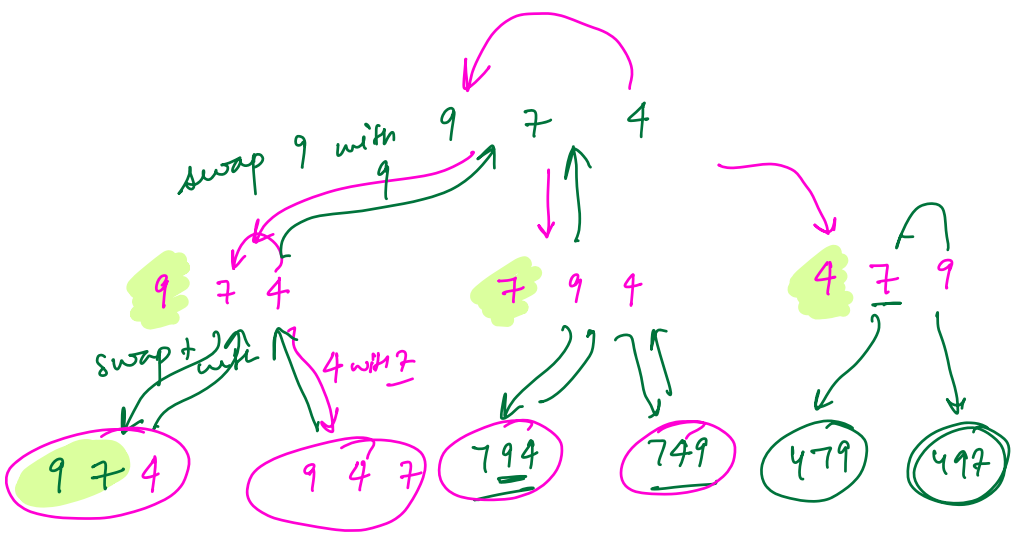
↳ rearrangements

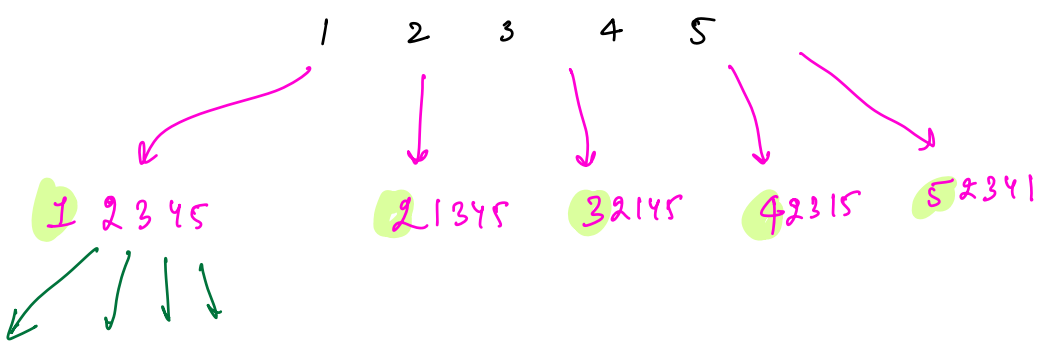


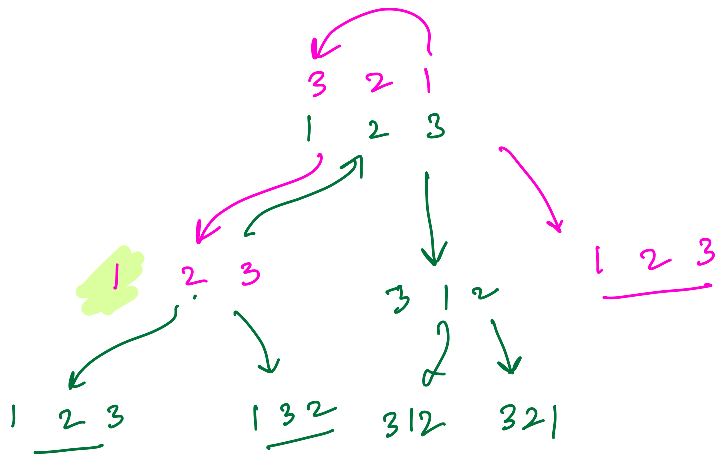
9 7 4

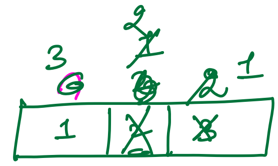
9 7 4
9 4 7
4 7 9
4 9 7
7 4 9
7 9 4

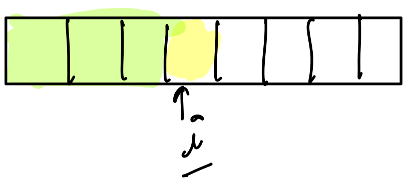
n n-1 n-2 ...





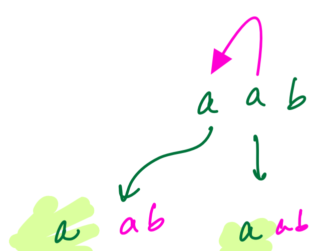







```
void  genallper( int i, int n, int arr[])
{
      if ( i==n) { finalList.insert( arr); }
                                return;

      for( j=i; j<n; j++)
      {
          if( j!=i && arr[i]==arr[j]) continue;
          swap( arr[j], arr[i]);
          genallper( i+1, n, arr);
          swap( arr[j], arr[i]);
      }
}
```

a a b



a a b
a b a
b a a

```
function (...) {
    // base condition

    Try all possibilities {
        [          ]  do
        funct()  recursive call
        [          ]  undo
    }
}
```