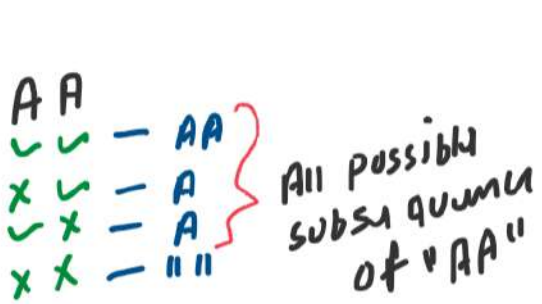
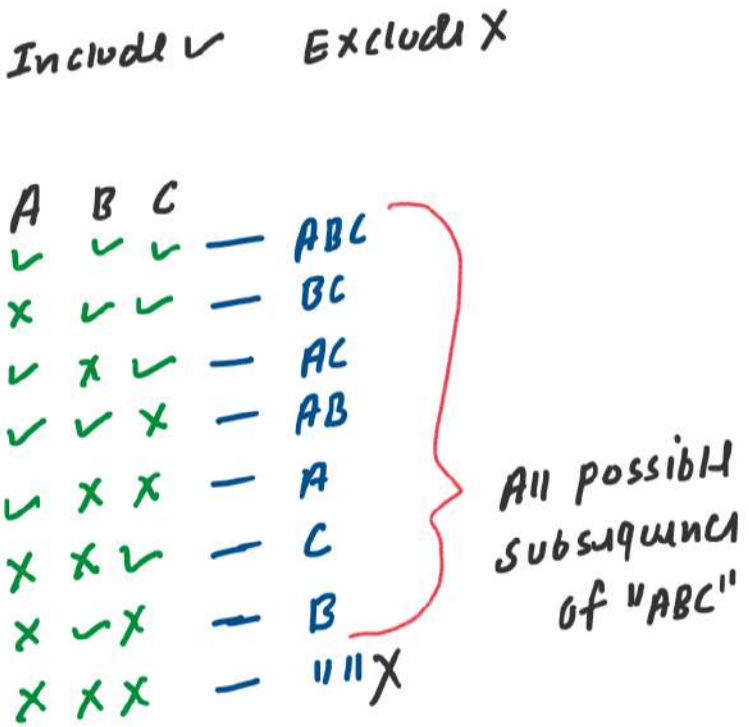
12. Power Set: Print all the possible subsequence of the String

Example 1:
Input: str = "abc"
Output: a ab abc ac b bc c
Explanation: Printing all the 7 subsequence for the string "abc".

Example 2:
Input: str = "aa"
Output: a a aa
Explanation: Printing all the 3 subsequence for the string "aa"





Logic

Include = 1 ✓ Exclude = 0 ✗

A	B	C	
1	1	1	⇒ 7
0	1	1	⇒ 3
1	0	1	⇒ 5
1	1	0	⇒ 6
1	0	0	⇒ 4
0	0	1	⇒ 1
0	1	0	⇒ 2
0	0	0	⇒ 0 ✗

powerset

$2^3 = 8$

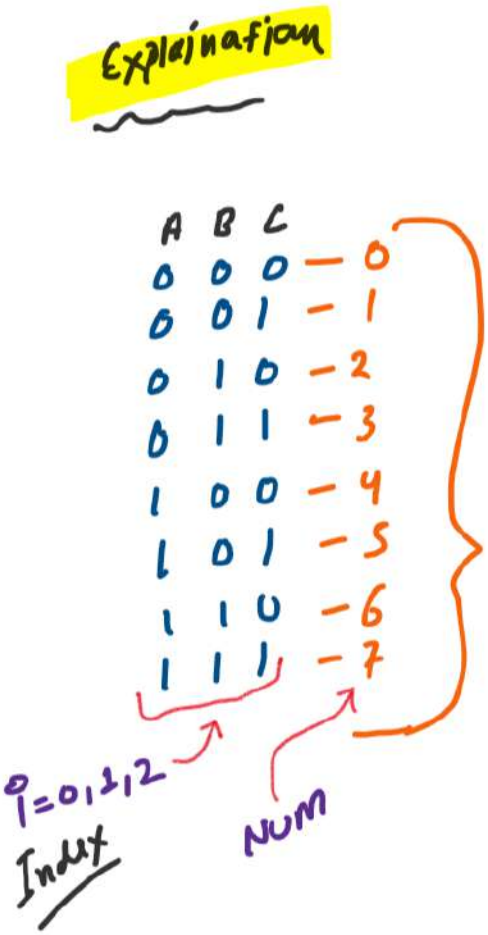
Total subsequence $= (2^n)$

where n is length of string

A	B	C		
✓	✓	✓	—	ABC
✗	✓	✓	—	BC
✓	✗	✓	—	AC
✓	✓	✗	—	AB
✓	✗	✗	—	A
✗	✗	✓	—	C
✗	✓	✗	—	B
✗	✗	✗	—	"" ✗

All possible subsequence of "ABC"

Explaination



STEP1 Traverse 2^n time

```
for( 0 to 2^n - 1) {
    for( 0 to N-1) {
        if( NUM & (1 << i))
            → creat sub string
    }
    print substring
}
```

STEP2 Traverse for each num for N time

STEP2 Check the set bit for each ith index

1 - set
0 - clear

```cpp
// 12. Power Set: Print all the possible subsequences of the String

#include<iostream>
#include<vector>

using namespace std;

void getSubsequences(string &str){
    int n = str.length();
    vector<string> ans;

    // Step 1: traverse for 2^n times
    int powerSet = 1 << n;
    for(int num = 0; num < powerSet; num++){
        string substring = "";

        // Step 2: traverse for each num for n times
        for(int index = 0; index < n; index++){
            char ch = str[index];

            // Step 3: check set bit for each index in num
            int mask = 1<<index;
            if(num & mask){
                // Create a substring
                substring.push_back(ch);
            }
        }
        // Step 4: Store the substring to the vector
        ans.push_back(substring);
    }

    cout << "Printing the subsequences: ";
    for(auto substr: ans){
        cout << substr << " ";
    }
}

int main(){
    string str = "abc";
    getSubsequences(str);
    return 0;
}
```

TRY TO solve leetcode - 78 subsets

- Time complexity for outer loop
  $O(2^N)$
- Time complexity for inner loop
  $O(N)$

Overall Time complexity $\Rightarrow O(2^N * N)$

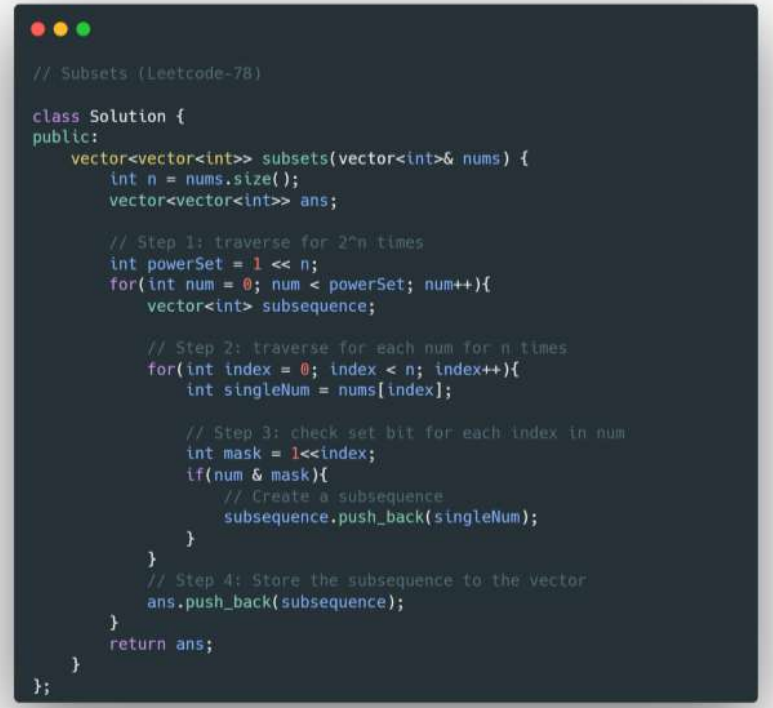
Space complexity $\Rightarrow O(2^N * N)$

Subsets (Leetcode-78)

Given an integer array nums of unique elements, return all possible subsets (the power set).
The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:
Input: nums = [1,2,3]
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]

Example 2:
Input: nums = [0]
Output: [[],[0]]



```cpp
// Subsets (Leetcode-78)

class Solution {
public:
    vector<vector<int>> subsets(vector<int>& nums) {
        int n = nums.size();
        vector<vector<int>> ans;

        // Step 1: traverse for 2^n times
        int powerSet = 1 << n;
        for(int num = 0; num < powerSet; num++){
            vector<int> subsequence;

            // Step 2: traverse for each num for n times
            for(int index = 0; index < n; index++){
                int singleNum = nums[index];

                // Step 3: check set bit for each index in num
                int mask = 1<<index;
                if(num & mask){
                    // Create a subsequence
                    subsequence.push_back(singleNum);
                }
            }
            // Step 4: Store the subsequence to the vector
            ans.push_back(subsequence);
        }
        return ans;
    }
};
```