Todays Content

1. Subsets/Subsequences
2. Check if there exists a subsequence with sum=0

#Subarray: Continous part of an array.

#Subsequence: Take any element in arr[].
Arrage them in increasing order of index

```
              0  1  2  3  4   5
Ex: arr[] = { 7  2  6  9  10  8 }
```

Subsequence:

{ 2 9 10 8 } #Subsequence { 7 6 2 10 } #Not subsequence

{ 7 6 9 8 } #Subsequence

{ 2 7 10 8 } #Not subsequence

Subset: Same as sequence No need to maintain order
We identify purely based on data it has

```
              0  1  2  3  4   5
Ex: arr[] = { 7  2  6  9  10  8 }
```

Subset

{ 2 9 10 8 } → { 2 8 9 10 } # Both are same subsets.

Note: An { } sequence/set is considered valid

```
          0  1  2
# arr[] = { 3  2  9 }
```

All Subsets: #8 subsets = 2^3	All Subsequence #8 subsequence = 2^3
{ }	{ }
{3} {2} {9}	{3} {2} {9}
{3 2} {9 2} {3 9}	{3 2} {2 9} {3 9}
{3 2 9}	{3 2 9}

	#Continous	#Order Index	#Count
Subarray	✓	✓	(N)(N+1)/2
Subsequence	*	✓	2^N
Subset	*	*	2^N

Given an arr[N] check if there exists a subset with sum = k.

Note: Cannot use any kind of extra space. ↳ Any elements/Need not be continous

Note: # Empty set is allowed.

Ex:

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ arr[\,] = \{ & 2 & -3 & 6 & 11 & 4 & -5 & 6\,\} \end{array}$$

k = 14 : { 2 6 6 } ✓ return True;

k = 16 : { 6 4 6 } ✓ return True;

Constraints

$1 <= N <= 20;$

$-10^6 <= arr[i] <= 10^6$

Idea:

```
i=0; i<N; i++){ *
    j=i+1; j<N; j++){
        if(arr[i]+arr[j]==k){
        }          ==
    }
}
```

Idea2: Generate all subset sums & compare == k.

$$\begin{array}{cccc} & 0 & 1 & 2 \\ Ex: arr[\,] = \{ & 2 & 3 & -6\,\} \end{array}$$

i:	2	1	0	
0 :	0	0	0	{ }
1 :	0	0	1	{2}
2 :	0	1	0	{3}
3 :	0	1	1	{2 3}
4 :	1	0	0	{-6}
5 :	1	0	1	{2 -6}
6 :	1	1	0	{3 -6}
7 :	1	1	1	{2 3 -6}

#obs: N = 3 # Subsets = 2^3 = 8

Bits: Each arr[i] mapped to bit

Numbers [2 1 0] : 3 arr[] elements hence 3 bits

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ 7 \end{bmatrix}$$

0 1 2 3

Eg: $arr[] = \{2 \quad 3 \quad -6 \quad 3\}$

#obs: $N = 4$ # Subsets $= 2^4 = 16$

Bits:

Numbers [3 2 1 0]

0
1
2
⋮
15

#Numbers		3	2	1	0		Sum
0	:	0	0	0	0	{ }	0
1	:	0	0	0	1	{arr[0]}	2
2	:	0	0	1	0	arr[1]	3
3	:	0	0	1	1	arr[0] + arr[1]	5
4	:	0	1	0	0	arr[2]	-6
5	:	0	1	0	1	arr[0] + arr[2]	-4
6	:	0	1	1	0	arr[1] + arr[2]	-3
7	:	0	1	1	1	arr[0] + arr[1] + arr[2] =	-1
⋮							
15	:	TODO					

#Generalize

Given arr[N] # SubSet $= 2^N$

#Numbers #bits

N-1 .... 2 1 0

0
1
2
⋮
2^N-1

```
For every number from [0.. 2^N-1]
   Generate it's bit's from [0.. N-1] &
   Map it to a subset & get it sum.

   if( sum == Target){
      return True;
   }
}
return False;
```

Constraints

$1 <= N <= 20;$

$-10^6 <= arr[i] <= 10^6$

```
boolean checkSum(vector<int> &arr, int k){ TC: O(2^N * N)  SC: O(1)
    int N = arr.size();
    for(int i=0; i<2^N; i++){
        # i : Generate N bits & map with subset & get sum;
        int sum=0;
        for( int j=0; j<N; j++){
            if( (i>>j)&1 ==1){ # j^th bit set => Consider arr[j] in Subset
                sum = sum + arr[j];
            }
        }
        # i : We have it's represective subset sum.
        if( sum==k){
            return true;
        }
    }
    return false;
}
```

28 Given ar[N] it contains all elements from 1.. N.

# 1 element from 1 to N repeats

# 1 element from 1 to N missing

Return both repeat & missing element

Note: No Extra space, No modifying array.

Constraints:

$1 <= N <= 10^6$

$1 <= ar[i] <= N.$

Ex: missing repeat

ar[5] = { 2 2 1 4 5 }

ar[7] = { 1 3 6 5 4 6 7 }

Idea: