AGENDA

Recap Subarray is a continuous part of the array.

Subsequence → A sequence generated by deleting 0 or more elements of the array.

arr[8] : -2 -3 6 2 4 -1 0 3

X	X	✓	X	✓	X	X	✓	[6, 4, 3]
X	X	X	X	✓	✓	✓	X	[4, -1, 0]
✓	✓	✓	✓	✓	✓	✓	✓	[-2 -3 6 2 4 -1 0 3]
X	X	X	X	X	X	X	X	[ ] empty subsequence

In a subsequence, elements have to be arranged based on their index value

— In other words, order in subsequence should be maintained.

arr[5]: 3 -1 0 6 2 (indices 0 1 2 3 4)

✓	X	✓	✓	X

→ [3 0 6] (indices 0 2 3) ✓

⇓

[0 6 3] (indices 2 3 0) X

subarray

continuous part of the array

subsequence

sequence generated by deleting 0 or more elements from the array.

can you comment something about them?

a[]: {-1, 4, 3, 9}

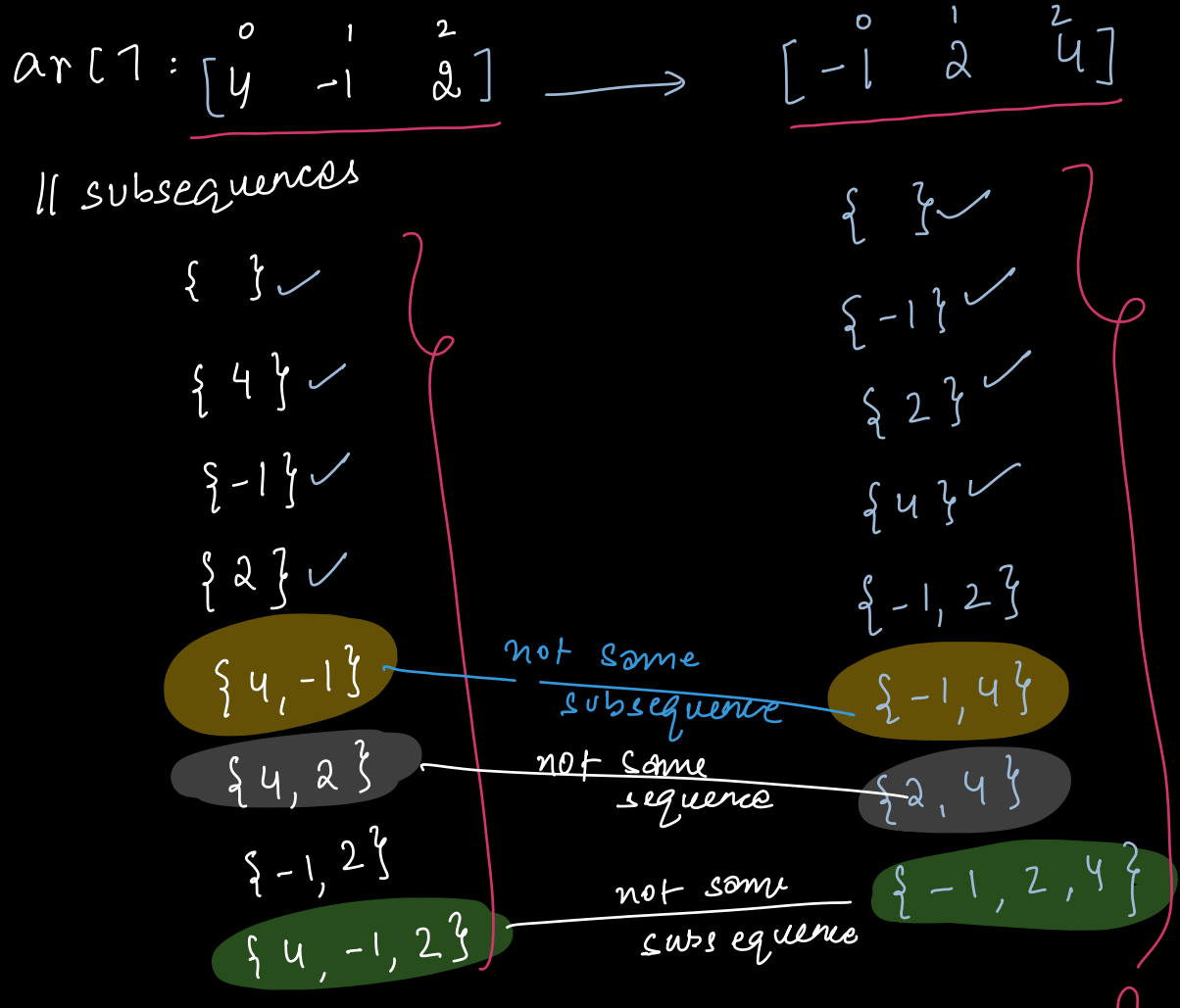
	subarray	subsequence
{-1, 4}	✓	✓
{4, 3, 9}	✓	✓
{4, 9}	✗	✓
{-1, 3, 9}	✗	✓
{3}	✓	✓

Observation 1

Take any subarray, and will also be a subsequence

Observation 2

All subsequences are not a subarray



subsequences of the originally given array & sorted array may not be same

Subsets : same as subsequence but the order does not matter.

ar[ ] : [4 (0) -1 (1) 2 (2)] —sort→ [-1 2 4]

subsets

{ }
{4}
{-1}
{2}
{4, -1}
{4, 2}
{-1, 2}
{4, -1, 2}

subsets

{ }
{-1}
{2}
{4}
{-1, 2}
{2, 4}
{-1, 4}
{-1, 2, 4}

subsets of given array are same as subsets of the sorted array.

Total no. of subsequences / subsets.

A[5]: a_0 a_1 a_2 a_3 a_4

$2 * 2 * 2 * 2 * 2 = 2^5$

Generalise it for n

$\hookrightarrow 2^n$

[2 -5] → (2)

Y Y

N N

{2 -5}

{2}

{-5}

{ }

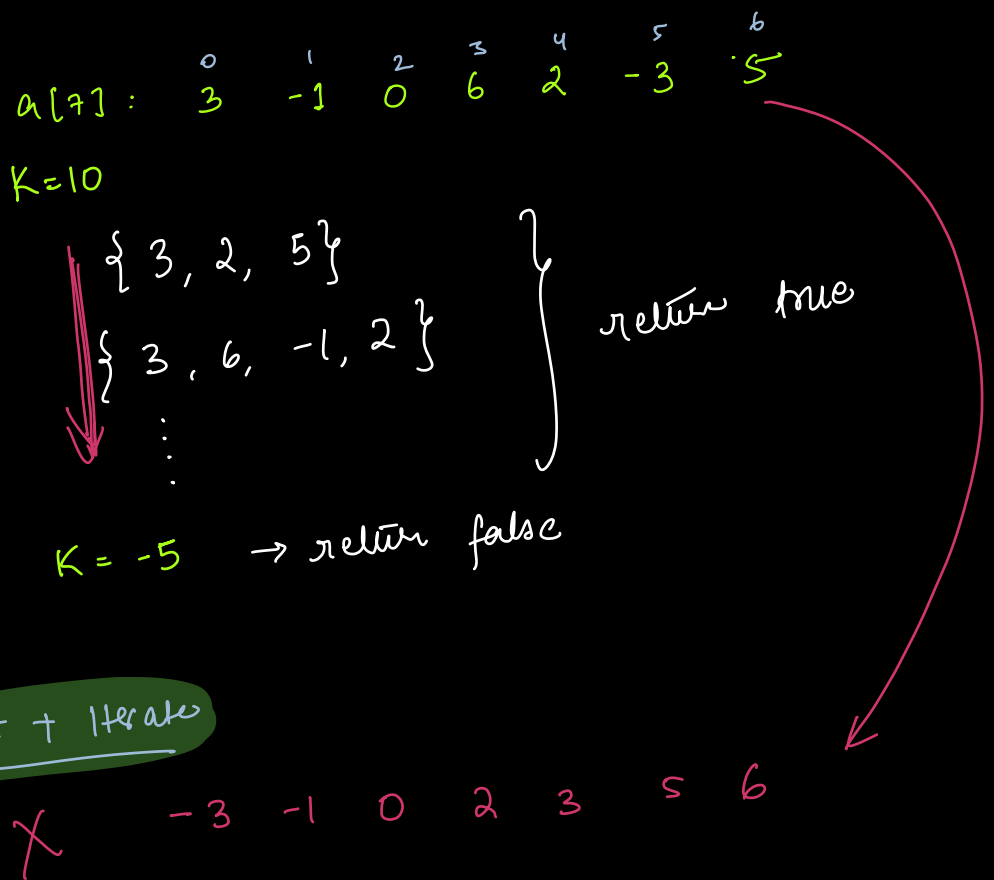
$2^2 = (4)$

no. of subsets [empty subset not included]

⇓

$2^n - 1$

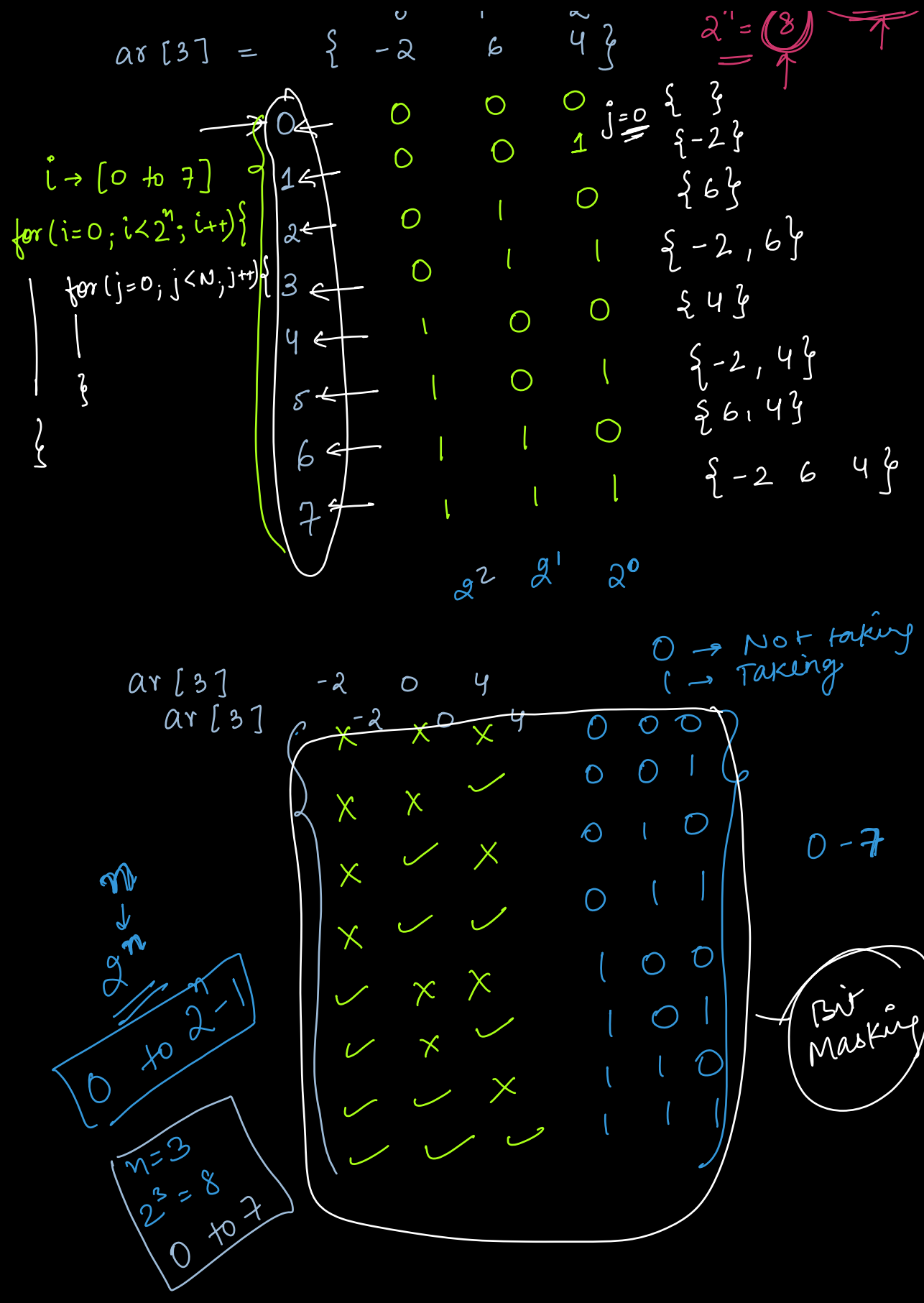
Q1. Given N array elements, check if there exists a subset with sum = K



Simple Approach

Generate all the setcets
For each subset,
check if sum of ele == K or not

Q. How to generate all the subsets?


ar[3] = { -2 6 4 }
2^n = (8)
i → [0 to 7]
for(i=0; i<2^n; i++){
for(j=0; j<N; j++){
}
}
0 1 2 3 4 5 6 7
0 0 0 j=0 { }
0 0 1 {-2}
0 1 0 {6}
0 1 1 {-2, 6}
1 0 0 {4}
1 0 1 {-2, 4}
1 1 0 {6, 4}
1 1 1 {-2 6 4}
2^2 2^1 2^0
ar[3] -2 0 4
ar[3]
0 → Not taking
1 → Taking
0-7
n
2^n
0 to 2^n - 1
n=3
2^3 = 8
0 to 7
Bit Masking

Pseudocode

for given N [total array elements]

i : [0 to $2^n - 1$]

for (i = 0 ; i < 2^n ; i++) { → $(1 << n)$ → 1×2^n

sum = 0

for (j = 0 ; j < n ; j++) {

if (checkBit (i, j)) {

sum += ar[j]

}

}

if (sum == k) return true

}

return false .

i = 0, 1, 2, ..., 2^n

Subset with sum k exists or not

BIT MANIP

TC

$O(2^n * n)$

SC → O(1)

$n \leq 20$

n = 3

ar [3] [-2 0 4]

k = 2

$(1 << n)$ → O(1)

2^n → $1 * 2^n$

exponential

101

i	j	sum
0	0	0
	1	
	2	
1	0	0
	1	-2
	2	
2	0	0
	1	0
	2	

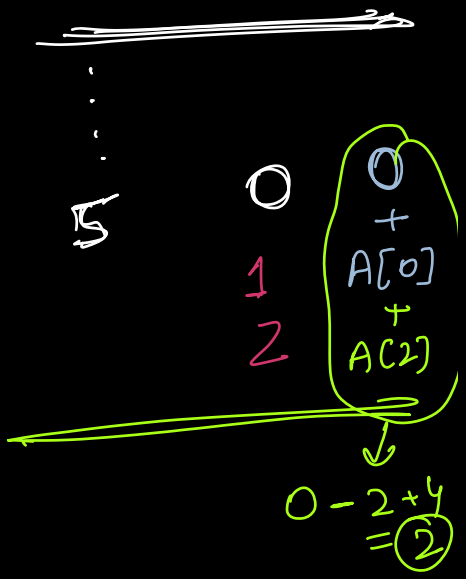
TC

Bit manipulation → $O(2^n * n)$

Backtracking → $O(2^n)$

DP → $O(n * K)$ (n: no. of ele of array, K: sum)

→ advanced classes.



Q2. Given all array elements, find **sum of all subsets sum**

Sol 1

In your prev ques → sum of all subsets & ~~compared with~~ K

sum these up.

TC → $O(2^n * n)$

Sol 2 Contribution Technique →

sum of all subarr sums

⇓

contr. of each ele in the subarray

⇓

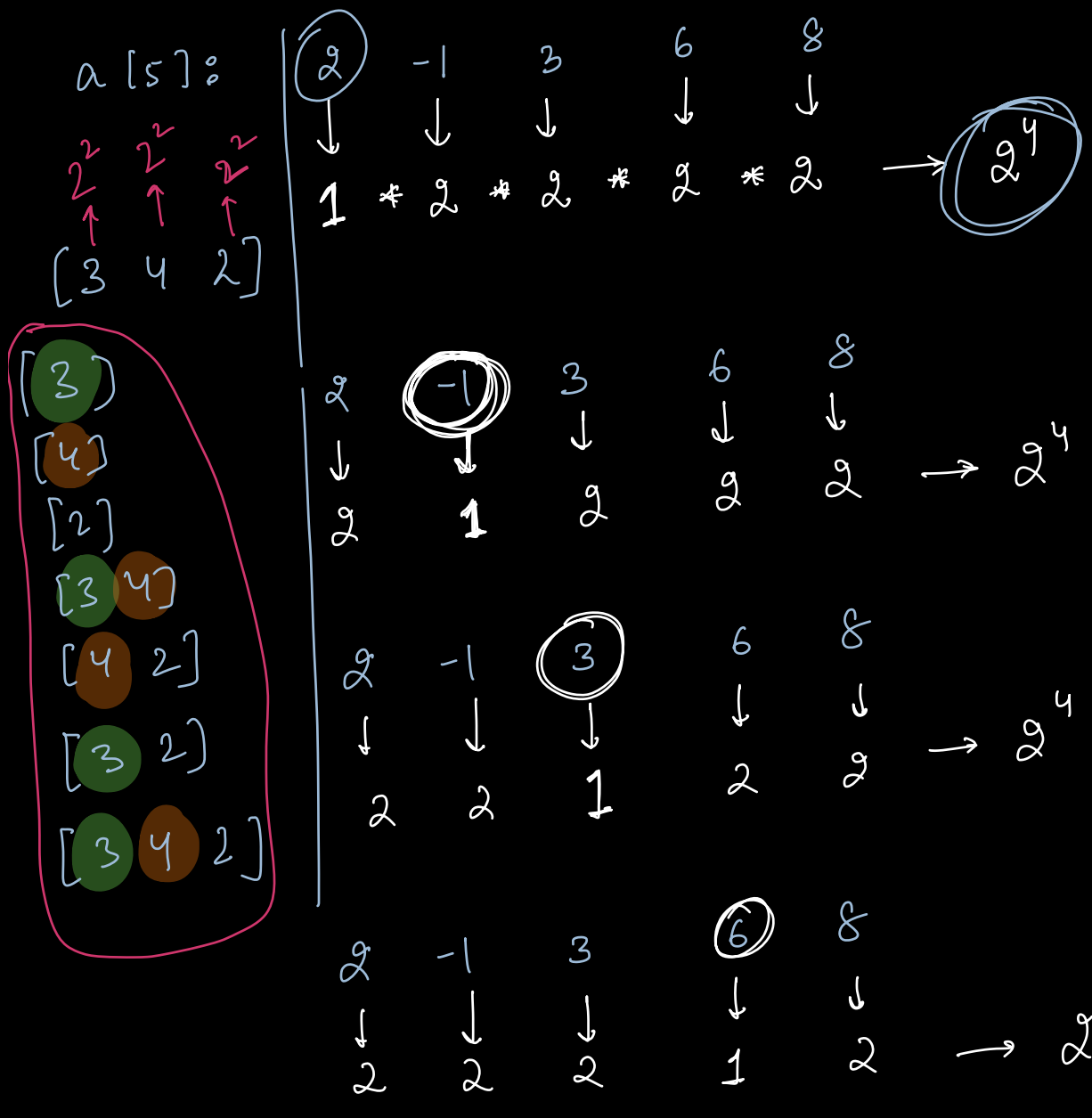
A[i] × { No. of times an ele is appearing in a subarray }

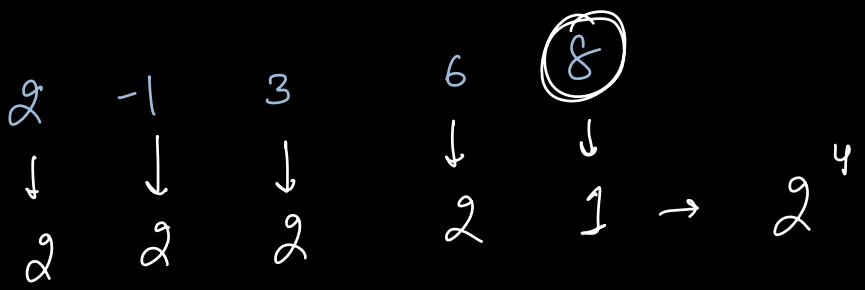
sum of all subsets sum

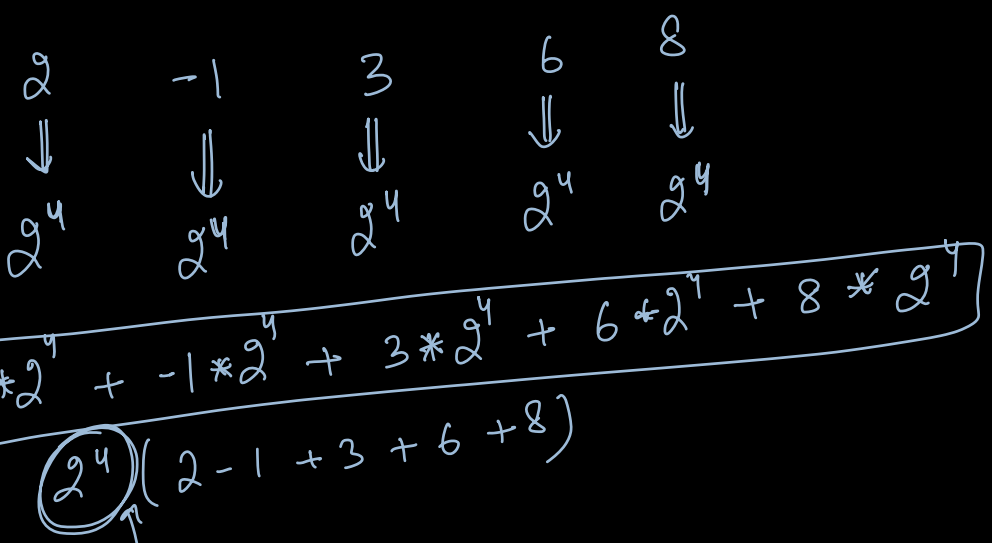
⇓

For each ele, find that in how many subsets it exists.

How can we find in how many subsets an ele appears.







Generalize for n

⇓

Each ele will appear 2^{n-1} times

```
sum = 0
for (i=0; i<N; i++) {
    sum = sum + (ar[i] * 2^(n-1))
}
return sum.
```

⇓

```
ar[i] * (1 << n-1)
```

TC → O(n)

SC → O(1)

Interview.

Given N array ele,

calc (sum of all subsets sum) / 2^N

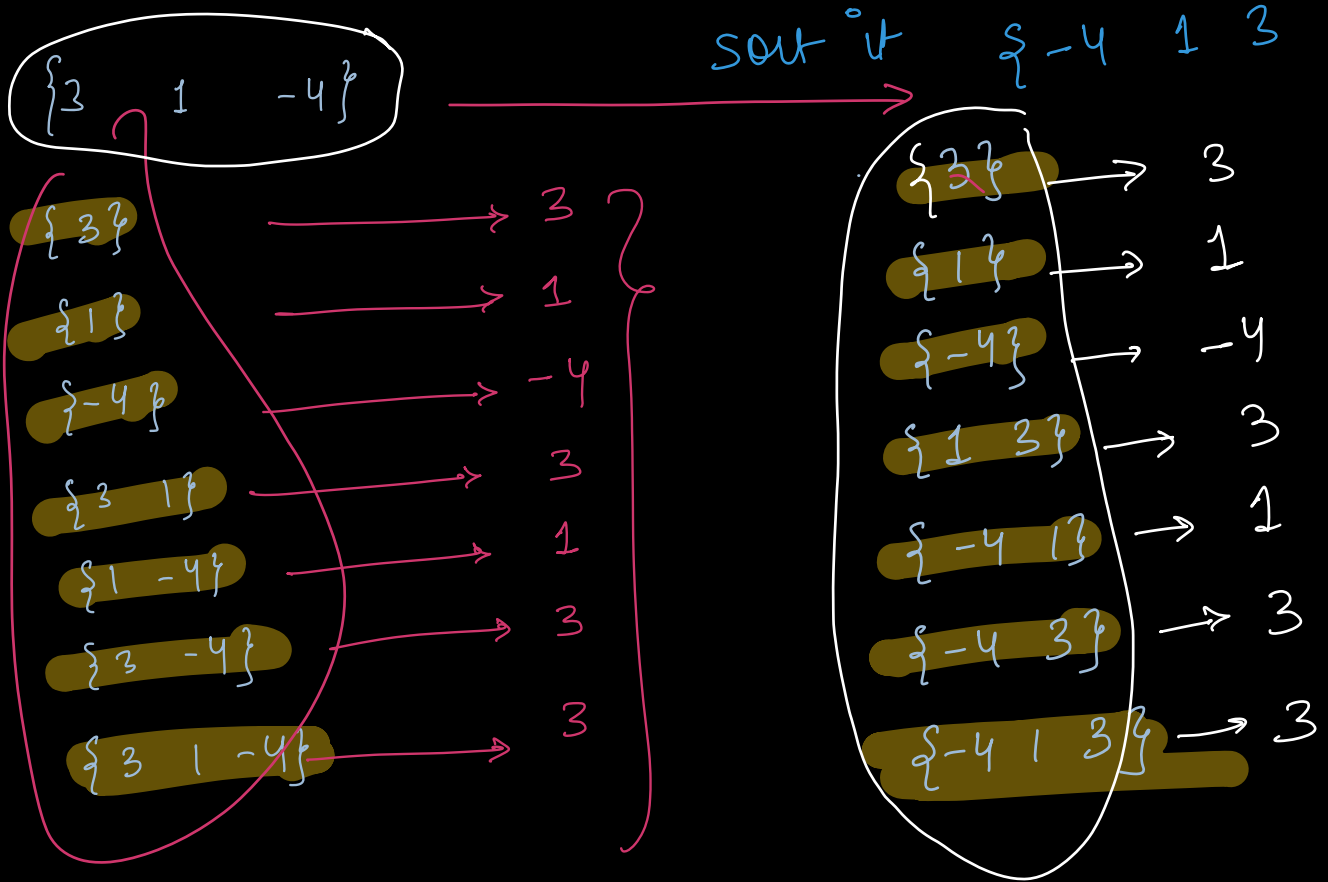
↓ use power function [if mod needs to be applied]

$$\frac{(\text{sum of all elements of array}) \times 2^{N-1}}{2^N}$$

$$(\text{sum of all elements of array}) \times 2^{N-1-N}$$

$$\Rightarrow \frac{\text{sum of all array ele}}{2}$$

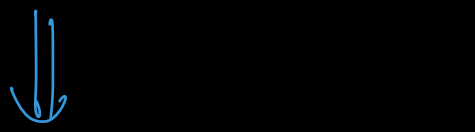
Q: Given N array ele, calc sum of MAX of every subsequences -

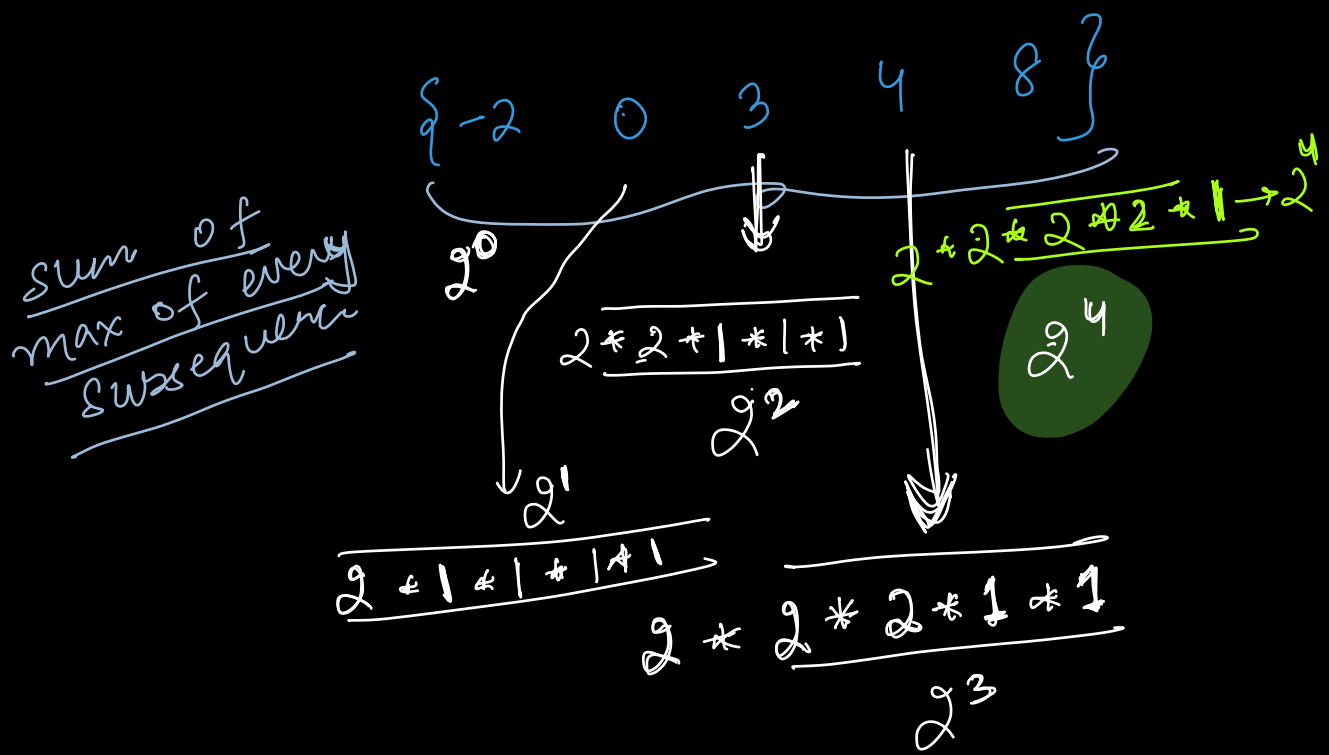


MIN, MAX, SUM, PROD values will be same?

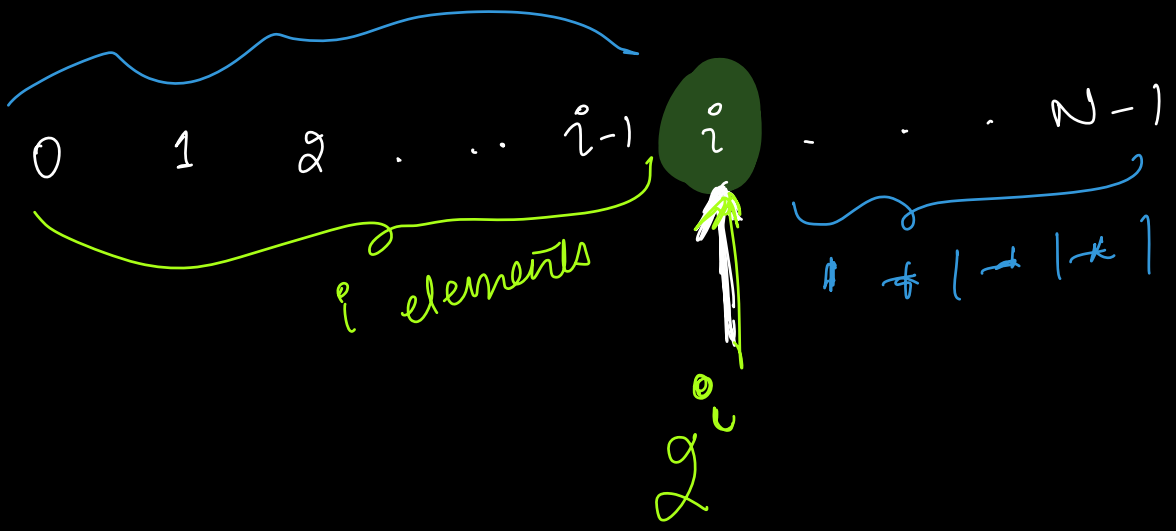
Even for subsequences we can sort array depending on the question.

Ex {-2 8 0 4 3}

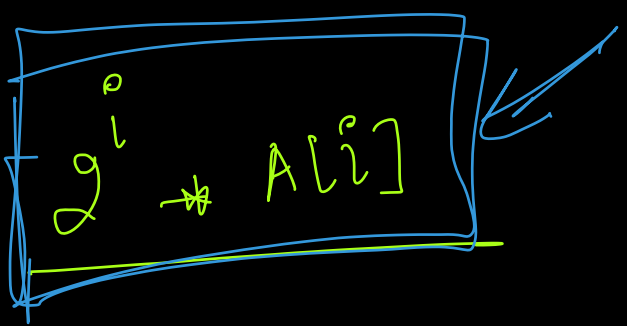




-2	0	3	4	8
2^0	2^1	2^2	2^3	2^4



① Sort the array



TC
O(n log n)
T
O(n)

② sum = 0

```
for (i=0; i<N; i++) {
    sum = sum + (2^i * A[i]);
}
```

SC → O(1)

Similarly, sum of MIN of all subsequences.

[4 1 2]

[4] → 4

[1] → 1

[2] → 2

[4, 1] → 4+1

[4, 2] → 4+2

[1, 2] → 1+2

[4, 1, 2] → 4+1+2