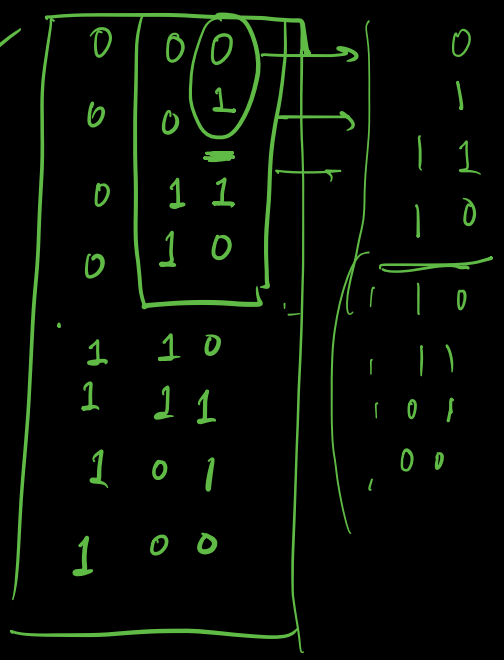
[3, -1, 2, 6]

List / Array

Gray code



Subarrays -) contiguous segment of an array.

Subsequences	Subset

You can delete 0 or more elements from the array and the sequence of elements that you get is called a subsequence.

[2, -3, 5, 6, 7, 10]
X ✓ X X ✓ ✓

ORDER MATTERS ↓ should be same as the original array.

= [-3, 7, 10] → a subsequence

Subsets the order doesn't matter !

[-3, 7, 1] is a subsequence of [4, 7, 1, -3, 4]

NO!

" is a subset of "

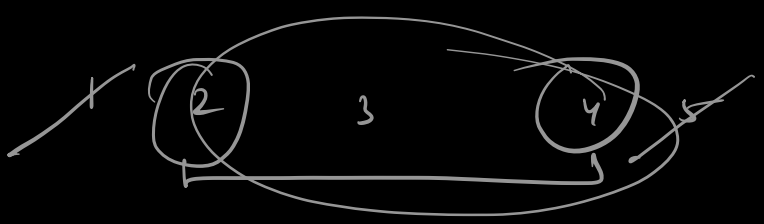
Yes

DNA sequence ATGC

String matching algorithm

LCS :)

Subsequences → Deleting 0 or more elements



① All subsequences are subsets

② All subsets are subsequences

Sort the array
and generate subsequences
⇒ subsets of original array

[1, 2, 3] ⇒ (8)

all Subsequences

[ ]
[1]
[2]
[3]
[1, 2]
[1, 3]
[2, 3]
[1, 2, 3]

[ ]
[1]
[2]
[3]
[2, 1]
[3, 1]
[2, 3]
[3, 1, 2]

(1, 2)

[1, 1, 2]
0 1 2

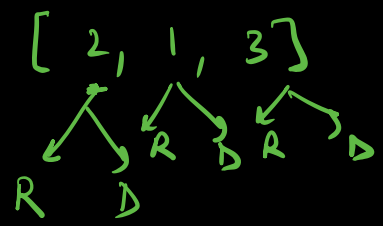
→

[ ]
[1]
[1]
[2]
[1, 1]
[1, 2]
[2, 1]
[1, 1, 2]

[ ]
[1]
[2]
[1, 2]
[1, 1]
[1, 1, 2]

Q1

How many subsequences of an array of size N. ?

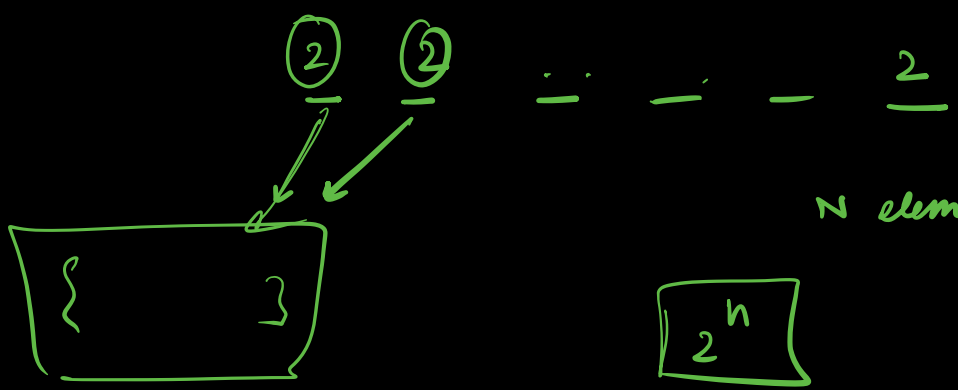


8 subsequences

Logic 1

Logic 2

16 options



N elements

2^n

Logic 3

Q 2 Given N elements. Check if there exists a subsequence which has sum of its elements as (K).

arr = [2, 1, 3, -1, 4]

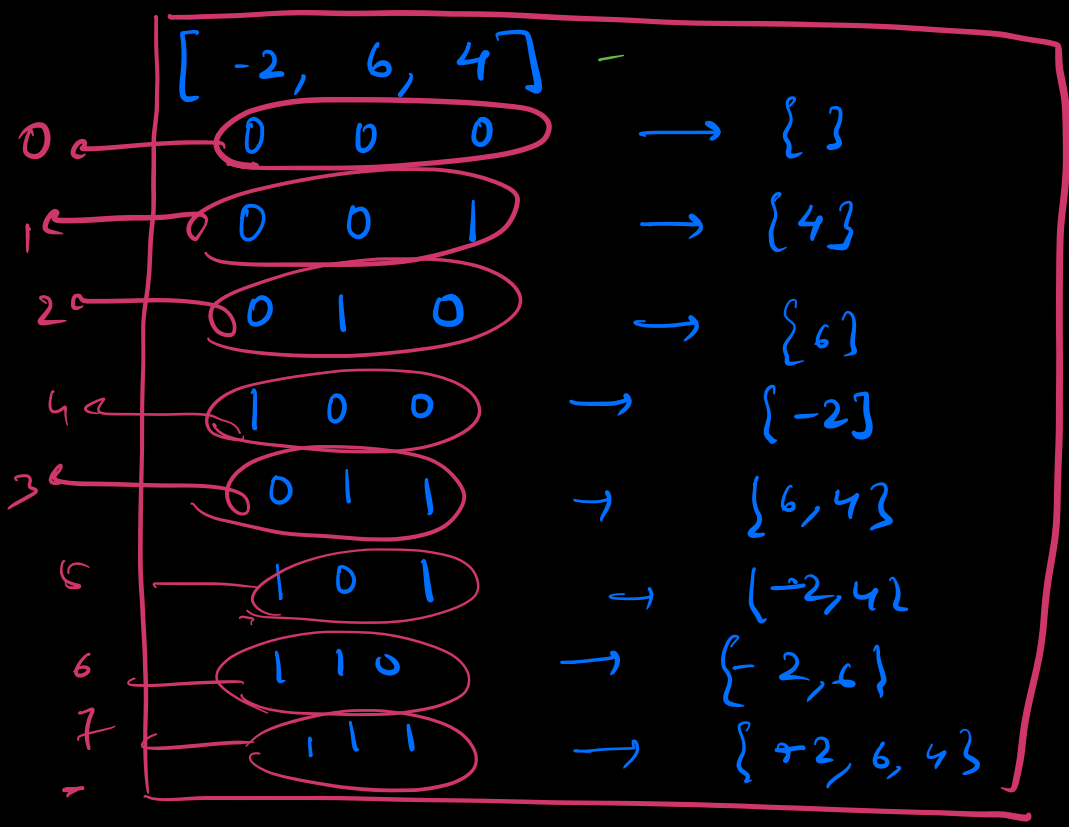
K = 0 ⇒ [1, -1] ⇒ True

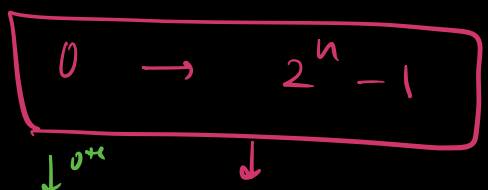
arr = [3, -1, 0, 6, 2, -3, 5]

K = 10 True ✓

B.F
Generate all possible subsequences
↳ Check the sum for all subsequence == K

TC $2^N * N$





n = 3 i = [0, ... 7]

```
for i in range (2**n):
    sum = 0
    for j in range (N):
        if i & (1<<j) != 0:
            sum += arr[j]
    if sum == k:
        return True
```

Tracks the sum of a subsequence

Check if a no. is a part of

subsequence i=

0, 1, 2 — (i)

0 & (1<<0)

0 & (1<<1) ✗

0 & (1<<2) ✗

0 → sum = 0 {3}

1 → sum = 0 {4}

2 →

3

4

5 → 101

6

7

1 & 1 = = 1

0 0 (1)

5 & (1<<0)

5 & (1<<2)

[-2, 6, 4]

sum = 2

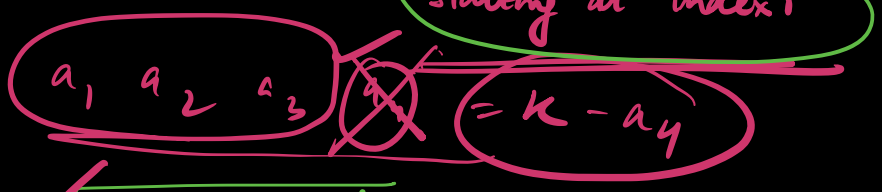
2.

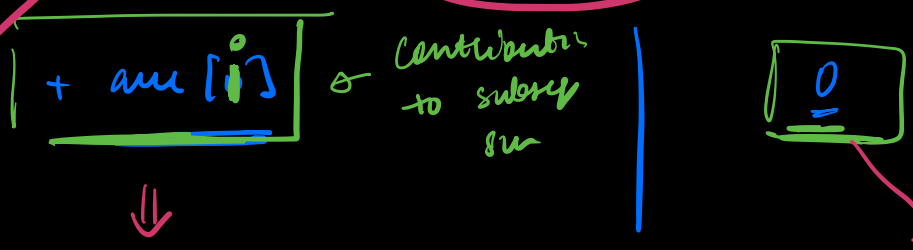
6 + 7 + (-3) 10

k<0 ✗

check (arr, k, i)

⇒ It return True if there exists a subsequence with sum == k in the arr starting at index i otherwise return false





check (arr, k - arr[i], i+1)

check (arr, k, i+1)

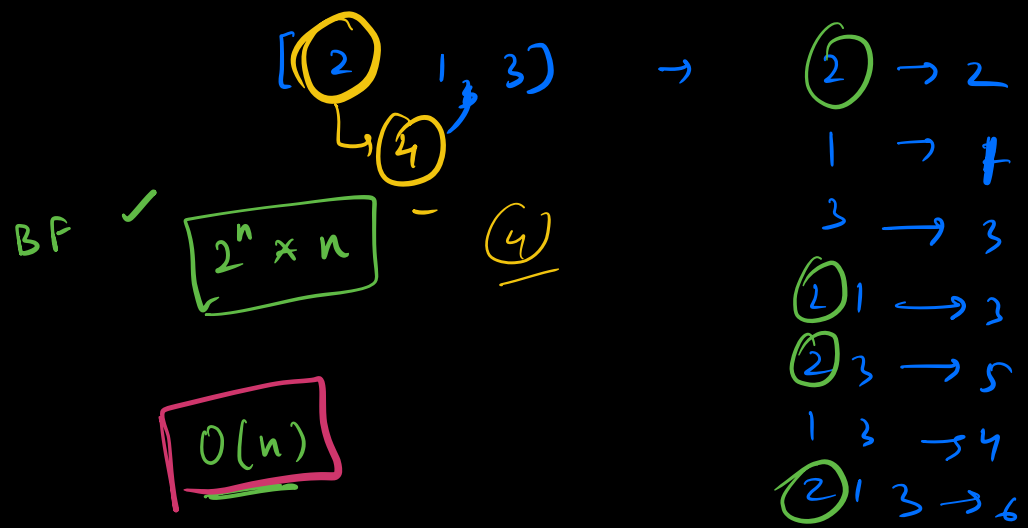
```
def check (arr, k, i):
    if i == len(arr):
        if k == 0:
            return True
        return False
    if check(arr, k-arr[i], i+1) or
            check(arr, k, i+1):
        return True
    return False
```

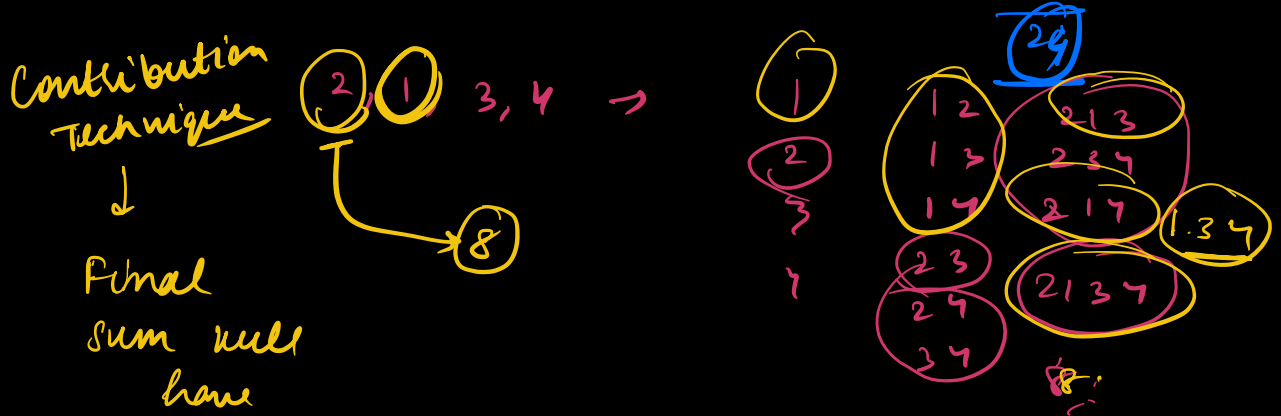
Base!

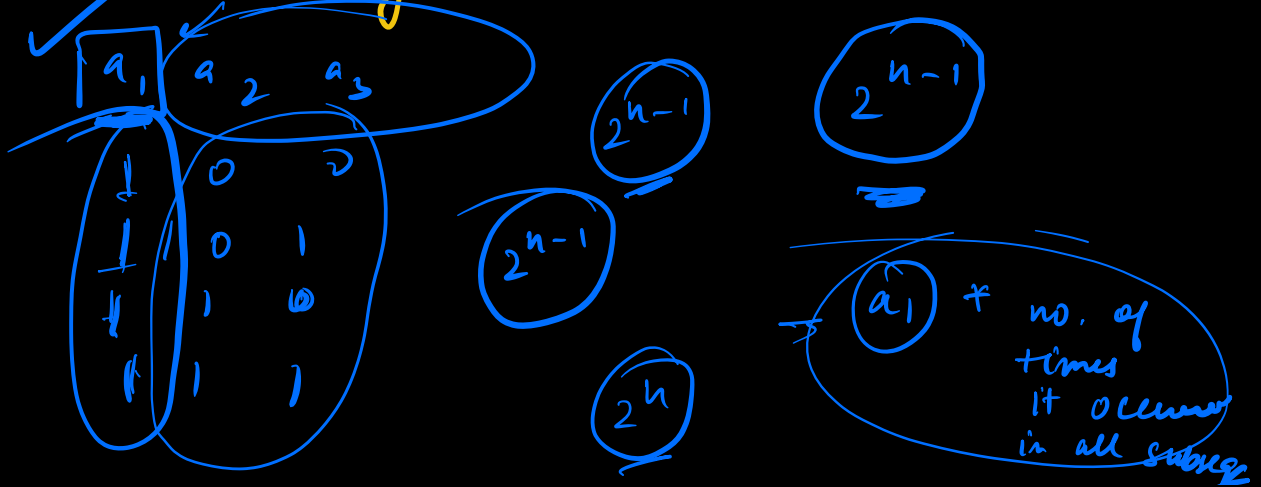
$O(2^n)$

Break - 5 mins.

Q Find sum of all the subsequences.







$2^n / 2 = 2^{n-1}$

```
ans = 0
for i in range(N):
    ans += (arr[i] * 2^(n-1))
return ans
```

Q Given N array elements. Find the sum of max of every subsequence / min?

$2^n * n$ Bf ✓

[3, -1, 4]

3
-1
4
3 -1
3 4
-1 4
3 -1 4

3 3 -1 4 4 4 = 24

3 ✓

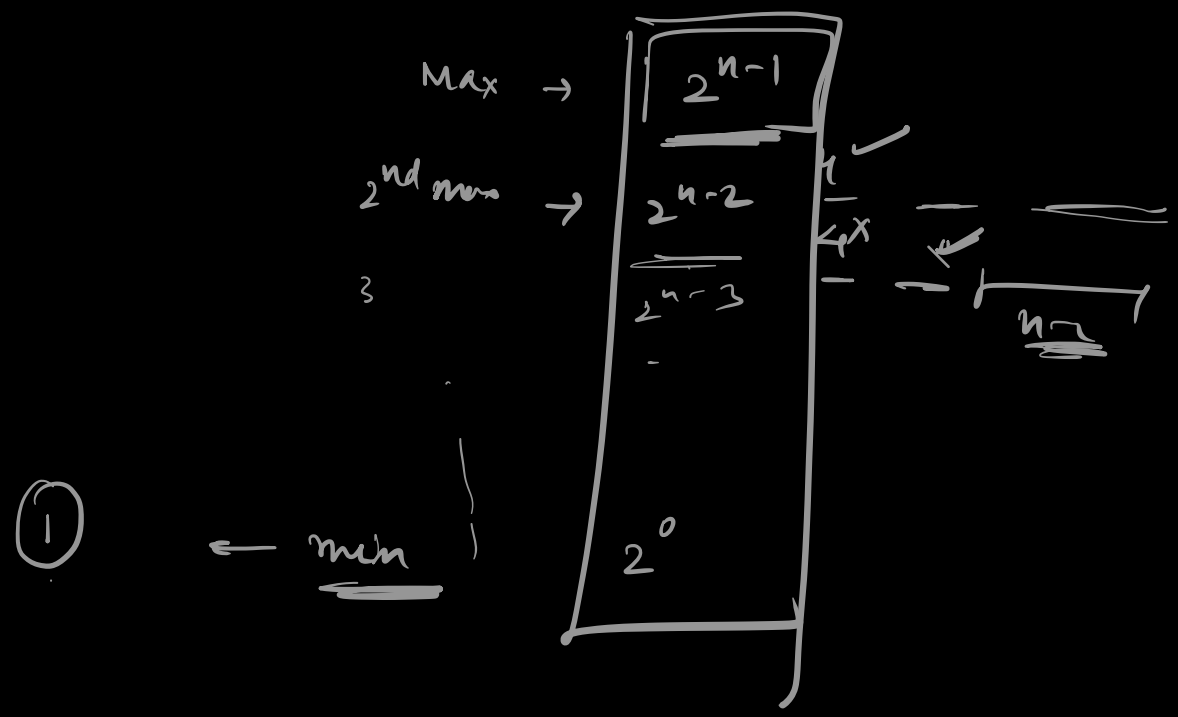
n-2

2^{n-2}

```
① a.sort()
② for i in range(n):
      sum += 2^i * arr[i]
```

TC = n log n

① ——— arr[0] → min



Q Find sum of $(max - min)$ of every subsequence.

$[-1, 3, 4]$

$\sum (max_i - min_i)$

$\sum max_i - \sum min_i$

$O(n \log n)$

	max	min
{} →	0	0
{-1}	-1	-1
{3}	3	3
{4}	4	4
{-1, 3}	3	-1
{-1, 4}	4	-1
{3, 4}	4	3
{-1, 3, 4}	4	-1

0 + 0 + 0 + 0 + 4 + 5 + 1 + 5 = 15

(*) OOPS → Class

↳ Read Quick

↳ Pillars of OOPs