Lecture 18 - Subsequence & subsets

Subarrays : contiguous segment of an array

Non-contiguous segments :

subsequence ← depends on the order → sets

Ex.

$[\ 2, -3, 5, 6, 7, 10\ ]$

$-3, 6, 7 \longrightarrow$ subsequence

$-3, 7, 6 \longrightarrow$ subset

1. All subarrays are subsequences
2. All subsequences are subsets

Q. How many subsequences for an array of N elements ? $\Rightarrow$ 2^N

(72) 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

Brute Force

[-2, 6, 4]

0	0	0	–	[]
0	0	1	–	[4]
0	1	0		[6]
1	0	0		[-2]
0	1	1		[6, 4]
1	0	1		[-2, 4]
1	1	0		[-2, 6]
1	1	1		[-2, 6, 4]

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

Ex. [-2, 6, 4] K = 2

[-2, 4]

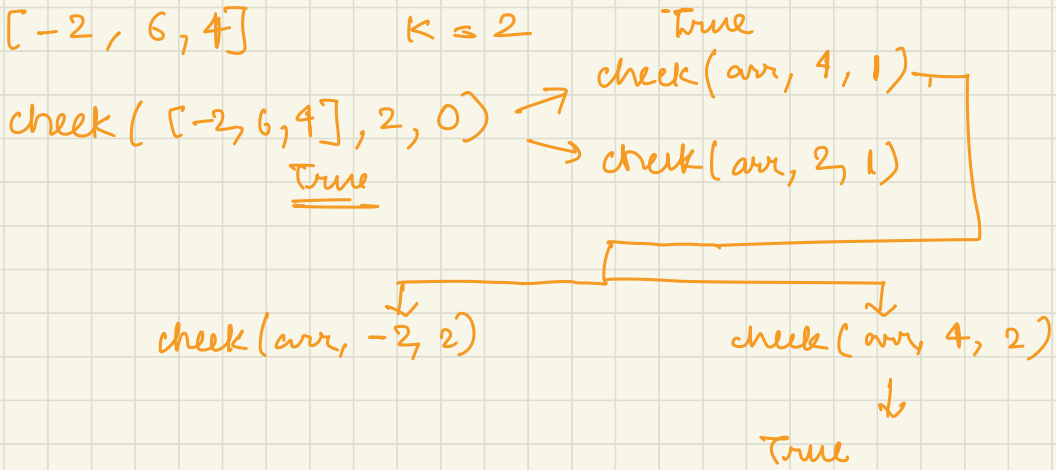
| 0 | (bit representation)

for i = 5 => sum == 2

TC: $O(2^N * N)$

Using recursion

```
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
```



(73) Find the sum of all subsequences

Brute Force

```
all-sum = 0
for i in range (2**n):
    sum = 0
    for j in range (n):
        if i & (1<<j) != 0
            sum += A[j]
    print(sum)
    all-sum += sum
return all-sum
```

$TC : O(2^N * N)$

Optimized

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

return ans
```

TC : O(N)

(74) Given N array elements, find sum of max of every subsequence

Brute Force

```
maxes = []
for i in range (2**n):
    max = -sys.maxsize - 1
    for j in range (n):
        if i & (1<<j) != 0
            if arr[j] > max:
                max = arr[j]
    maxes.append (max)
return sum(maxes)
```

TC : $O(2^N * N)$

Optimized

In every subsequence where max(arr) is present

→ max(arr) will be the max → 2^{N-1} subs—

2^{nd} largest (arr) → 2^{N-2} subseqn—

& likewise

```
sum = 0
arr.sort()
for i in range(n):
    sum += 2^i * arr[i]
return sum
```

TC: O(n log n)

75 Find the sum of (max - min) of every subsequence.

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

TC : O(n log n)