Today's Content

- → Revise Subarray
- → Subsequences vs Subsets
- → Check Subset with given sum
- → Sum of all Subsets
- → Sum of max of all Subsets

Recursion: → Gray Code → (Adv Recursion) → Try once

→ k^{th} character → (Problem Solving Session)

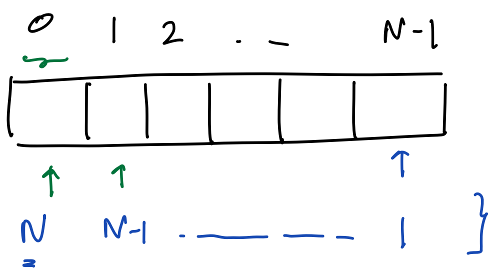
↳ Hint in Doubts Session

→ $a < 0$: $a \% m = (a \% m + m)$

1) Recur → 3 Steps
2) Code
3) TC → Recu Relation
   ↳ Recu Tree
4) SC → (Max stack size)

Subarray Basics : Continuous part of an array.

// Subarray : [s e]



: Total Subarrays = (1+2+3 .. N)

$$= \frac{(N)(N+1)}{2}$$

Subsequence : Sequence generated by deleting 0 or more elements from your array.

	0	1	2	3	4	5	6	7	
ar[8]:	3	-2	0	1	8	7	4	9	
	✗	✓	✓	✗	✓	✗	✓	✗	→ {-2, 0, 8, 4}
	✓	✓	✓	✓	✗	✗	✗	✗	→ {3 -2 0 1}
	✓	✓	✓	✓	✓	✓	✓	✓	→ {All Elements}
	✗	✓	✗	✗	✓	✓	✗	✓	→ {-2 8 7 9}
	✗	✗	✗	✗	✗	✗	✗	✗	→ {Empty Sequence}
→Seq:	✗	✗	✓	✓	✓	✗	✓	✓	{0, 1, 8, 4, 9}

{1 0 8 4 9}

Ex: [1 2 3 4 5]

a) : [1 2 3 4 5]

b) : [4]

c) [2 3 5]

d) [5 4 3]

Note: Subsequence ordered based on index value

Subarrays vs Subsequence

$$
\begin{array}{ccccccc}
 & & 0 & 1 & 2 & 3 & 4 \\
ar[5] = & \{ & -3 & 0 & 1 & 2 & 6 & \}
\end{array}
$$

	Subarray	Subsequence
[1 2 6]	✓	✓
[-3 1 2]	✗	✓
[0 1 2]	✓	✓
[-3 1 6]	✗	✓
[6 1 0]	✗	✗

// All Subarrays → Subsequences YES

// All Subsequence → Subarrays False

Sorting in Subsequence

ar[3] : { 3 -2 1 } (indices 0 1 2) $\xrightarrow{\text{sort}}$ { -2, 1, 3 } (indices 0 1 2)

All Subsequence	All Subsequen
{ }	{ }
{ 3 }	{ 3 }
{ -2 }	{ -2 }
{ 1 }	{ 1 }
{ 3 -2 }	{ -2, 3 }
{ -2, 1 }	{ -2, 1 }
{ 3, 1 }	{ 1, 3 }
{ 3, -2, 1 }	{ -2, 1, 3 }

{ 3 -2 } → max 3, min -2, sum 1

{ -2, 3 } → max 3, min -2, sum 1

Note: If we sort, data subsequence will also change

$$\underbrace{ar[\,] \xrightarrow{\text{sort}} ar[\,]}_{\text{subsequence will change}}$$

Note: In Subset data can only be unique

Subsets : Exactly Same as Subsequence order won't matter

ar[3] = $\{\overset{0}{3}\ \overset{1}{-2}\ \overset{2}{1}\}$ $\xrightarrow{\text{Sort}}$ $\{\overset{0}{-2},\ \overset{1}{1},\ \overset{2}{3}\}$

All Subsets

{ }

{3}

{-2}

{1}

{3 -2}

{-2, 1}

{3, 1}

{3, -2, 1}

All Subsets

{ }

{3}

{-2}

{1}

{-2, 3}

{-2, 1}

{1, 3}

{-2, 1, 3}

N Ele

$$\left.\begin{array}{l} {}^{N}C_{0} + {}^{N}C_{1} + {}^{N}C_{2} + \dots {}^{N}C_{N} \\ = 2^{N} \end{array}\right\}$$

Idea: Valentine day

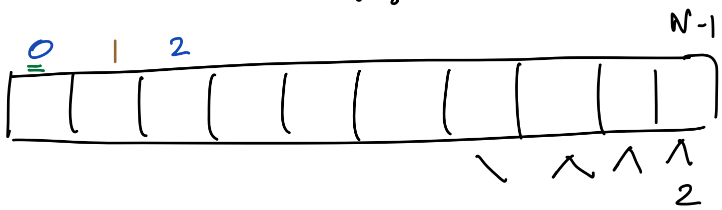
Pen & Love letter

3 — 5 = 15

Count Number of Subsesequence?

↳ By deleting 0 or more

// Given N Elements?



$= 2^N$ } Subsequences

Note: In 2^N Subsequence, 1 is Empty

// Given N → 2^N Subsequence

Given N distinct Elem → 2^N Subsets

Ex: {1 2 2} {1 2 2}

Subsequence

{ } {1} {2} {2}

{1 2} {2 2} {1 2}

{1 2 2}

Subsequ → 2^N

// Subseq → order : 2^N

// Subset → not matter : 2^N Distinct

{1 4} ↔ {4 1} { Same, Subsets}

Subsets

{ } {1} {2} {2}

{1, 2} {2 2} {1 2}

{1 2 2}

If data is repeating Subsets will change.

8) Given N distinct elements, → 2^N , $(N)(N+1)/2$

Check if there exists a subset with sum = k

↳ bool (T/F)

Ex1:

	0	1	2	3	4	5	6
ar[7] :	3	-1	0	6	2	-3	5

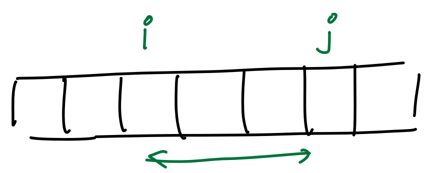
k = 10 : {-1, 6, 5}
{3, 2, 5}
{6, 2, -1, 3}
return True

k = 20 : return false

For every subset get sum & compare == k

Ideas:

1) Pf[] —



×

2) 2 loops —

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

2 Elements

3) Single loop — ×
4) HashSet — ×
5) Sort + 2pointer — ×
6) Subarrays. ×
7) Carry forward Sum + Backward ×

Ex: $ar[3] \Rightarrow \{3\ -2\ 1\} \Rightarrow 3 \Rightarrow$ [8 subsets] $\Rightarrow [0, 2^3-1]$

(indices above the array: 0 1 2)

$N \Rightarrow [2^N \text{ subset}] \Rightarrow [0, 2^N-1]$

i		2	1	0	Empty
i=0	0 :	0	0	0	→ { } = 0
i=1	1 :	0	0	1	→ {3} = 3
i=2	2 :	0	1	0	→ {-2} = -2
	3 :	0	1	1	→ {3, -2} = 1
	4 :	1	0	0	→ {1} = 1
	5 :	1	0	1	→ {3 1} → 4
	6 :	1	1	0	→ {-2 1} → -1
	7 :	1	1	1	→ {3, -2, 1} → 2

```
i = 0; i < 2^N; i++) {
    for every i, check all
    N bit positions.
    sum = 0
    j = 0; j < N; j++) {
        if (checkBit(i, j)) {
            sum = sum + ar[j]
        }
    }
    if ( sum == k) {
        return True
    }
}
return False
```

TC: $[2^N * N]$

SC: $O(1)$

10:40 break

$[2^N] \longrightarrow [N * k]$

Backtracking

Number of Ele (N)

given Sum (k)

Dynamic programming

$2^N = (1 << N)$

28 Given N distinct Elements, sum of [Subset Sums]

{3 1 4}

{ } → 0
{3} → 3
{1} → 1
{4} → 4
{3 1} → 4
{3 4} → 7
{1 4} → 5
{3 1 4} → 8

$\boxed{ans = 32}$

Idea: for every subset iterate & get sum

TC: $O(2^N * N)$ SC: $O(1)$

Idea: Contribution Technique

→ $3*(4) + 1*(4) + 4*(4)$

→ $12 + 4 + 16 → 32$

In how many subset Each Element is present?

	0	1	2	3	
ar[] :	3	2	6	8	8

Subset { 1 2 2 2 } = 8

Ex:

{3}
{3 2}
{3 6}
{3 8}
{3 2 6}
{3 2 8}
{3 6 8}
{3 2 6 8}

Ex: 8

0^{th}

_ _ _ _ _ _ _

In how many sub 0^{th} index = 2^{N-1}

$1 \times 2 \times 2 \times 2 \text{ -- -- -- } 2 = 2^{N-1}$

// In how many subsets i^{th} index is present = 2^{N-1}

```
s = 0
i=0; i< N; i++){
|    s = s + ar[i] * (2^(N-1))
}
```

TC: O(N)

SC: O(1)

Zeta ⇒ // Constraints ⇒ $(1 \le N \le 10^5)$

Given N distinct elements.

calculate (sum of all subsetsums)$/2^N$

⇒ sum of subsetsums

$$= \left(a_0 * 2^{N-1} + a_1 * 2^{N-1} + a_2 * 2^{N-1} - \ldots a_{N-1} * 2^{N-1}\right) / 2^N$$

$$= (2^{N-1})\{a_0 + a_1 + \ldots a_{N-1}\} / 2^N$$

$$= \{a_0 + a_1 + a_2 - a_{N-1}\} / 2$$

= (sum of array elements)/2

Q) Given an array, find the sum of [max of every subsequence.]

A : 3 1 -4

[_]	:	0
[3]	:	3
[1]	:	1
[-4]	:	-4
[3 1]	:	3
[3 -4]	:	3
[1 -4]	:	1
[3 1 -4]	:	3
		10

Idea: For every subsequence get max and add it to the sum.

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

Idea2: Contribution Technique

$3*4 + 1*2 + -4*1$

$\Rightarrow 12 + 2 - 4 \Rightarrow 10$

// Sorting → {max/min/sum} won't change in subsequence.

A : -4 1 3

Ans 1 2 4

{ _ _ 3 }

2 2 = 4

$\Rightarrow -4*1 + 1*2 + 3*4 \Rightarrow 10$

Ex: [3 2 6 4 5]

	0	1	2	3	4
Sort :	2	3	4	5	6
Count =	2^0	2^1	2^2	2^3	2^4

$\Rightarrow$ $2 + 6 + 16 + 40 + 96 = 160$

// Pseudo Code

// Sort in ascending

sum = 0;

i = 0; i < N; i ++) {

sum = sum + arr[i] * (1<< i)

}

TC: (N log N + O(N)) = O(N log N)

SC: O(1)

TODO :

sum of (max of every subsequence) ✓ 3 ✓

–

sum of (min of every subsequence) ✓ 3 Try

sum of [max - min of every subsequence] 2

google

Actual Question