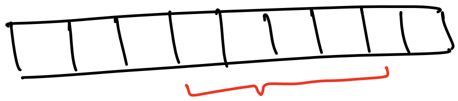
Subarray :- Contigous part of the array



subarray.

Subsequence :- sequence generated by deleting Zero or more elements from array.

arr : { -2, -3, 6, 2, 4, -1, 0, 3 }

0	1	2	3	4	5	6	7	
-2	-3	6	2	4	-1	0	3	
✗	✗	✗	✓	✓	✓	✗	✓	{2, 4, -1, 3}
✗	✗	✗	✓	✗	✗	✗	✗	{2}
✓	✓	✓	✗	✗	✓	✗	✓	{-2, -3, 6, -1, 3}
✗	✗	✗	✗	✗	✗	✗	✗	{ }
✓	✓	✓	✓	✓	✓	✓	✓	Subsequence { - - }

{6, 2, 4} ✓

{4, 2, 6} ✗

Empty sequence is also a valid Subsequence.

⇒ { 4, 2, 0 } → Not Valid. subsequence

{4 3 6} ⇒

⇒ In a subsequence elements needs to be arranged based on their index values.

Ex :- arr : { 2, 3, 8, 1, 5 } (indices 0, 1, 2, 3, 4)

1) { 3, 1, 5 } ✓

2) { 8, 2, 5 } ✗ (indices 2, 0, 4)

3) { 8, 1, 5 } ✓

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

	Subarray	Subsequence.
{-1, 4}	✓	✓
{4, 3, 9}	✓	✓
{4, 9}	✗	✓
{-1, 3, 9}	✗	✓

Observations

1. Every subarray is a subsequence.
2. Every subsequence is NOT a subarray.

{4, -1, 2} $\xrightarrow{\text{SORT}}$ {-1, 2, 4}

Subseq.

{4, -1, 2}		{-1, 2, 4}
{ } <		{ }
{4} <		{-1}
{-1}		{2}
{2}		{4}
{4, -1}		{-1, 2}
	Not same subsequence.	{2, 4}
{4, 2}		{-1, 4}
{-1, 2}		{-1, 2, 4}
{4, -1, 2}	Not same.	

Subsequences of given array and the sorted array may not be same.

SUBSETS :

Same as subsequences but Order doesn't matter.

{ 4, -1, 2 } $\xrightarrow{\text{Sort}}$ { -1, 2, 4 }

{ }		{ }
{ 4 }		{ -1 }
{ -1 }		{ 2 }
{ 2 }		{ 4 }
		{ -1, 2 }
{ 4, -1 }	Same Subset.	{ 2, 4 }
{ 4, 2 }	same	{ -1, 4 }
{ -1, 2 }		{ -1, 2, 4 }
{ 4, -1, 2 }	same.	

Subsets of a given array and Sorted array are Same.

Total number of subsequences | subsets.

arr[5] : a_0 a_1 a_2 a_3 a_4

I|E I|E I|E I|E I|E.

Every element has ② choices

→ Include

→ Exclude.

$2 * 2 * 2 * 2 * 2 \Rightarrow 2^5$

for (n) elements :-

No. of subsequences | subsets : 2^n

# of subsequences | subsets excluding the empty:

$2^n - 1$

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

Arr : { 3, -1, 0, 6, 2, -3, 5 }   K = 10.

{ 3, 2, 5 } ⇒ True. 2^n

{ 3, -1, 0, 6, 2 } ⇒ True.

Approach#1

↳ for every subset, find the sum and check if sum == K.

⇒ No. of subsets = 2^N.

N=3   { -2 6 4 }

0 → Exclude

1 → Include.

Bit masking

⇒ 2^3 = 8 subsets

0	0	0	{ }
0	0	1	{ 4 }
0	1	0	{ 6 }
0	1	1	{ 6, 4 }
1	0	0	{ -2 }
1	0	1	{ -2, 4 }
1	1	0	{ -2, 6 }

1 1 1 $\{-2, 6, 4\}$

⇒ If any bit is SET, include that element in the subset.

Traverse over all the subsets.

2^n

i → subset

j → Bit

```
for ( i = 0 ; i < 2^n ; i++ ) {
    sum = 0;
    for ( j = 0; j < N ; j++ ) {
        if ( CheckBit( i, j ) ) {
            sum += a[j];
        }
    }
    if (sum == K)  return true;
}
return false;
}
```

Traverse over the Bits

$\{-2, 6, 4\}$

K = 10

i ⇒ 0 to 7.

1) i = 0 ⇒ 0 0 0

j → 0 to 2 → N-1

if j^{th} bit is SET / UNSET in i

↳ CheckBit

0^{th} Bit, 1^{st} bit, 2^{nd} bit

sum = 0

2) i = 1, j ⇒ 0, 1, 2 ⇒ sum += arr[j] ⇒ sum += arr[0]

001
210

{-2}

sum = -2

3) $\underline{i=2}$, $\underline{j} \Rightarrow$ 0, 1, 2

✗ ✓ ✗

sum += 6 ⇒ sum = 6.

0 1 0

2 1 0

4) $\underline{i=6}$, $j \Rightarrow$ 0, 1, 2

✗ ✓ ✓

1 1 0

2 1 0

sum += arr[1] + arr[2]

= 6 + 4 = 10.

return true.

TC : $O(2^n \cdot n)$ ⇒ Exponential.

SC : $O(1)$

- Bit manipulation Sol^n : $O(N \cdot 2^N)$
- Backtracking Sol^n : $O(2^N)$
- Dynamic Programming : $O(N \cdot K)$

Q. Given an Array, find the sum of all subsets sum.

{ 4, -1, 2 }

{ } → 0

{ 4 } → 4

{ -1 } → -1

{ 2 } → 2

{ 4, -1 } → 3

{ 4, 2 } → 6

{ -1, 2 } → 1

{ 4, -1, 2 } → 5

return 20

⇒ Contribution technique :-

↳ Sum of all subarray sums.

→ Contribution of each element in subarray

$$\text{sum} += (A[i] * x)$$

↳ No. of times A[i] is appearing subarrays.

⇒ Sum of all subset sums :-

→ for every element, find that how many times it appears in subsets.

$\{2, -1, 3, 6, 8\}$ → 32 subsets.

1) 8: $2*2*2*2*1 = 2^4$

2) 6: $2*2*2*1*2 = 2^4$

3) 3: $2*2*1*2*2 = 2^4$

4) -1: $2*1*2*2*2 = 2^4$

5) 2: $1*2*2*2*2 = 2^4$.

$$\text{Sum} = 8*2^4 + 6*2^4 + 3*2^4 + -1*2^4 + 2*2^4$$

$$= 2^4\underbrace{(8+6+3-1+2)}_{\text{Sum of Array.}} \Rightarrow 2^4 * 18.$$

$$\boxed{\text{Ans} = \left(\sum_{i=0}^{n-1} arr[i]\right) * 2^{n-1}}$$

```
for (i=0; i< N; i++) {
    sum += arr[i]
}
sum *= 2^(n-1) → 1<<n-1   Left Shift
return Sum;
```

TC: O(N)

SC: O(1)

Interview Problem Given an array, find sum of subset sums divided by 2^N.

$$\text{sum of subset sums} \Rightarrow (\text{array sum}) * 2^{N-1}$$

$$\Rightarrow \frac{(\text{array sum}) * 2^{N-1}}{2^N}$$

$$\Rightarrow \frac{(\text{array sum})}{2}.$$

Q Given N array elements, calculate sum of max of every subsequence.

{ 3 1 -4 }

{ } → 0 +
{3} → 3 +
{1} → 1 + ⇒ 10
{-4} → -4 +
{3,1} → 3 +
{3,-4} → 3 +
{1,-4} → 1 +
{3,1,-4} → 3

Sort the Array :-

{ -4, 1, 3 }

{ } → 0

{-4} → -4

{1} → 1

{3} → 3

{-4, 1} → 1

{1, 3} → 3

{-4, 3} → 3

{-4, 1, 3} → 3.

$\Rightarrow$ $\underbrace{\{x, y, z\}, \quad \{z, y, x\}, \quad \{y, z, x\}}_{\text{Same sum | Prod | Max | Min.}}$

$\Rightarrow$ {-2, 8, 0, 4, 3}

↓ Sort

{ -2, 0, 3, 4, 8 }

How many subsequences will have A[i] as a max element.

1) 8 : $2 * 2 * 2 * 2 * 1 \Rightarrow 2^4$ → No. of subsequences in which 8 will be MAX. element.

2) 4 : $2 * 2 * 2 * 1 * 1 \Rightarrow 2^3$

3) 3 : $2 * 2 * 1 * 1 * 1 \Rightarrow 2^2$

4) 0 : $2 * 1 * 1 * 1 * 1 \Rightarrow 2^1$

5) -2 : $1 * 1 * 1 * 1 * 1 \Rightarrow 2^0$

Sum $\Rightarrow$ $8 * 2^4 + 4 * 2^3 + 3 * 2^2 + 0 * 2^1 + (-2) * 2^0$

Step I : Sort

Step II :
```
for ( i = n-1 ; i >= 0 ; i-- ) {
      sum + = ( arr[i] * 2^i ) ;
}
```

Generalize this for N elements.

$a_0 \quad a_1 \quad a_2 \; - - - - \; a_{n-2} \quad \underline{a_{n-1}}$ $\Rightarrow$ Sorted.

1) $a_{n-1} \Rightarrow 2 * 2 * 2 - - - 2 * 1 \Rightarrow 2^{n-1}$

2) $a_{n-2} \Rightarrow 2 * 2 * 2 - - - 2 * 1 * 1 \Rightarrow 2^{n-2}$

$$\boxed{Sum = \sum_{i=0}^{n-1} a_i * 2^i}$$

Summation.

HW :- Sum of MIN of every subsequence.