Todays Content:

1. Subsequences Intro
2. Subsequence with given sum
3. Sum of max of all subsequence
4. Intro to Subsets vs Subsequences

1min

Today DSA: 1

- 20th: Contest
- 23rd: Holiday
- 25th: Hashing

Subarrays vs Subsequence vs Subsets : 25mins

Subarray : Continous part of an array.

Subsequence : Sequence obtained by deleting none or more ele from arr[]

1. Data should be arranged based on inc order of index
2. Empty subsequence is also valid.

Eg:

$arr[6] = \begin{Bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 1 & 9 & 6 & 8 \end{Bmatrix}$

{1 6 8} subseq → = {x x ✓ x ✓ ✓}

3 4 0

{9 6 3} order miss → = {3 x x 9 6 x}

{2 9 8} subseq → = {x 2 x 9 x 8}

{1 9 6 8} subseq → = {x x 1 9 6 8}

{2 1 9 6} subseq → = {x 2 1 9 6 x}

{ } subseq → = {x x x x x x}

{3 2 1 9 6 8} sub → = {x x x x x x}

Subarr[] vs Subsequence

$arr[6] = \begin{Bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 1 & 9 & 6 & 8 \end{Bmatrix}$

	Subarray:	Subsequence
{3 2 1}	True	True
{1 9 6 8}	True	True
{3 1 6 8}	Not	True
{2 1 9}	True	T

Obs:

1. All subarrays are Subsequences Yes
2. All subsequen are Subarrays No

Properties of Subsequence

	0	1	2
ar[3] =	3	1	8

sort[] in inc → ar[] = {1 3 8}

Sum	min	All Subsequences		All Subsequences	Sum	min
0	–	{ }	match	{ }	0	–
3	3	{3}	match	{3}	3	3
1	1	{1}	match	{1}	1	1
8	8	{8}	match	{8}	8	8
4	1	{3 1}	not matching, different order	{1 3}	4	1
9	1	{1 8}	match	{1 8}	9	1
11	3	{3 8}	match	{3 8}	11	3
12	1	{3 1 8}	not matching, different order	{1 3 8}	12	1

obs: If we sort ar[]

1. Order before & after sorting change
   But data remains same.

2. Sum of Subsequences
   Max of Subsequences
   Min of Subsequences
   } are not effected by sorting

Count of subsequences:

ar[1] = {6}
allsub:
{ } {6}
count = 2

ar[2] = {2 5}
all sub
{ } {2}
{5} {2 5}
count = 4

ar[3] = {3 5 10}
8 subsequences

ar[N] = 2^N Subsequences

ar[N] = $(N)(N+1)/2$ Subarrays

Q) Given an ar[N], check if there exists a subsequence with sum = k

Constraints

$1 <= N <= 20$

$1 <= ar[i] <= 10^6$

Eg: ar[5] = { 2 5 3 11 7 } (0 1 2 3 4)

k = 20 : {2 11 7} True

k = 8 : {5 3} True

k = 9 : {2 7} True

k = 19 : {5 3 11}

k = 50 : { } No

Idea:

1. Prefix Sum *
   Subsequence need not be continous
2. Iterate on ar[] & take sum *
   Need not start from 0 & not continous
3. Kadanes *
   Apply for subarrays

Idea2: Generate all subsequence & get their sum == k:

int ar[] = { 7 5 3 } 0 → That ele is not *

1 → That ele is present ✓

Decimal

i = 0	{ 0 0 0 }	→ { }
i = 1	{ 0 0 1 }	→ { 3 }
i = 2	{ 0 1 0 }	→ { 5 }
i = 3	{ 0 1 1 }	→ { 5 3 }
i = 4	{ 1 0 0 }	→ { 7 }
i = 5	{ 1 0 1 }	→ { 7 3 }
i = 6	{ 1 1 0 }	→ { 7 5 }
i = 7	{ 1 1 1 }	→ { 7 5 3 }

obs: ar[3]: i = 0, 1, .. 7 : 8 numbers

For every i : [Generate binay of 3 bit]

```
                0  1  2  3
int ar[] = { 5  2  6  8 }
```

Decimal

	3	2	1	0	: pos	sum = ar[0] + ar[1] + ar[2] + ar[3]
✓ i = 0	0	0	0	0	= { }	
✓ i = 1	0	0	0	1	= { ar[0] }	
✓ i = 2	0	0	1	0		
✓ i = 3	0	0	1	1	= { ar[0] + ar[1] }	
i = 4	0	1	0	0		
i = 5	0	1	0	1		
i = 6	0	1	1	0		
i = 7	0	1	1	1		
i = 8	1	0	0	0		
i = 9	1	0	0	1		
i = 10	1	0	1	0		
✓ i = 11	1	0	1	1	= { ar[0] + ar[1] + ar[3] }	= 15
→ i = 12	1	1	0	0		
i = 13	1	1	0	1		
i = 14	1	1	1	0		
i = 15	1	1	1	1		

```
bool subSeq(int ar[], int k){
    int N = ar.length; // subseq → 2^N  iterate : i ⇒ [0, 2^N - 1]
    for(int i = 0; i < 2^N; i++){ ------------------> 2^N      TC: O(N*2^N)
        // For every i ⇒ Get Binary length = N
        int sum = 0;
        for(int p = 0; p < N; p++){ ----------> N
            // check if for i, check if p th bit position is set or not
            if( (i >> p) & 1 == 1) { // p th bit pos is set
                sum = sum + ar[p]
            }
        }
        if(sum == k) {return True}
    }
    return false;
}
```

Q2: Given ar[N] calculate sum of max of every subsequence

Ex: ar[3] = { 3 1 -4 }

All SubSequences	Max
{ }	0
{ 3 }	3
{ 1 }	1
{ -4 }	-4
{ 3 1 }	3
{ 1 -4 }	1
{ 3 -4 }	3
{ 3 1 -4 }	3

ans = 10

Ex: ar[3] = { 3 4 3 } (0 1 2)

All SubSequences	Max
{ }	0
{ 3 }	3
{ 4 }	4
{ 3 }	3
{ 3 4 }	4
{ 4 3 }	4
{ 3 3 }	3
{ 3 4 3 }	4

ans = 25

Idea1: Generate all subsesequence
get max of all them & calculate Sum
TC: O(N * 2^N)

Idea2: Add contribution of each element.

Sum = 3*4 + 1*2 + -4*1
= 12 + 2 - 4 = 10

Calalate, occurence of each ele?

In how many subsequences 7 is max = 8

Ex1: ar[ ] = { 4 7 2 5 8 10 }
{ x 7 x x * * }
{ x 7 x 5 }
{ x 7 2 x }
{ x 7 2 5 }
{ 4 7 x x }
{ 4 7 x 5 }
{ 4 7 2 x }
{ 4 7 2 5 }

Idea: No: of elements < 7 = 3
With 3 ele, no: of sub = $2^3 = 8$

Idea2: For every ar[i], Calculate No: of elements < ar[i] in ar[ ]?
to calculate, no: of subsequence in which ar[i] is max

Obs: For every ar[i], to calculate no: of subsequences in which it's max, we need to calculate, no: of ele are < ar[i]

Hint: Sort arr[]

Ex1: ar[6] = { 4 7 2 5 8 10 } (indices 0 1 2 3 4 5)

↕

ar[6] = { 2 4 5 7 8 10 } (indices 0 1 2 3 4 5)

#elements < ar[i] = 0 1 2 3 4 5

#occurences = 2^0 2^1 2^2 2^3 2^4 2^5

ar[i] max = 1 2 4 8 16 32

#contribution = $2 + 8 + 20 + 56 + 128 + 320 =$

$= 10 + 20 + 320 + 184 =$ ans

Ex2: ar[3] = { 3 4 3 } (indices 0 1 2)

↓

ar[3] = { 3 3 4 }

Contribu = $3*2^0 + 3*2^1 + 4*2^2$

$= 3 + 6 + 16 = 25$

```
long sumMax(int arr[]){ TC: O(NlogN)
    int N = ar.length;
    long ans = 0;
    Arrays.sort(ar); // TC: O(NlogN), N = no: of elements
    for(int i=0; i<N; i++){
        // no: of ele < ar[i] = i
        // Count of subsequence in which ar[i] = 2^i

        long con = ar[i] * (2^i)
        ans = ans + con;
    }
    return ans;
}
```

Q3: Sum of min of every Subsequence : TODO

Subset:

Set obtained by deleting none or more ele from arr[]

1. Empty Set is also valid

Note: Order doesn't matter.

Eg:

	0	1	2	3	4	5
arr[6] =	3	2	1	9	6	8

{1 6 8} —subset→ = { x x ✓ x ✓ ✓ }

{9 6 3} —subset→ = {3 x x 9 6 x }

{2 9 8} —subset→ = {x 2 x 9 x 8 }

Eg: {2 4 1} —sort in inc→ {1 2 4}

All Subsets:

{ }

{2}

{4}

{1}

{2 4}

{2 1}

{4 1}

{2 4 1}

All Subsets:

{ }

{2}

{4}

{1}

{2 4}

{1 2}

{1 4}

{1 2 4}

Note: Sorting won't effect Subsets