Subsets & Subsequences

0 1 2 3

① [3, -1, 2, 6]

N = 4

s e

→ Subarrays.

Contiguous part of an array.

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

2

4 * 5 = 10

$\frac{}{2}$

Subsequence

→ need not be contiguous.

DNA ⇒ A T G C.

String matching algorithms

Longest Common Subsequence.

Knapsack Problem.

Subsequence

Generated by removing 0 or more elements from the given array.

A =

0	1	2	3	4	5	6	7
-2	-3	6	2	4	-1	0	3

X X X X X X X X ⇒ [ ] empty

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ⇒ A itself.

X ✓ ✓ ✓ ✓ ✓ ✓ ✓ ⇒ [-3, 6, 2, 4, -1, 0, 3]

X ✓ X ✓ X ✓ ✓ ✓

⇒ [-3, 2, -1, 0, 3]

1 3 5 6 7

Observation

1. All subarrays are subsequences

2. All subsequences are NOT subarray.

↓

no need of being contiguous

NOTE: Subsequence : elements have to be placed/ arranged based on index values.

indices have to be sorted.

```
      0  1  2  3  4
A =  [3, -1, 0, 6, 2]
      ✓  X  ✓  X  ✓
```

[3, 0, 2] => 0, 2, 4

[0, 3, 2] => 2, 0, 4

[2, 0, 3] => 4, 2, 0

Quiz-3

```
     0   1   2
    [4] [-1]  2]
     x
```

```
     1   0
    [-1, 4]
```

[4, -1]

Subsets: Exactly same as subsequence but order does not matter.

picking some elements

0 or more.

Quiz-4

a subset of [4,-1,2]

X

[1, 1, 2, 2, -1, 4, 5]

[2, 1, 1, -1, 5]

Subset? ✓

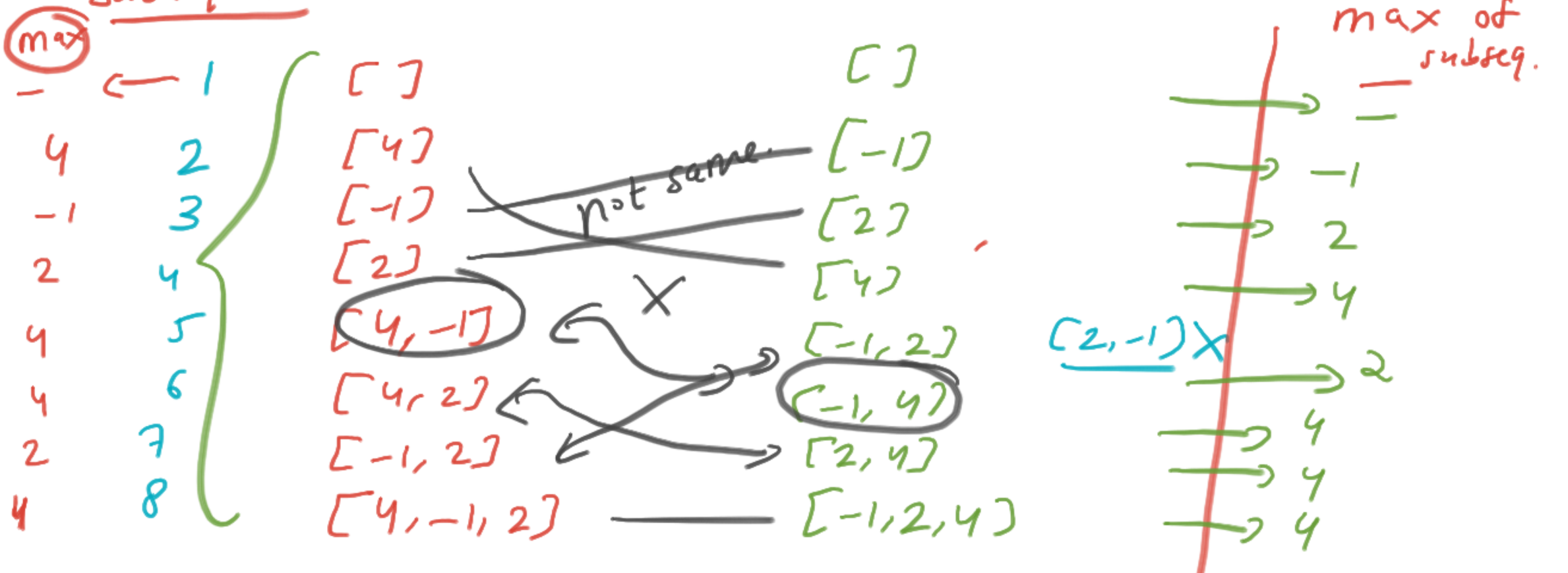
=) All ~~subsets~~ are ~~subsequences~~. ✗

✓ All subsequences are subsets.

Sorting affects subsequences.

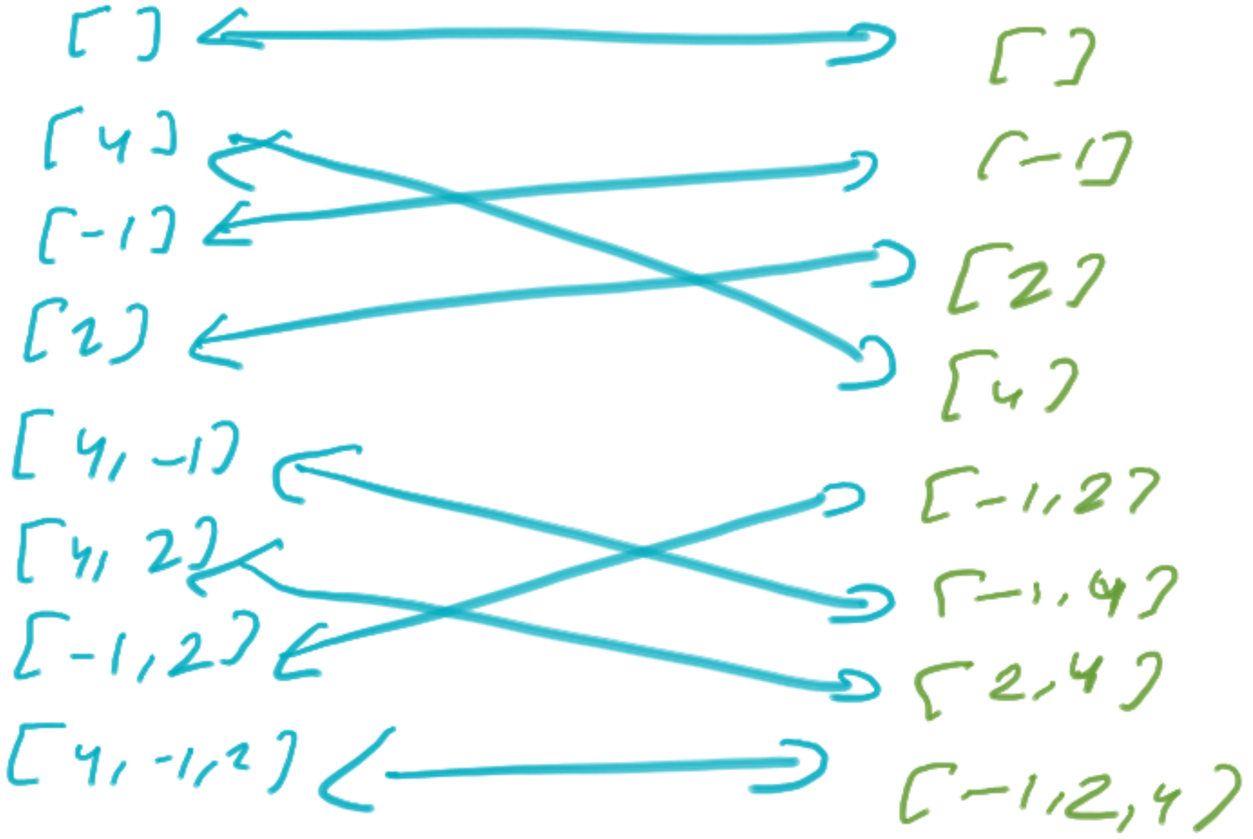
Sort an Array.

A = [4, -1, 2] ⟹ [-1, 2, 4]



Subsets

[4, -1, 2] ⟹ [-1, 2, 4]



Sorting does not affect subsets but subsequence.

Q1 Count no. of subsequences for a given array of size N.

N = 2 ⇓ 4

N = 3 ⇓ 8.

X X => []

X ✓ => [1]

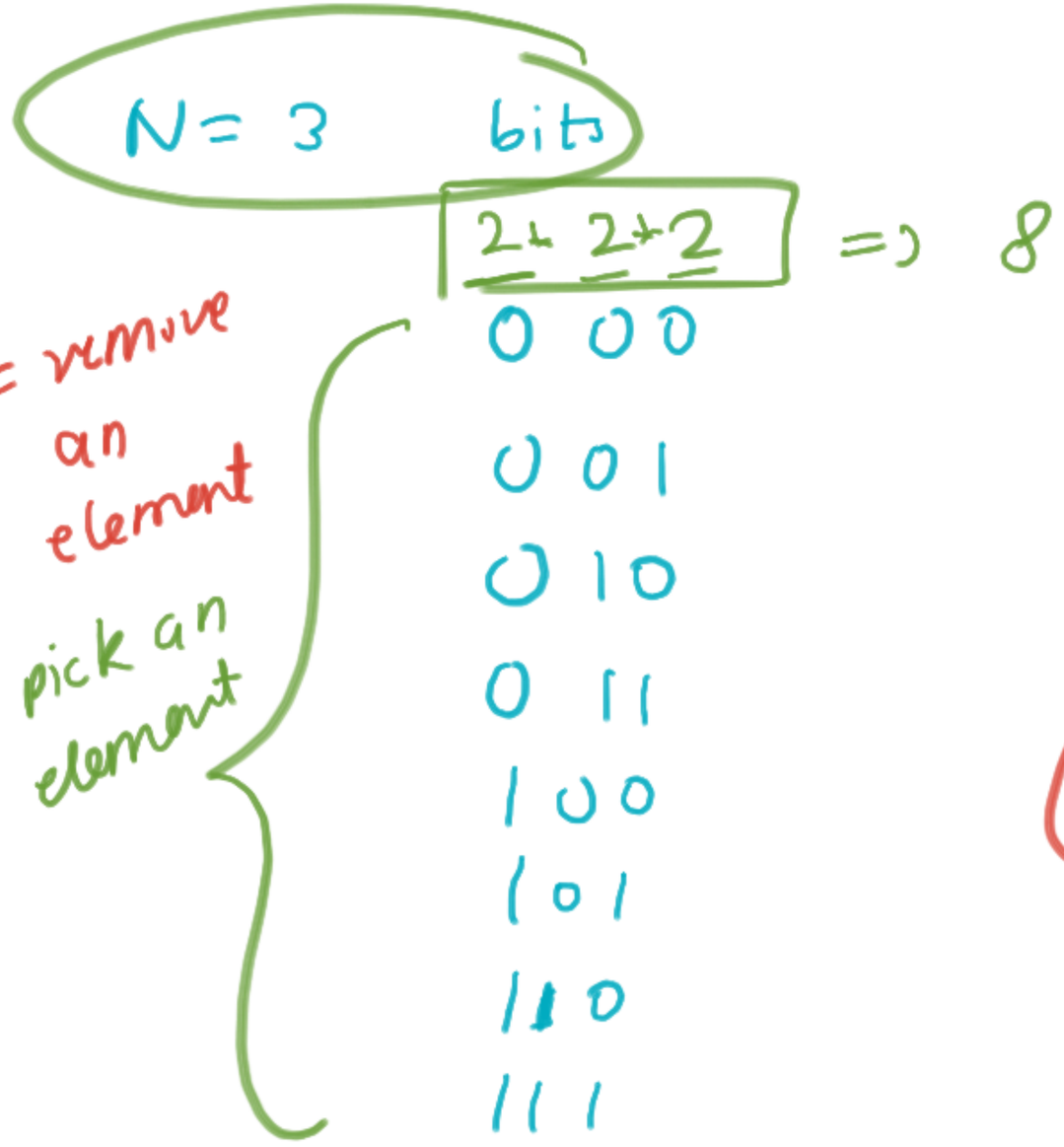
✓ X => [-1]

✓ ✓ => [-1, 1]

[a1, a2, a3, a4 ... aN]

2 * 2 * 2 * 2 * ... * 2

$= 2^N$.



$N \Rightarrow 2^N$ subsequences.

$(2^N - 1)$ non-empty subseq.

Q2 Given (N) elements, check if there is a subset with given sum (X).

True / False

a subset

↓

at least 1

For now, ✓ subset = subseq

Eg:

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

X = 0

Ans = True.

Eg 1

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

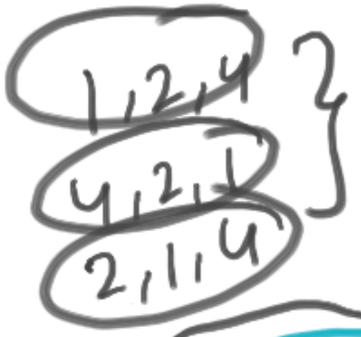
X = 10

3, -1, 2, 6

5 + 2 + 3 = 10

1, 3, 6

$X = 10^6$



Subseq

11

Subsets

A = [1, 2, 3, 4]

↓ ↓ ↓ ↓

$2 * 2 * 2 * 2 = 2^4$

= 16

Subarrays

2

$\frac{4 * 5}{2}$ = 10

N = 10

2^N

2^N

$2^{10} = 1024$

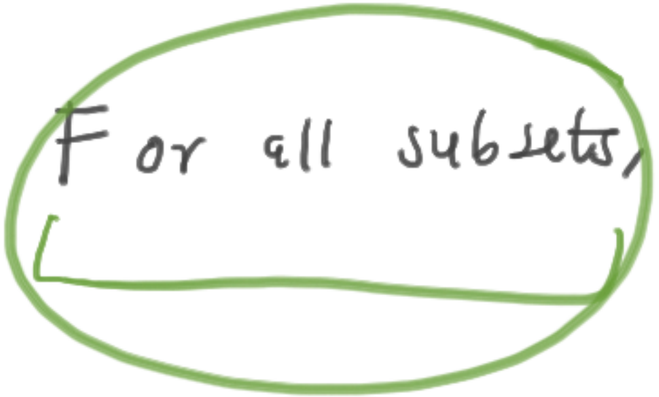
$\frac{N(N+1)}{2}$

N^2

100

Brute Force

For all subsets, check if sum = X.



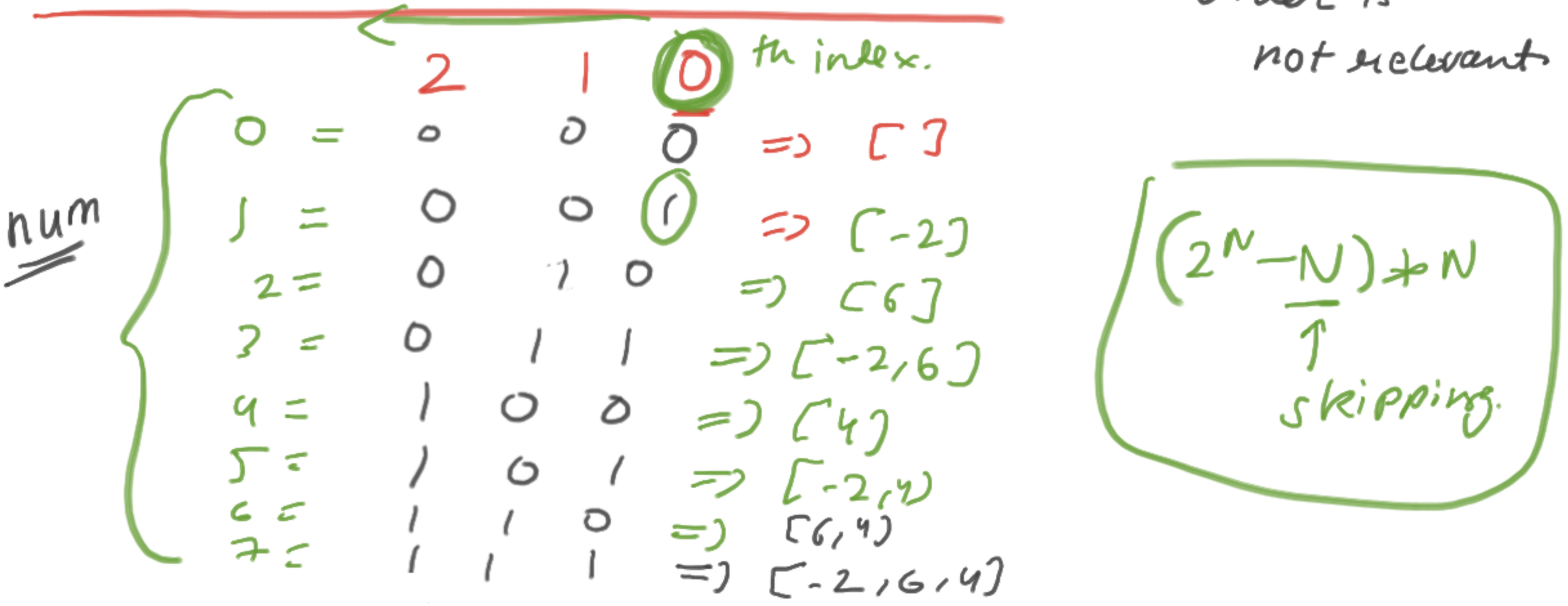
⇒ Find all subsets.

(2) How to generate all subsets

Currently. Subsets = Subsequences.

Order is not relevant

```
      0  1  2
A :  [-2, 6, 4]
```



$(2^N - \underline{N}) + N$

↑ skipping.

N = 3

non-empty ×

[0, 7]

```
for num in range(1, 2**3):
    # map this num to a subset.
    curSubset = [], sumSubset = 0
    for idx in range(0, 3):    # [0,2]
        if checkBit(num, idx):
            curSubset.append(A[idx])
            sumSubset += A[idx]
    if sumSubset == X:
        return True.
```

N N — Generalize.

2 1 0
1 0 1

num & (1<<idx)

Checking if idx bit is set in the num or not.

TC:

Outer loop $1 \rightarrow 2^N$

2^N

Inner loop $0 \rightarrow N$

N times.

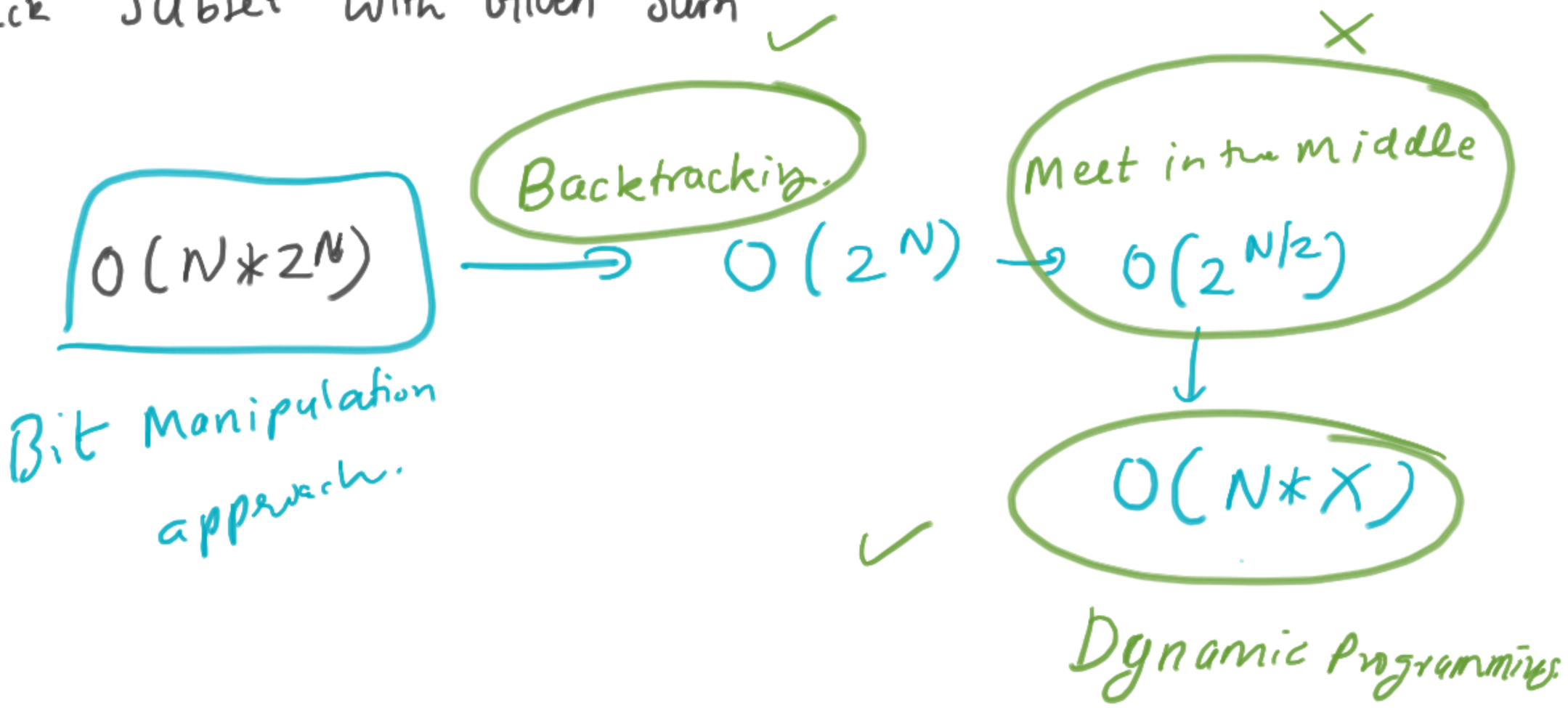
$O(2^N * N)$

for every num

checking all the N bits

SC: $O(1)$

Check Subset with Given Sum



Part - 2

Sum of all

=> Contribution Technique.

Q 2

Google

Given N array elements, find the sum of all subset sums.

= subsequence

Find sum of each subset & add those sums.

Es 1. N = [1, 2]

~~[ ]~~ empty X

[1] => 1

[2] => 2

[1, 2] => 1 + 2 = 3

3 + 2 + 1 = 6

N = [-1, 1]

[ ]

[-1] = -1

[1] = 1

[-1, 1] = -1 + 1 = 0

-1 + 1 + 0 = 0.

---

(1) Brute force.

Generate all subsets & find their total sum.

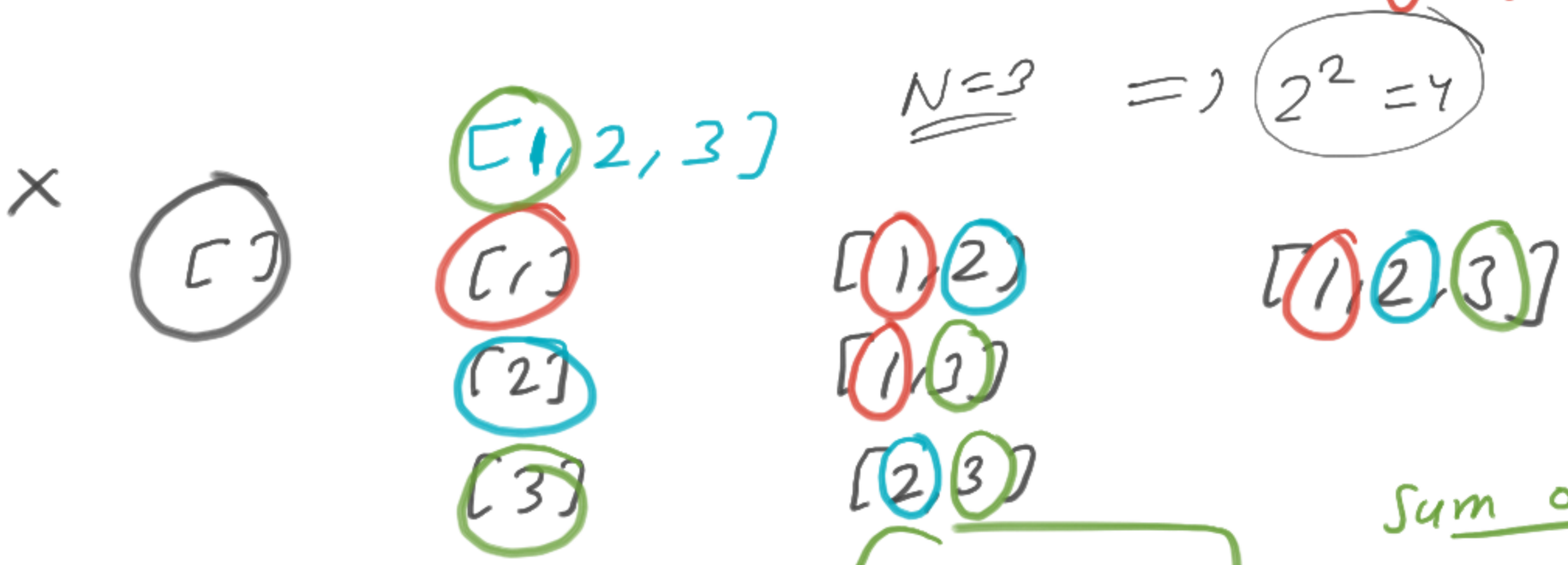
TC: $O(N * 2^N)$

$\Rightarrow$ $O(N)$ time

sum of all => contribution technique. for every elt find contribution.

(Q) In how many subsets, is the element P[i] going to be present for an array of size N.

N=3 => $2^2 = 4$

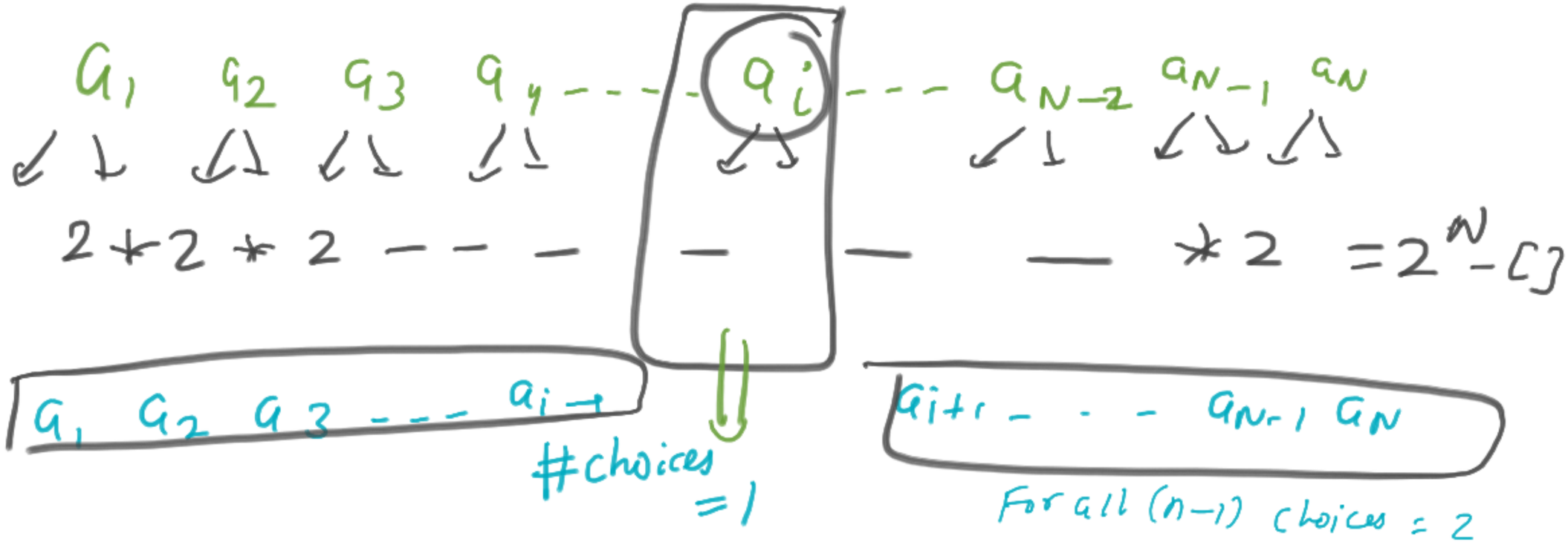


1: appearing in 4 subsets.
2: appearing in 4 subsets.
3: appearing in 4 subsets

Sum of subsets

1 * 4 = 4
2 * 4 = 8
3 * 4 = 12

subsets $= (2^N) - 1$ excluding empty



1, 2, 3, 4

↑ ↑ ↑ ↑

$2 * 2 * 1 * 2 = 2^3$

3

3

1, 3

2, 3

4, 3

1, 2, 3

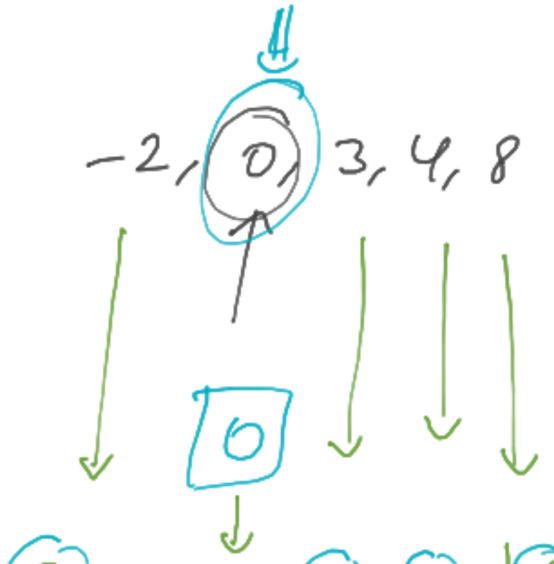
1, 4, 3

2, 4, 3

1, 2, 3, 4

$2^{(N-1)}$

$N = 5$



$= 2^{(N-1)}$

multiplying 2 $(N-1)$ times.

Contribution of each element: $(n-1)$ 2's

$0 * 2^4 = 0$

$-2 * 2^4 = -2 * 16 = -32$

$3 * 2^4 = 3 * 16 = 48$

$4 * 2^4 = 4 * 16 = 64$

$8 * 2^4 = 8 * 16 = 128$

$0 + (-32) + 48 + 64 + 128$

= sum of all subset sums.

$= 2^4 (0 - 2 + 3 + 4 + 8)$

$= (2^{N-1})$ sum of the array.

p = power(2, N-1) ⇒ O(log N)

s = sum(A) ⇒ O(N)

return p*s

sum of all subset sums.

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

SC: O(1)

Q4 Calculate $\frac{\text{sum of all subset sum}}{2^N} = \frac{\text{sum} * 2^{N-1}}{2^N}$

$= \boxed{\frac{\text{sum}}{2}}$

Q5 Given N array elements, find the sum of max of every subsequence.

subset

max will remain same

order will not matter.

Eg 1 [1]

[1] => max 1

[] X

ans = 1

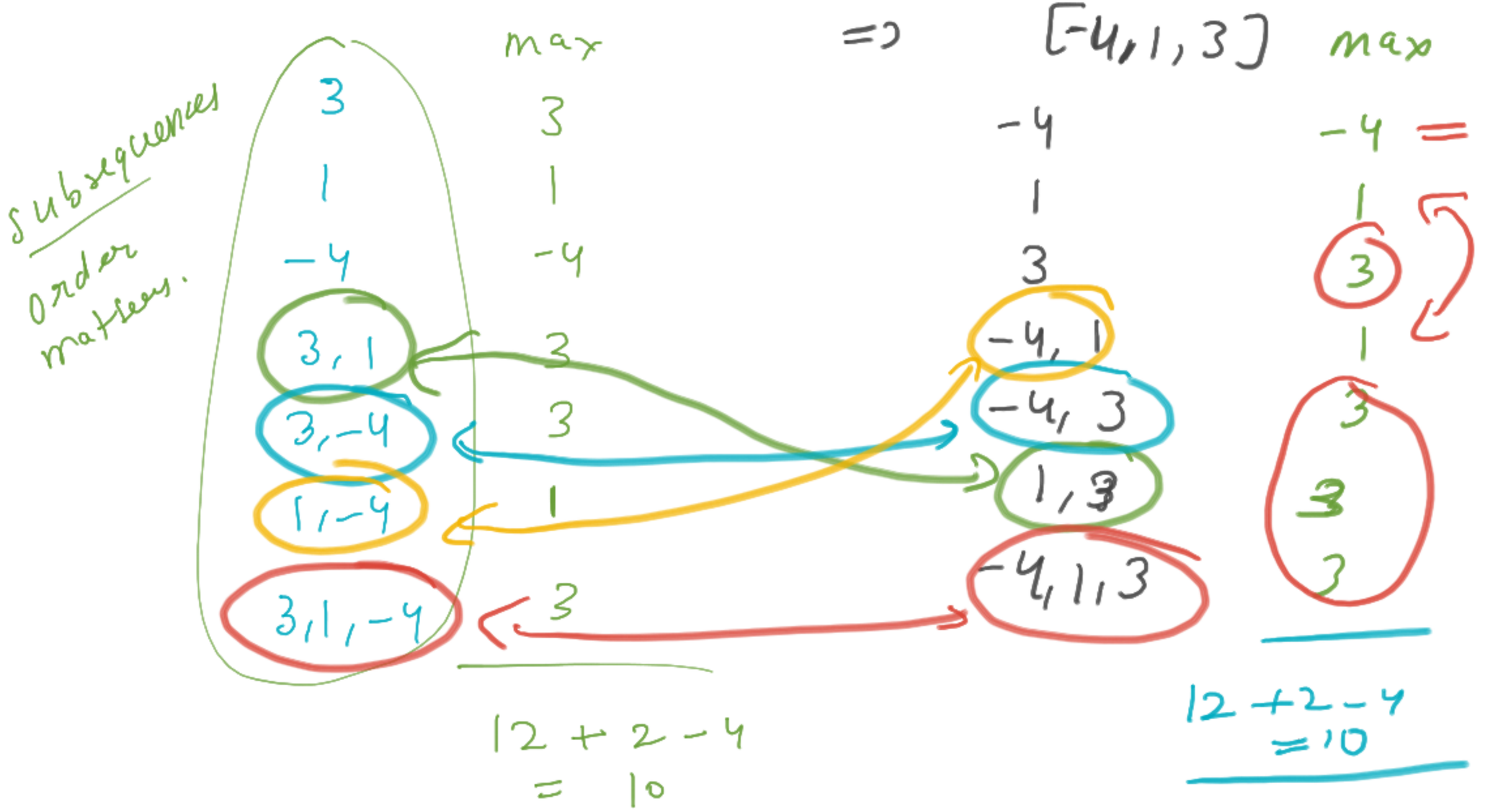
Eg 2 [-1, 1]

	max
[-1]	-1
[1]	1
[-1, 1]	1

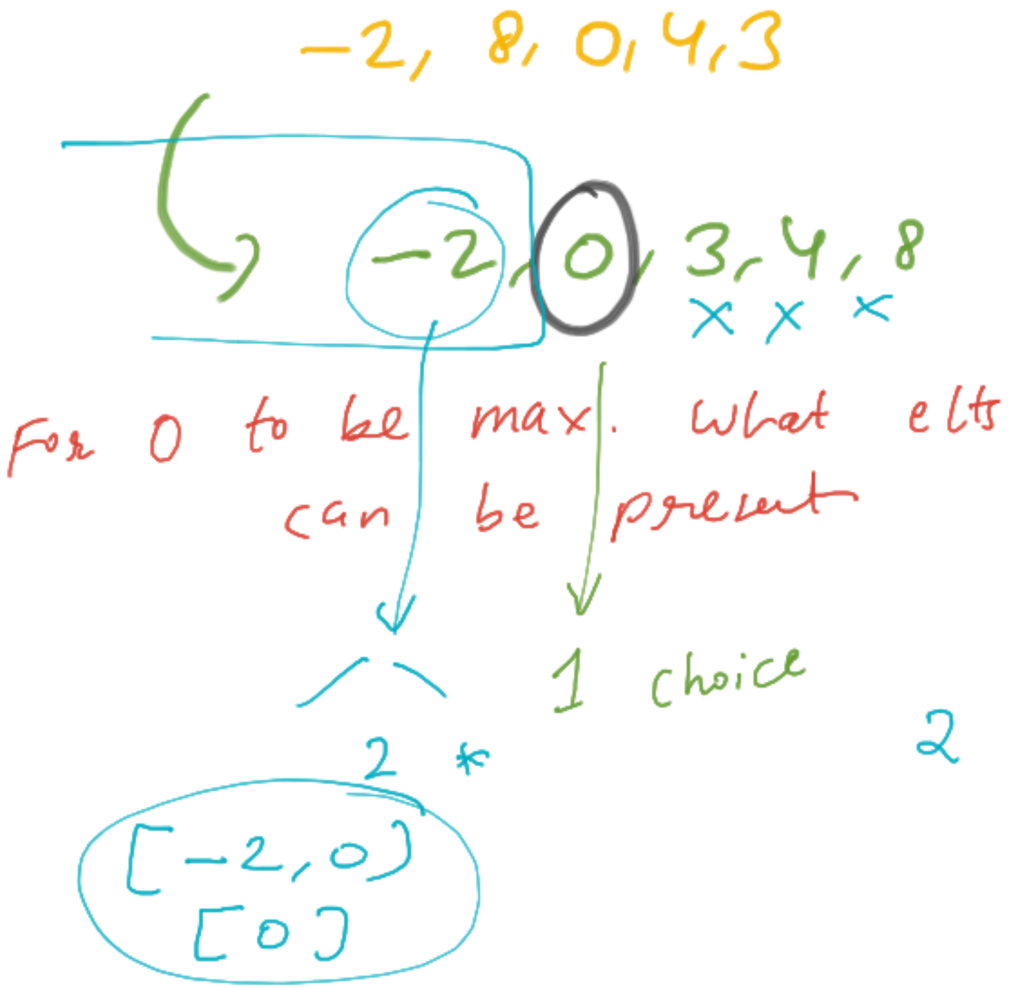
-1+1+1 = 1

Eg 3

$A = [3, 1, -4]$ (indices 0, 1, 2)

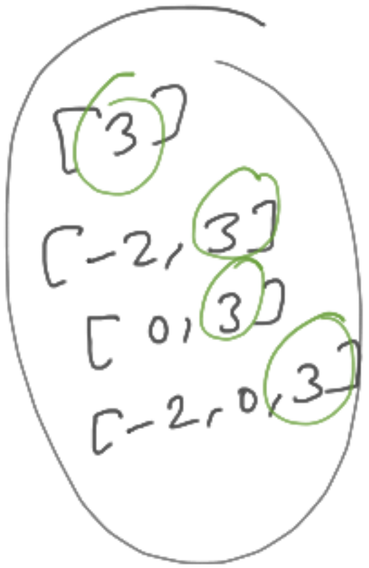


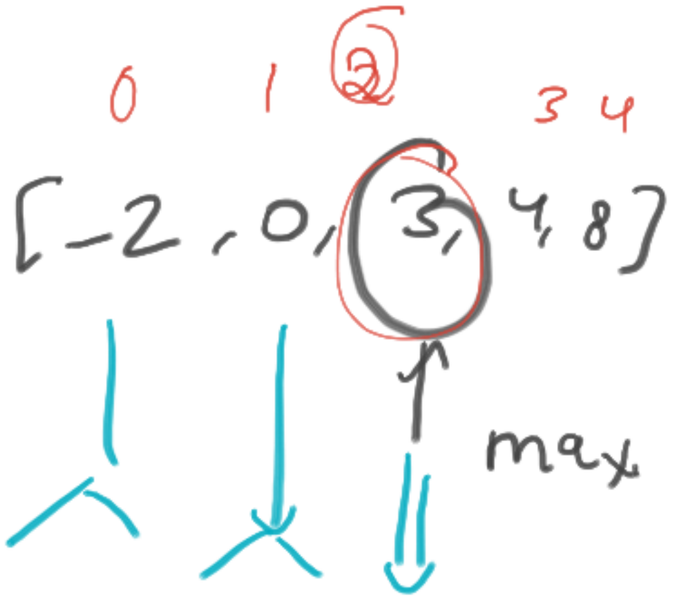
Sum of max of every subsequence after sorting array.



Contribution Technique

For every elt in how many subsequences does it appear as the max element.

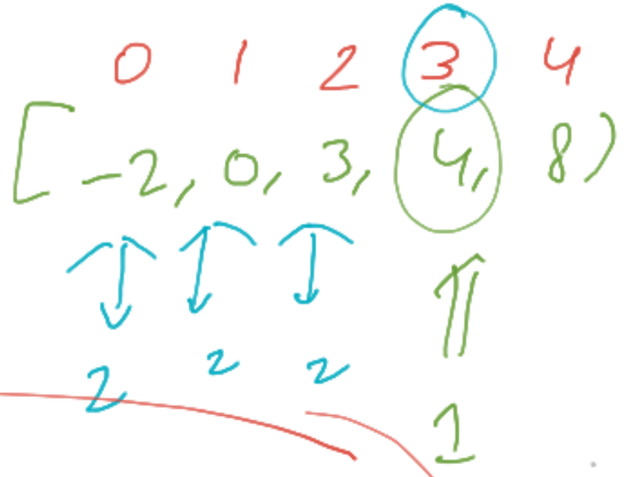




Index 2

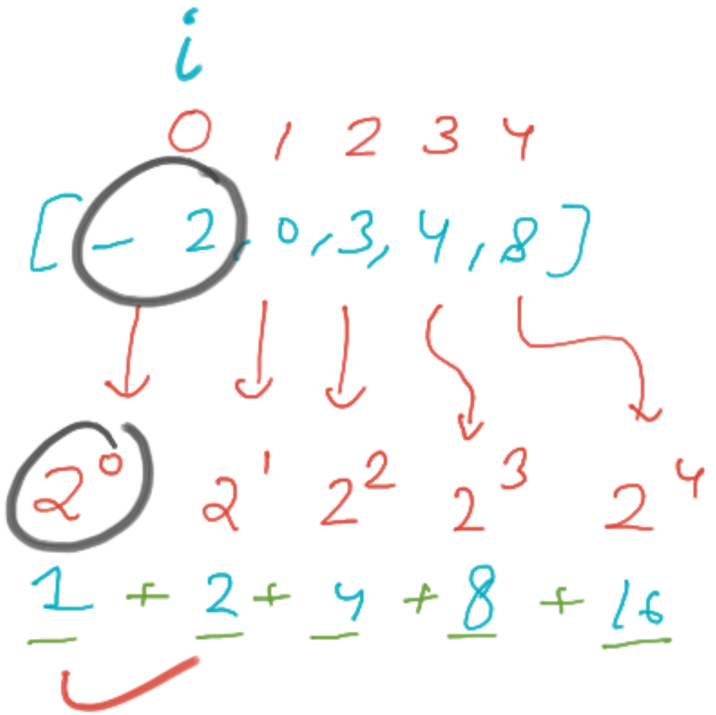
2 elements are before it

$2 * 2 * 1 = 4$ choices. $= 2^2$



$= 2^3 = 8$

Idx i $\Rightarrow 2^i$



$$\sum_{i=0}^{N-1} a[i] * 2^i$$

TC: O(N)

In how many subsequences, it is contributing.

Contribution

$1 * (-2) = -2$

$2 * 0 + 0$

$4 * 3 + 12$

$8 * 4 + 32$

$16 * 8 + 128$

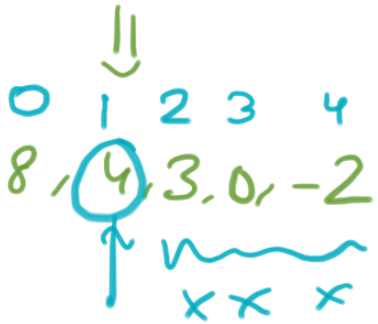
```
p = 1
s = 0
for i in range(0, N):
    s += p * a[i]
    p *= 2
```

2^i

(e) Find sum of min for every subsequence.

-2, 8, 0, 4, 3

sort

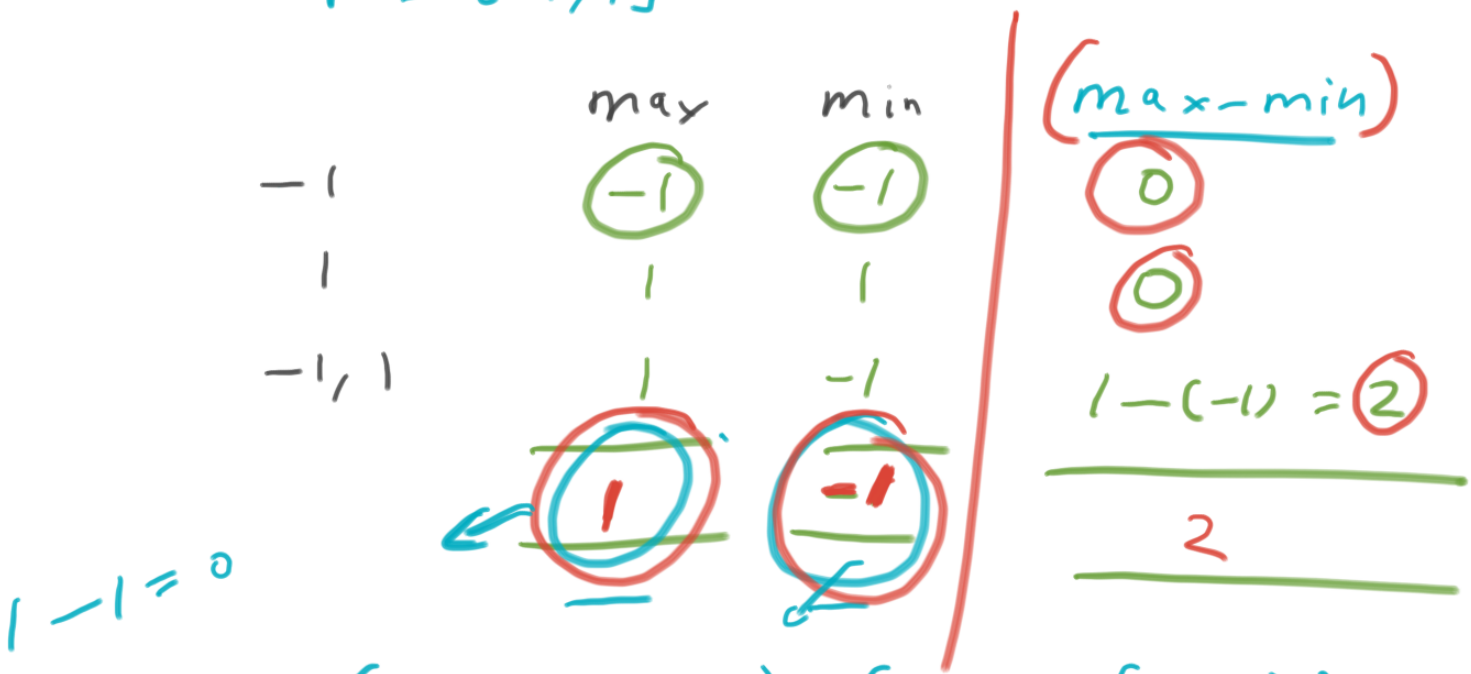


Just sort in reverse direction.

$2^i * A[i]$

(Q) Find sum of (max−min) of every subsequence.

A = [-1, 1]



(sum of max) - (sum of min)

= 1 − (−1) = 2

Eg 2 A = [3, 1, -4]

-4 * 4 = -16
+1 * 2 +2
+3 * 1 +3
= -11

	max		min	max - min
[3]	3	>=	3	0
[1]	1	>=	1	0
[-4]	-4	>=	-4	0
[3,1]	3	>=	1	2
[1,-4]	1	>=	-4	5
[3,-4]	3	>=	-4	7
[3,1,-4]	3	>=	-4	7
	10	—	-11	21

10 - (-11)
= 10 + 11 = 21

max min

1 - 1

0

Sum of max = 1+2+4+8+16 = 31

Sum of min = 31

sum of max - sum of min = 31 − 31 = 0

1) Pre-Read before OOPS Python.

2) Try to cover backlogs.

Revise

$(a*b)\%m = \{(a\%m)*(b\%m)\}\%m$

$a^N \% d$

$T(N) = 2T(N/2) + O(N)$

$\Rightarrow O(N \log N)$

Assumption

$a^N \% d$

$a^N \% d$

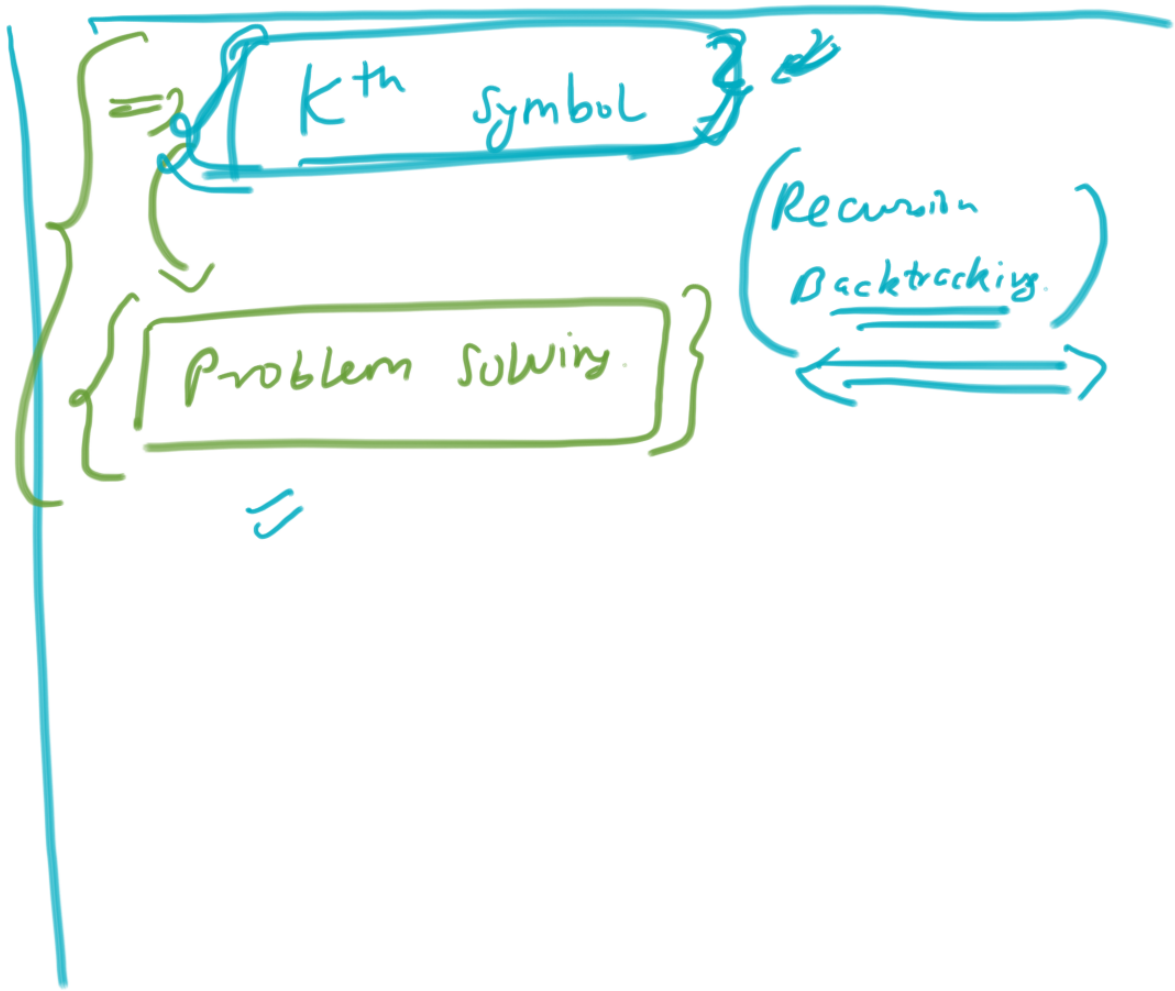
```
def power(a, n, d):
    if (n == 1)
        return a
    hp = power(a, n//2, d)
    if n & 1 == 0:
        return ((hp % d) * (hp % d)) % d
    else
        return (a * [          ]) % d
```

TC: O(log N)

$2^3 \% 5$

$= 8 \% 5$

$= \boxed{3}$



Revise

Recall ⇒ Ask Questions.

1. Visit your notes/ class notes
2. Visit archived lectures just see the bookmark heading.
   Frame questions which you can recall later.
3. Bookmark/ mark some problems for later
   - struggled with it.
   2 hr → 30 min.
   ↳ Understand deeply