Q⇒ You are given an array. find the subarray with maximum sum in the array.

$N \leq 10^5$

cumulative sum / prefix sum

[-2, -2, 4, -1, -2, 1, 5, -3]

2 2 2 2

candidates for answers

global-max = 6 ✗ 7

pref-sum = 4 + (-1) = 3 + (-2) = 1 + 1 = 2 + 5 = 7 - 3 = 4

if the pref. sum becomes ≤ 0, then we discard everything before it & start freshly a new array

Kadane's Algorithm

at each point of time we iterate

global_max = 0 → progress

max_val_ends_here = 0

[3, -1, 2]

```
for (i : arr)
    max_val_ends_here += arr[i]
    if (max_val_ends_here > globalmax){
        update globalmax
    }
    if (max_val_ends_here < 0)
        max_val_ends_here = 0;
```

Q → You have an unsorted array of length N, & elements are from the range 1 – N. There is one element from the range present twice. Find the repeated element.

arr[2]

$N \le 10^7$

N = 5

[-1, -3, -2, 5, -2]

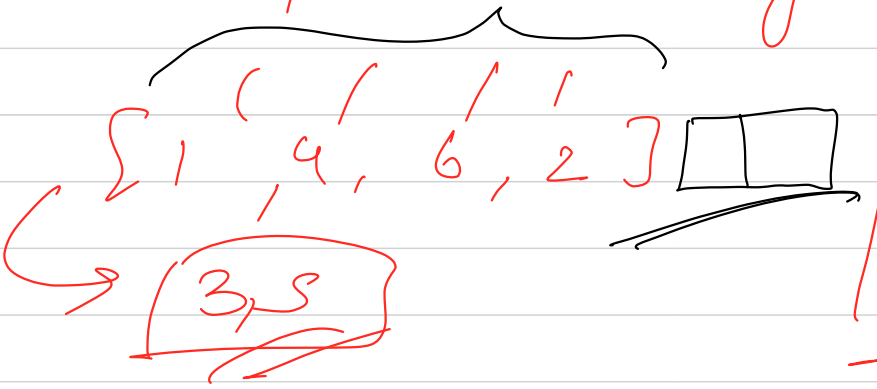
1 2 3 4 5

We can mark the presence of an element arr[abs(arr[i])]

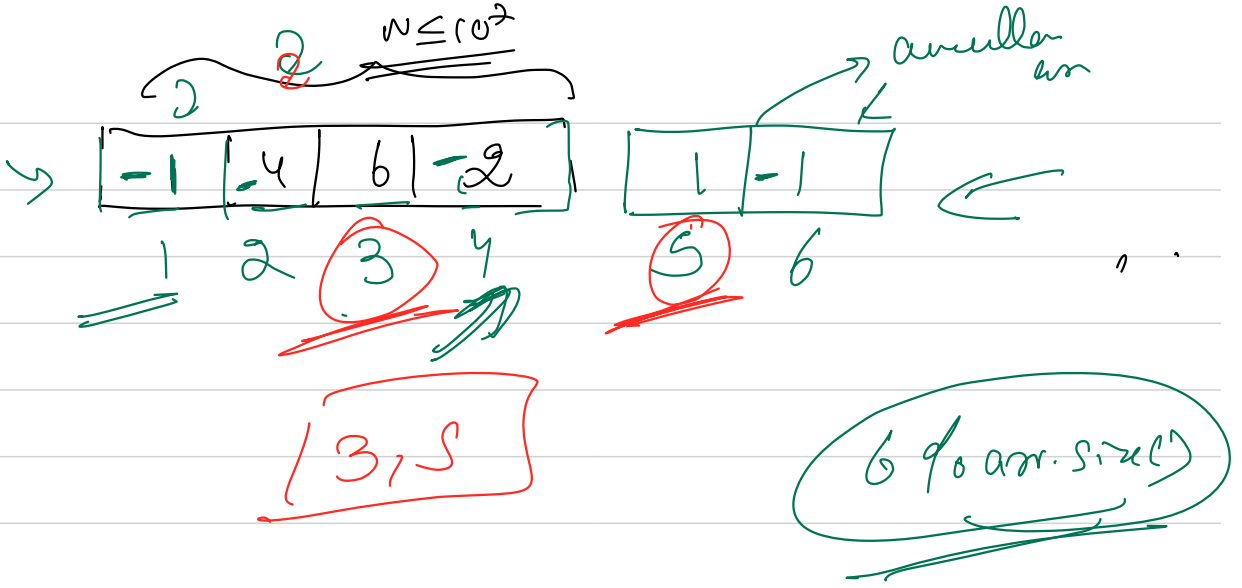
You can't take extra space

arr[1] arr[3] arr[2]

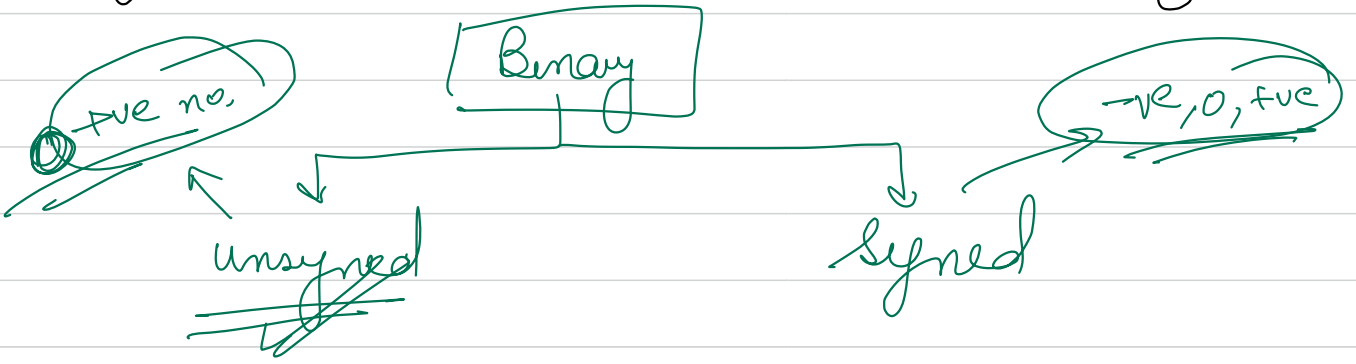
Q2 You have an array of size N, & you could get elements in the range [1 – N+2]. No element is repeated. Find the 2 missing numbers



$N \leq 10^7$
Avoid extra Space


N ≤ 10^2
2
accumulator
-1 -4 6 -2
1 -1
1 2 3 4 5 6
3, 5
6 % arr.size()

How integers are stored in memory → Binary



sizeof(int) → 16 byte → to store an integer we require

16 byte → 64 bit

32 bits in memory

bit → 0, 1

2

+ve only

0/1 0/1 0/1 - - - - - - - - - -

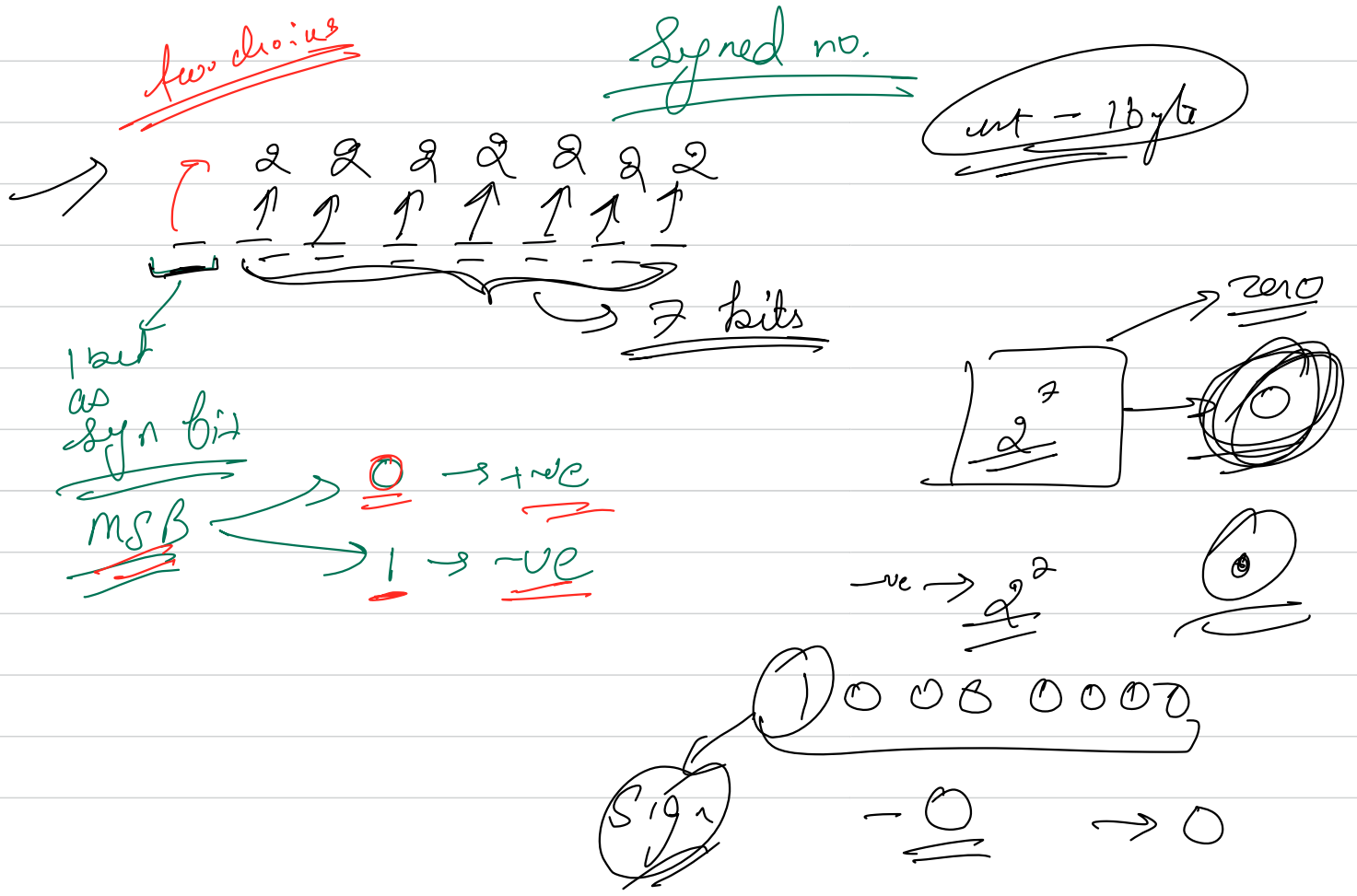
↓ ↓ ↓ - - - - -

2 2 2

8

2^{32}

1 1 1 1 1 1 1 1 1 1 1 1 1 , - - - - - - :)


two digits
Signed no.
int - 16 byte
7 bits
1 bit as sign bit
MSB
0 → +ve
1 → -ve
zero
-ve
Sign

2's complement form

→ 10101110 → 2's complement

↓ 1's complement

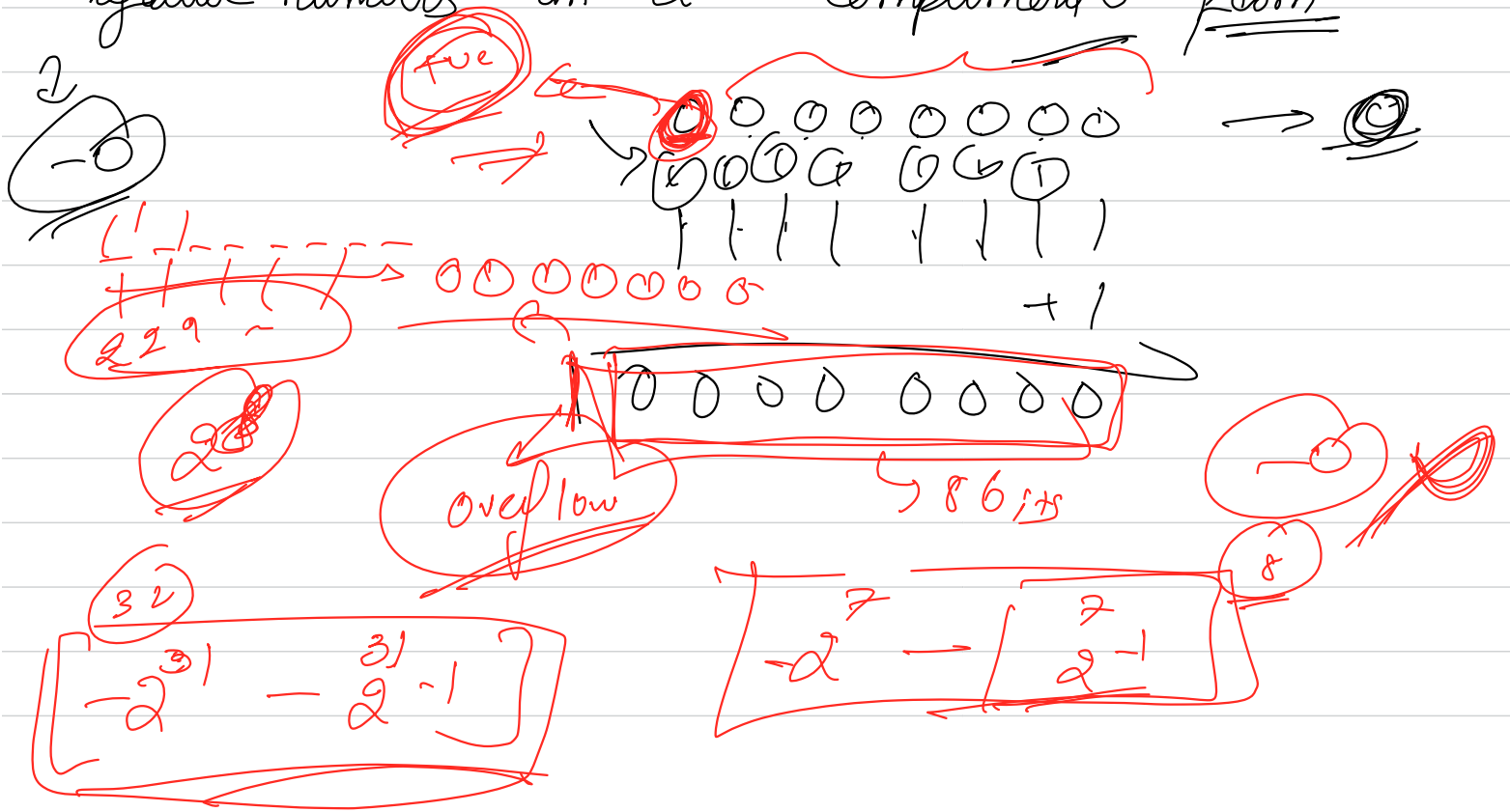
(1)

01010001

+1

01010010 → 2's complement

we represent positive numbers in binary form and negative numbers in 2's complement form



→ arrays are linear data structure

→ arrays can be initialized at compile / runtime

int arr[7]

int *arr = new int[n];

Dynamic

→ You cannot shrink or grow the array in runtime

→ Always take contiguous memory space

→ Passed by "reference in func"

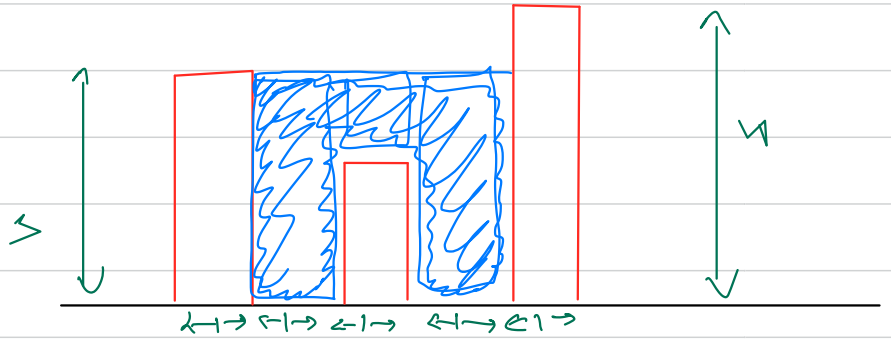
* Rain Water Harvesting

Given array of non-negative values. Each index represent height of a bar & each bar has a width 1.

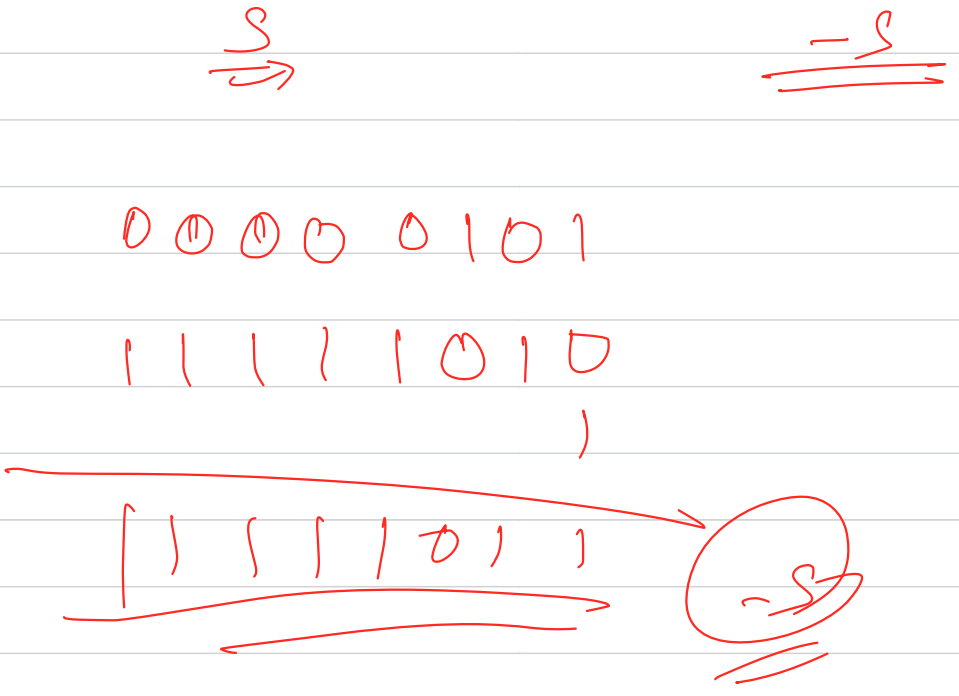
Compute how much water can be trapped in the array.

$N \leq 10^6$

3 0 2 0 4



3 + 1 + 3 ⇒ 2


S
-S
00000101
11111010
1
11111011
-S

For all elements we will try to mark the presence of the elements by going to arr[abs(arr[i])] & make the element negative.

If the element is already negative we got the repeated elem at ith idx

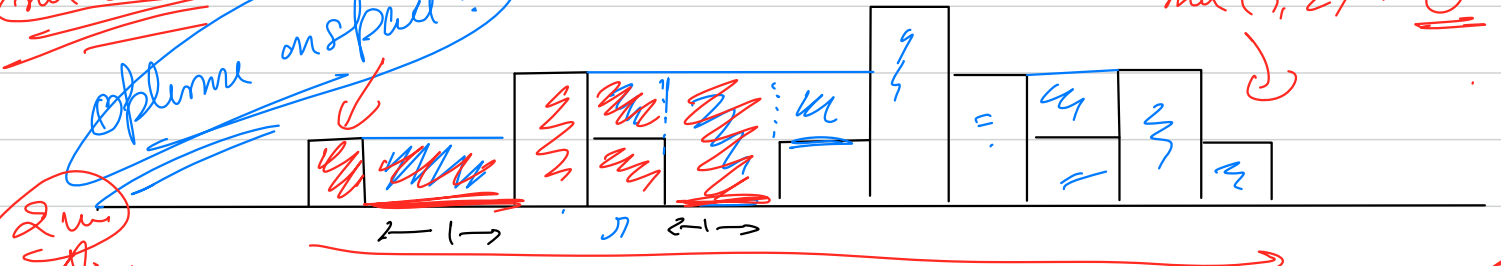
Hint

0 1 0 2 1 0 1 3 2 1 2 1

O(n)

min(right, left) = 0

Volume on spot ??



min(3, 2) → 1 → 1

min(3, 2) → 0 → 0

2 ptr

right	3	3	2	3	3	3	0	0	2	0	0	1
left	0	1	0	2	2	2	2	3	3	3	3	3

for each block calc the max height block to its left & to the right

$\forall i$ ans += min(right_max[i], left_max[i]) − a[i]