Q ⇒ You're given an array of integer values. Find the min cost to remove all the elements from the array. The cost to remove any one element is equal to the sum of elements at that point of time in array.

$n \rightarrow$ length of array

$n \leq 10^6$

Ex → $[\overset{\times}{4}, \overset{\times}{1}, \overset{\checkmark}{6}]$ → ① Remove 4, cost = 11

→ ② Remove 1, cost = 7

→ ③ Remove 6, cost = 6

Total cost = $\underline{\underline{24}}$ ← minimize

↳ at any point of time all the remaing elements add up to the cost.

$$\begin{array}{ccc} \times & & \times \\ [4, & 1, & 6] \end{array}$$

↳ 6, 4, 1

$$\underline{4+1+6+1+4+1}$$

↳ $\underline{\underline{17}}$

The main trick is to reduce the no. of times big elements actually participate in the $\underline{\text{cost}}$.

→ Sort the array in dec $\underline{\text{order}}$.

→ Take total Sum of array.

→ In every iteration remove the biggest element from total <u>Sum.</u>

```
...
totalSum = 0
for (i=0; i<n; i+) {
    totalSum += arr[i]        → O(n)
}
totalCost = 0
arr.sort(—)                   → O(nlogn)
for (i=0; i<n; i++) {
    totalCost += totalSum;    → O(n)
```

(6, 4, 1)

totalSum = ~~11~~ ~~5~~ ~~1~~ 0

totalCost = 0 + 11 + 5 + 1

```
    ↳ totalSum -= arr[i]
}
return totalCost;
```

Space → $\underline{O(n)}$

→ $O(n) + O(n\log n) + O(n)$

→ $O(n\log n)$

JOIN THE DARKSIDE

Q ⇒ You're given an array of size n, with all integer values. Apart from it you're given a value p, which is also integer. (p will be present in the array also) You need to rearrange the array such that all the elements less than p goes to the left of p & remaing goes to the right.

Solve it in better than O(nlogn) complexity.

O(1)

Space

Ex ⇒ [9, 6, 3, 1, 4, 8] $p = 4$

↳ $[\underbrace{1, 3}\ 4,\ \underbrace{8, 9, 6}]$

this can be randomly arrays

less than p   p   greater than p

two ptr

↑ i → maintain partition

2i

3   1   4   6   8   9

↑ j

p = 4

pivot

O(n)   O(1)

partition algo

5 4 3 2 1

p=4 2 3 1 4 5

p= 1 1 3 2 4 5

p=3 1 2 3 4 5

f (arr, start, end) ⇒ i = partion (arr, p)

f (arr, start, i-1)

f (arr, i+1, end)

Quicksort

$p = 9$

$[\ 9,\ 6,\ 1,\ 4,\ 8,\ 3\ ]$

n

$\hookrightarrow [\ 6,\ 1,\ 4,\ 8,\ 3,\ 9\ ]$

$p = 8$

$(n-1)$

$\hookrightarrow [\ 6,\ 1,\ 4,\ 3,\ 8,\ 9\ ]$

$p = 6$

$[\ 1,\ 4,\ 3,\ 6,\ 8,\ 9\ ]$

$n-2$

$\vdots$

$\downarrow$

p = largest elmt

$$\left(n + (n-1) + (n-2) + (n-3) \ldots\ldots \right)$$

$$\downarrow$$

$$O(n^2)$$

$k = 4$

$[\,9 \quad 6 \quad 1 \quad 4 \quad 8 \quad 3\,]$

n

$+\ n$

$[\,3,\ 1,\ 4,\ 9,\ 8,\ 6\,]$

$k = 8$

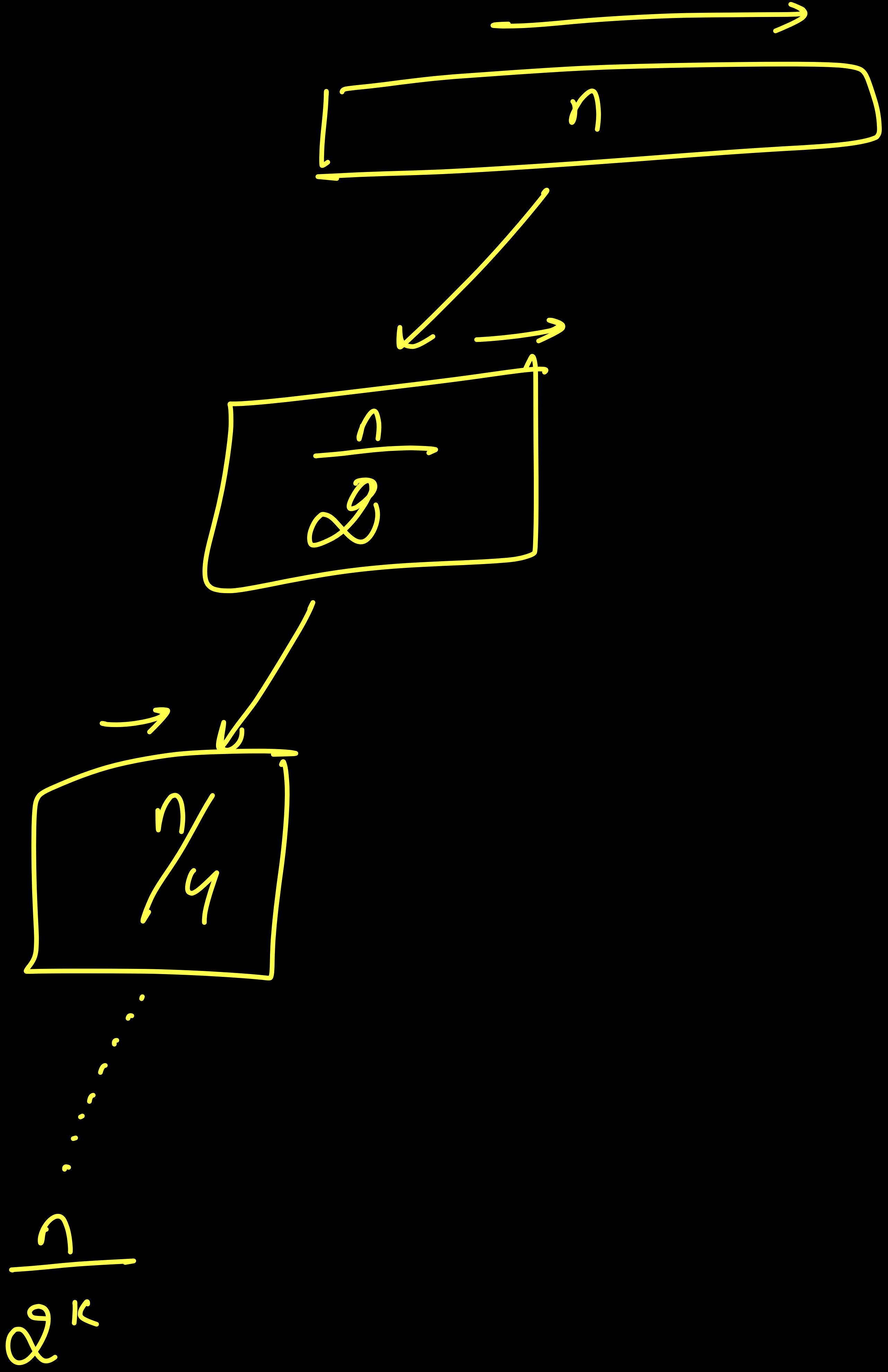
$\hookrightarrow (n \log n)$

Randomise Q.S → $\Theta(n\log n)$

→ $O(n^2)$

Space → merge → $O(n)$

→ quick → $O(\log n)$


n
n/2
n/4
n/2^k

$$\frac{n}{2^k} = 1$$

$$\underline{\underline{\left(k = \log_2 n \right)}}$$

$$\begin{tikzcd} \boxed{Q.S} \arrow[d, "\times"] \\ n\log n \end{tikzcd} \qquad \begin{tikzcd} MS \arrow[d] \\ n\log n \end{tikzcd}$$

H.W

Binary Search

↳ # Search Space → It is either a collection of elem: that are present within which we want to find an element.

↳ Binary Search → It is going to divide your search space into two equal halves. Based on some property we find how the 2 halves are diff then, we discard one half & accept one half & then repeat the process.

Play a Game ← Guess My Birthdate: 365 questn

→ 1st questn → is my birthday in H_1 or H_2

ans ⇒ H_1

2nd guess → is my birthday in Q_1 or Q_2

ans → Q_1

3rd que → is my birthday before 14th feb or not

ans → after

4th que → is my b.day after 8th march ?,?

ans → yes

5th que → is my b.day before 15th march

ans → yes

6th que → is my b.day b·w 13th of ou

ans → Yes

7th Qu → is my b·day on 06 after 10th mess

↳ after

Q: You are given an array which is arranged in ASC order. You have been given an element x, find the index at which x is present & if it is not present return -1.

$$\begin{array}{ccccccccc} & & & & & \downarrow hi & \downarrow lo & & \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ [\,1, & 3, & 9, & 11, & 16, & 18, & 22, & 27\,] & \\ & & & & & & \uparrow mid & & \end{array}$$

$\boxed{x = 19}$

$[lo, hi] \to \underline{S.S}$

Search space is the array <u>only</u>

mid point → $\frac{7+0}{2} \to \frac{7}{2} \to \boxed{\underline{3}}$

```
while (lo <= hi) {
    mid = (lo+hi)/2
    if (arr[mid] == x) {
        return mid;
    } else if (arr[mid] < x) {
        lo = mid+1
    } else {
            hi = mid-1          // discard right half
        }
    }
    return -1;
```

$$n \xrightarrow{1} \frac{n}{2^1} \xrightarrow{2} \frac{n}{2^2} \xrightarrow{3} \frac{n}{2^3} \cdots\cdots \xrightarrow{k} \underline{\underline{\frac{n}{2^k}}}$$

n ← size of
search
space

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

taking $\log_2$ on both side

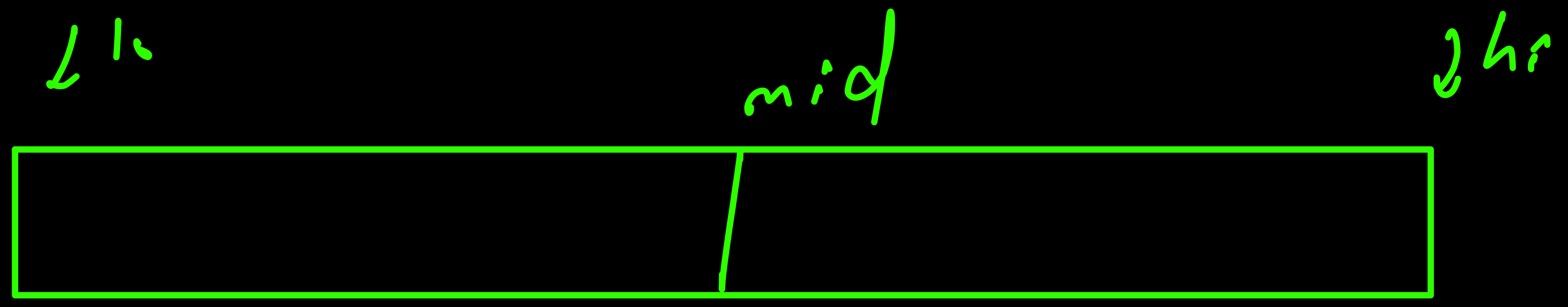
$$\boxed{k = \log_2 n}$$

$$T.C = \underline{\underline{O(\log n)}}$$

$n = 10^3$ → l.s ↓ 10^3 items   B.S ↓ 10

$n = 10^6$ → l.s ↓ 10^6 B.S ↓ 20

$n = 10^9$ l.s ↓ 10^9 B.S ↓ 30



$$mid = \frac{(lo + hi)}{2}$$

sum can overflow

add and sub lo in the numerator

$$\frac{lo + hi + lo - lo}{2}$$

$$\frac{2lo + (hi - lo)}{2}$$

$$\frac{\cancel{2}lo}{\cancel{2}} + \frac{(hi - lo)}{2} \Rightarrow lo + \frac{(hi - lo)}{2}$$

(ASC)

Q→ You are given a sorted integer array of size n. You are have an element x.

Return the index of first value $>= x$.

↳ lower_bound

Ex → $[\overset{0}{1}, \overset{1}{2}, \overset{2}{2}, \overset{3}{4}, \overset{4}{5}]$

$x = 2$ ans → $\underline{\underline{1}}$

$x = 3$ ans → 3

$x = -1$ ans → 0

$x = 7$ ans → length of array → $\underline{\underline{5}}$

Brute force → linear Search → $O(n)$

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

Case-1

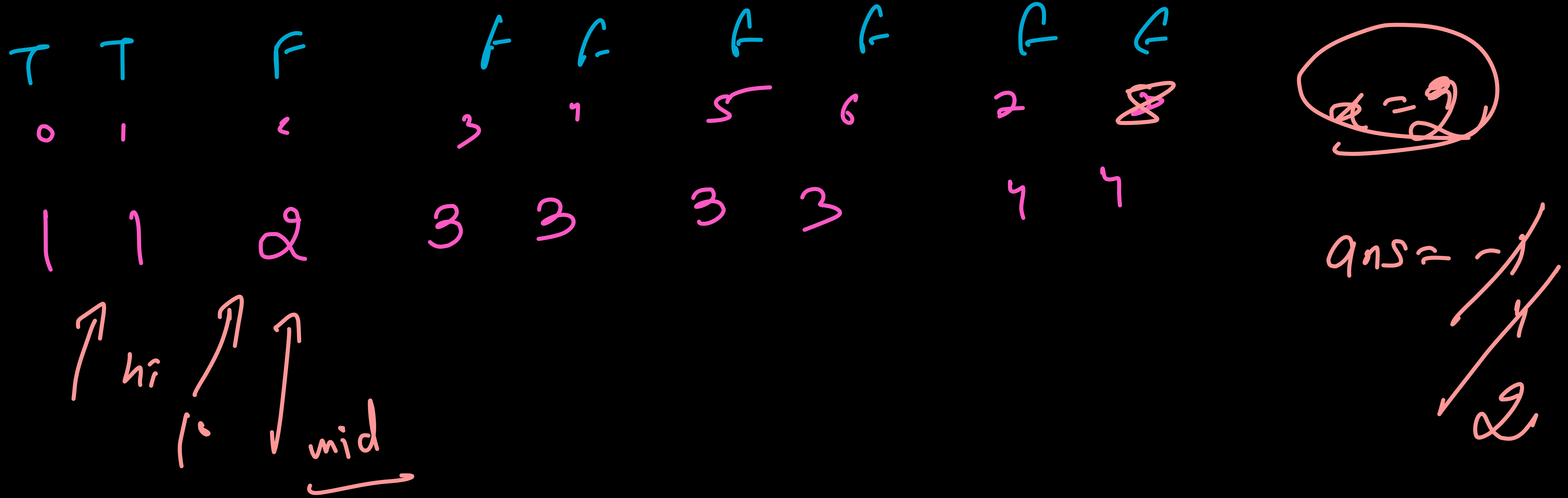
$x = 2$

↑ mid

1 1 2 2 2 3 3 4

↑ mid

```
lo = 0     hi = n-1
ans = -1;
while( lo <= hi) {

    mid = lo + (hi-lo)/2

    if ( arr [mid] < x) {
        lo = mid+1

    } else {    // arr [mid] >= x)
        ans = mid ;  ⟶  candidat

        hi = mid-1

}
```



$t < x$

lower bound → first value $\geq x$

$< x$ → ready

upper bound → first value $> x$.

$\leq x$? value

0 1 2 3 4 5 6

1, 1, 2, 2, 3, 4, 5

$x = 2$ → upper bound → 4 → 3