Majority Element

↳ Hashing (Dictionary)

- Tc → O(n)
- Sc → O(n)
  - ↳ Hashtable → (key, value)

freq. ↑

n = 7

↳ Majority Element

- > n/2
- > 7/2
- > 3

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

- 2 → 4 * → 2
- 1 → 3

Boyer-Moore Voting Algorithm

nums → [2, 2, 1, 1, 1, 2, 2]

0 1 2 3 4 5 6

2 → 4 > 7/2

candidate = ~~None~~ ~~X~~ ~~X~~ 2

count = 0 ~~X~~ ~~X~~ ~~X~~ 0 ~~X~~ ~~X~~ ~~X~~ 0

1

0	1	2	3	4
2	3	4	3	3

3 → 3 > 5/2

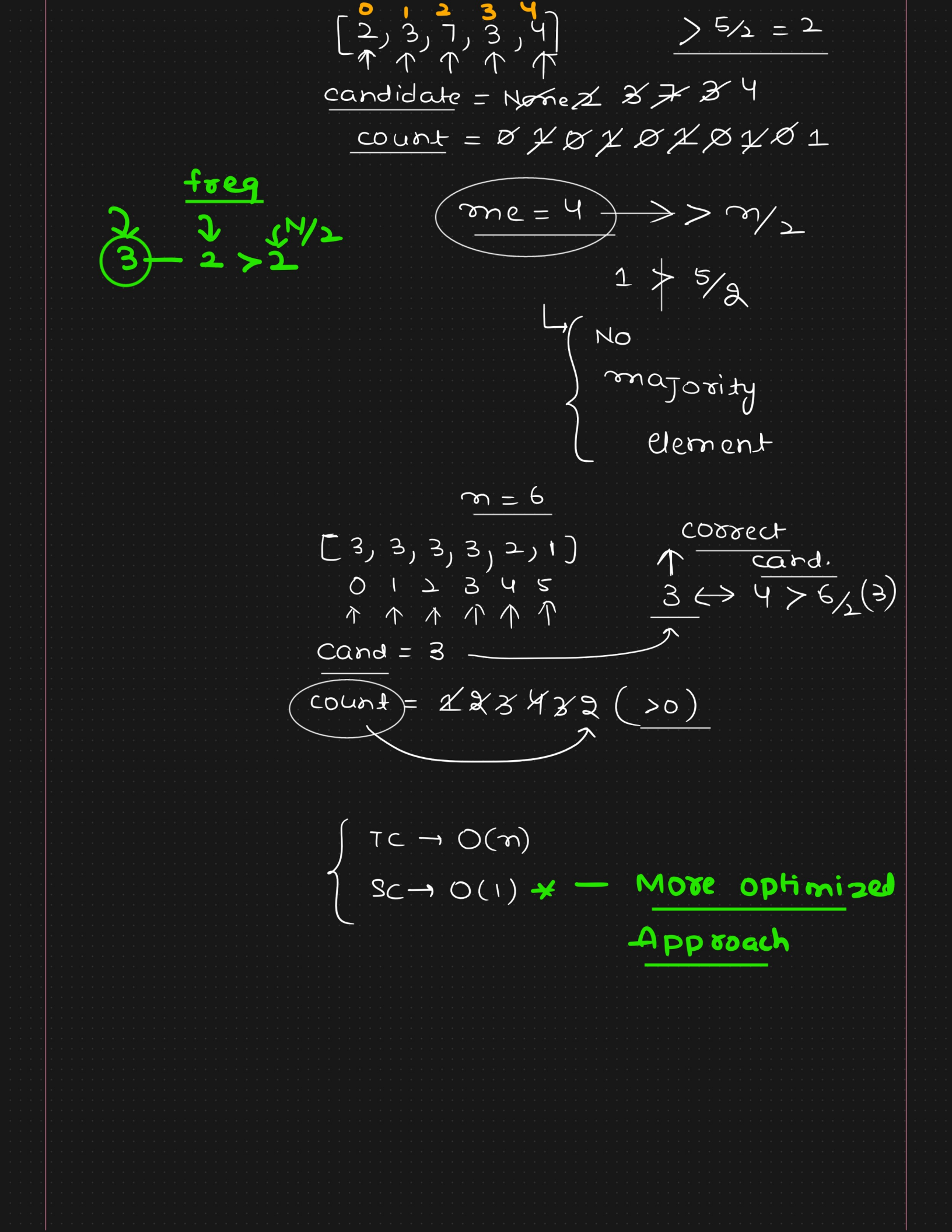
↳ Yes

↓

3

candidate = ~~None~~ ~~X~~ ~~3~~ ~~4~~ 3

count = ~~0~~ ~~X~~ ~~0~~ ~~X~~ ~~0~~ ~~X~~ ~~0~~ ~~X~~ 1 2


0 1 2 3 4
[2, 3, 7, 3, 4]
> 5/2 = 2
candidate = None 2 3 7 3 4
count = 0 1 0 1 0 1 0 1 0 1
freq
3
2 > 2
N/2
me = 4
> n/2
1 ≯ 5/2
No majority element
n = 6
[3, 3, 3, 3, 2, 1]
0 1 2 3 4 5
correct card.
3 ↔ 4 > 6/2 (3)
Cand = 3
count = 1 2 3 4 3 2 (>0)
TC → O(n)
SC → O(1) *
* — More optimized Approach

Count of number of inversions in an array

inversion { i < j — (1)
nums(i) > nums(j) — (2)

0	1	2	3	4	5	6	7
70	50	60	10	20	30	80	15

n = 8

i < j & nums(i) > nums(j)

4 Inversion

70 → 50, 60, 10, 20, 30, 15

50 → 10, 20, 30, 15

60 → 10, 20, 30, 15

10 → X

20 → 15

30 → 15

80 → 15

15 → X

70	50
0	1

Inversion → { 0 < 1
70 > 50

Count of inversions = 17

2 for loops (Brute force approach)

↳ $\Theta(n^2)$

Divide & conquer Approach

nums → (70, 50, 60, 10, 20, 30, 80, 15)

0 1 2 3 4 5 6 7

70 — 15, 20, 30 — 3
10 — x
50 — 15, 20, 30 — 3
60 — 15, 20, 30 — 3

Small Problem
⇓
Single element
⇓
inversion = 0

Recursive Tree

C1 — 17

nums, 0, 7, 5+3+9

10, 15, 20, 30, 50, 60, 70, 80

10, 50, 60, 70

m = (0+7)/2

C2 — nums, 0, 3, 1+1+3 = 5

C9 — 20 − 15, 30 − 15

m = 4+7/2 = 5

15, 20, 30, 80

nums, 4, 7, 0+1+2 = 3

80 C14 nums, 6, 6 — 0

C13 — nums, 6, 7, 0+0+1

50 − 10, 70 − 10, 60

mid = 0+3/2 = 1

50, 70 C3 — nums, 0, 1, 0+0+1

10, 60 C6 — nums, 2, 3, 0+0+1

20, 30 C10 — nums, 4, 5, 0+0+0

15, 80

0 C12 nums, 5, 5 — 30

C15 nums, 7, 7 — 0

C11 nums, 4, 4 — 20

10

C4 → 70 — nums, 0, 0, 0

C5 → 50 — nums, 1, 1, 0

nums, 2, 2, 0

60

C7

nums, 3, 3, 0

C8

↑10

1

15

C13

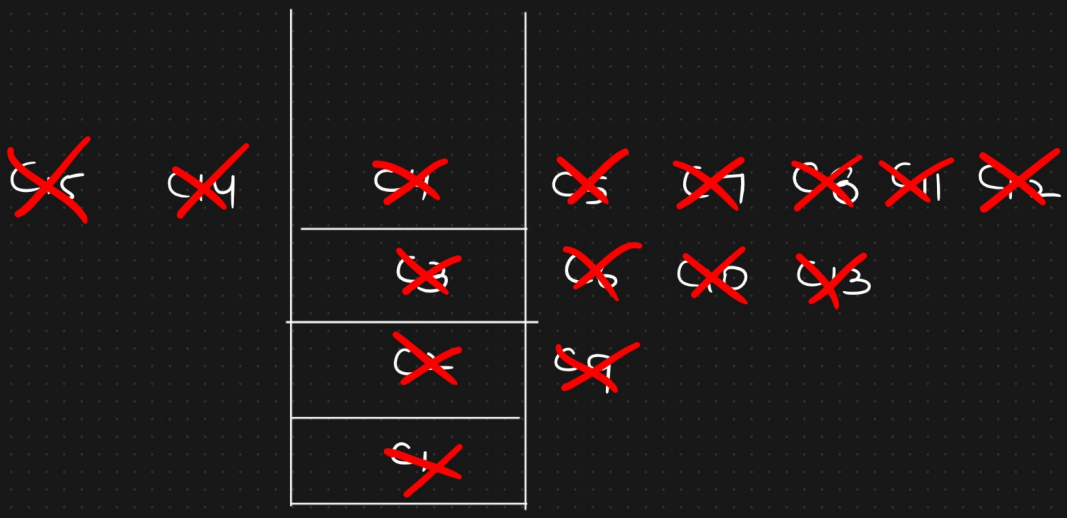
Left → 80

Right → 15

70 → 50 (valid inversion)

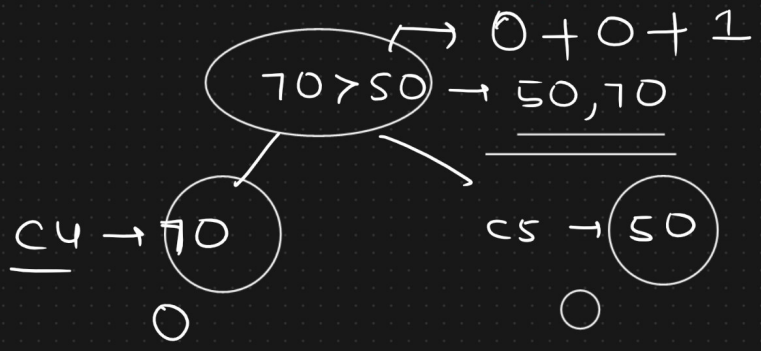
60 → 10 (↓)

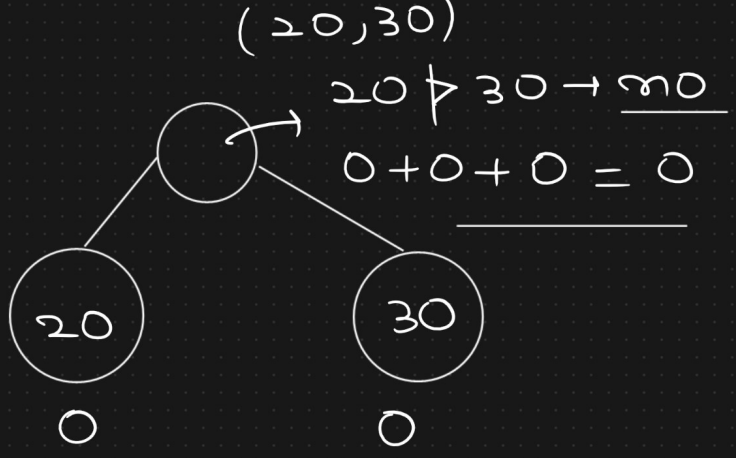
Stack → Recursion

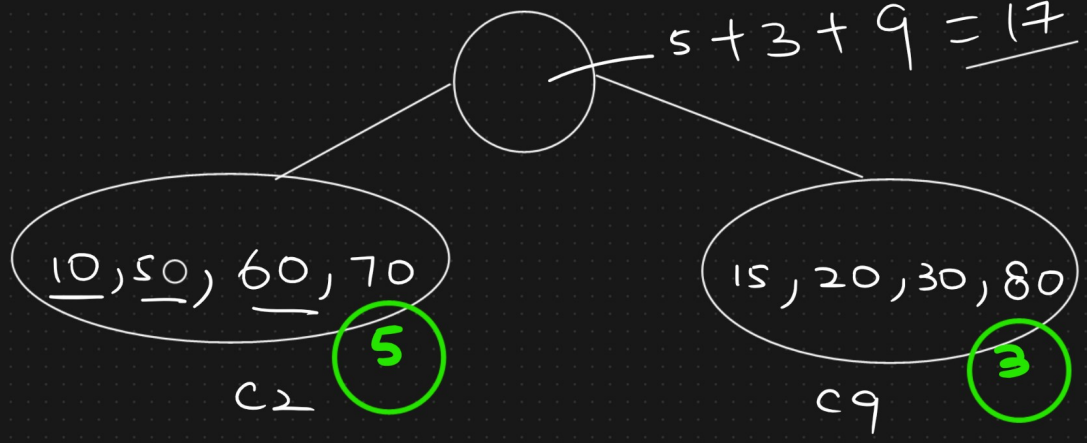


Left side right side

nums(i) > num(j)







10 → X

50 → 15, 20, 30 (3)

60 → 15, 20, 30 (3)

70 → 15, 20, 30 (3)

Pseudocode

T(n) ↳

```
countInversion(nums, i, j):
    if i == j:                    → Small problem
        cnt = 0                     ↓↓
        return nums(i)              Single element
    else:                         ← Divide & conquer
        mid = i + (j-i)/2
        cntL = countInversion(nums, i, mid)      T(n/2)
        cntR = countInversion(nums, mid+1, j)    T(n/2)
```

c

Left subtree

Right subtree

finalCnt = mergeProcedure(nums, l, mid, r)

finalCnt = cntL + cntR + currCnt

return finalCnt

n

Recurrence Relation

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= \Theta(n \log n)$$

Brute force approach	Divide & conquer approach
$\Theta(n^2)$	$\Theta(n \log n)$
	↓
	Optimized

Strassen's Matrix Multiplication

Matrix multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\times5+2\times7 & 1\times6+2\times8 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix}$$

Three for loops

↳ $\Theta(n^3)$

Brute fore approach

Divide & conquer approach

↳ $$T(n) = 7T\left(n/2\right) + n^2$$

$$T(n) = O(n^{2.8074})$$

(2) nums1 = [1, 2, 4)

nums2 = (3, 5, 7)

nums → [1, 2, 3, 4, 5, 7]
(indices: 0 1 2 3 4 5)

median → 1) sort the array

2) odd → 4 (middle value)

even → (3+4)/2

= 3

(4) Dividend = 9

Divisor = 3

9/3 = 3 Not allowed

Mod, Divison, multiplication