Hashmap

27 August 2023 13:05

Stores key value pair.

key	value
food item	price
maggi	50
coke	40
pizza	100
<String>	<int>

frequency

[1, 2, 5, 2, 1, 3, 6]

key	value
1	2
2	2
3	1
5	1

<int, int>

Time: O(1)

First unique char in a string

27 August 2023 13:11

0 1 2 3 4 5 6 7 8 9

str = leetcode

2 for loops.

ans = 4 'c'

str = racecare

ans = -1 'x'

ans = 'v' 2

Time: O(N^2), Space: O(1)

O(N)

str = loveleetcode

ans = 2

freq.

l - 2
o - 2
v - 1
e - 3
t - 1
c - 1

Time: O(N)

O(1)

Time - O(N)

Space - ~~O(N)~~ O(1)

'a' to 'z'

c - 1

d - 1

26

Two sum

27 August 2023 13:30

index ?

target = 17

0 1 2 3 4 5

arr = { 3, 1, 5, 8, 10, 7 }

10, 7 index = [4, 5]

① Two loops. $O(N^2)$

② Sorting + 2 pointer $O(N \log N)$

③ Hashmap $O(N)$

i i i i i i

arr = { 3, 1, 5, 8, 7, 10 }

target = 17

[i, map[diff]]

diff = target − curval

17 − 3 = 14

17 − 1 = 16

17 − 10 = 7

Map

values, arr	index
3	0
1	1
5	2
8	3
7	4

Time : $O(N)$

Time : O(N)

Space : O(N) → map

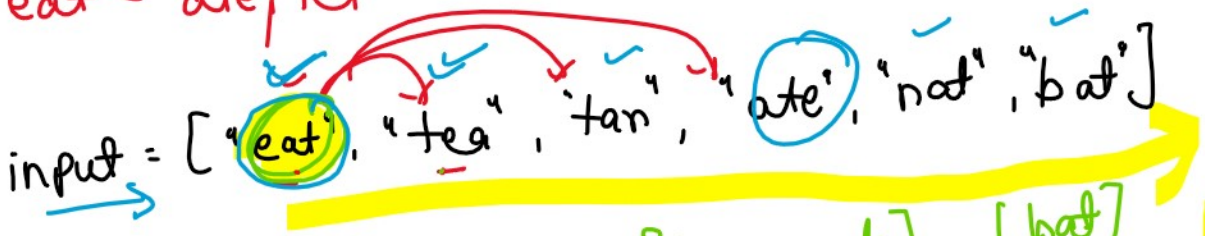
Group anagrams

27 August 2023 13:49

State - taste

cat - act

eat - ate/tea



ans = [eat, tea, ate], [tan, nat], [bat]

M

M = (N * m) ≈ O(N^3)

HashMap<String, List<String>>

key will be sorted string.

sorted

aet → [eat, tea, ate]

ant → [tan, nat]

abt → [bat]

⑤ O(N log N)

O(M * N log N)

sorting / word

"leetcode" ← N

O(N log N)

bb = 196

ac = 196

③

ait, ate, tea

[eat]

aet → [eat ___

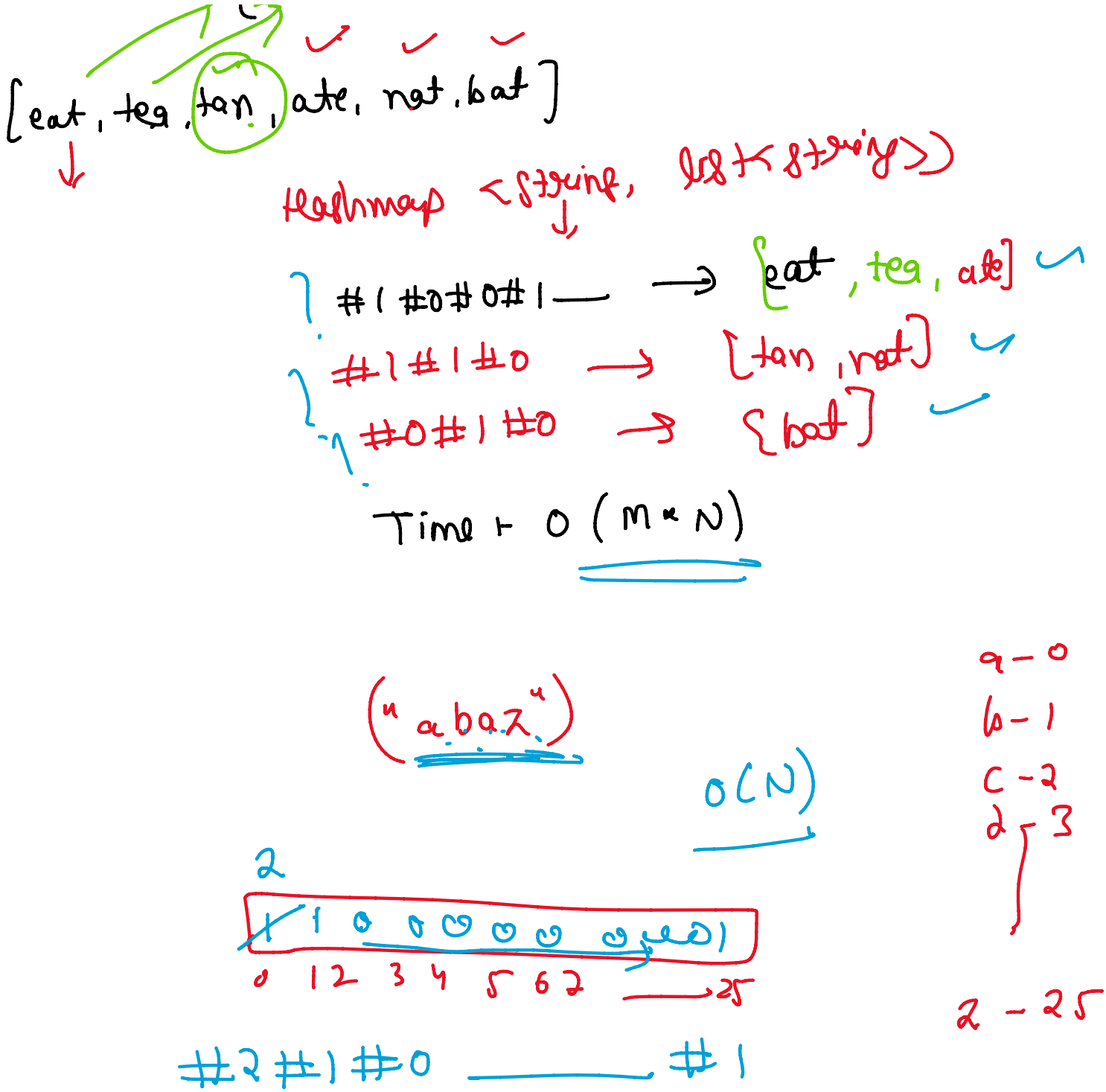
length

freq of each char.

| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

0 1 2 3 4 5 6 ___ 25

[#1#0#0#0___#1#0#0 #1 ___]


[eat, tea, tan, ate, nat, bat]
HashMap <String, list<String>>
#1#0#0#1 → [eat, tea, ate]
#1#1#0 → [tan, nat]
#0#1#0 → [bat]
Time : O(M * N)
("abaz")
O(N)
a-0
b-1
c-2
2 - 3
2 - 25
2
1 0 0 0 0 0 0 1
0 1 2 3 4 5 6 2
25
#2#1#0 #1

Subarray sum equals k

27 August 2023 14:28

[3 4 7 2 -3 1 4 2 1] k = 7

ans = 6

[3,4]
[7]
[4,2,1]
[1,4,2]
[7,2,-3,1]
[2,-3,1,4,2,1]

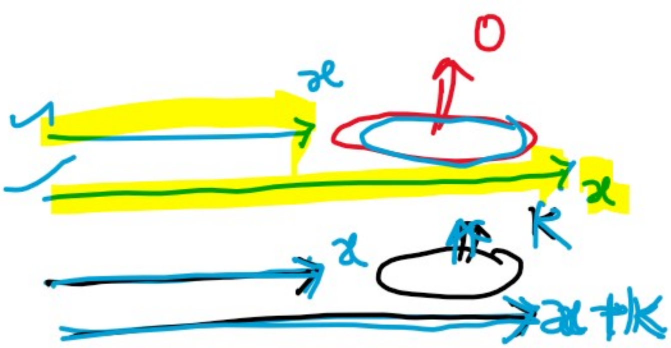
~~O(N²)~~ ← generating all subarrays.

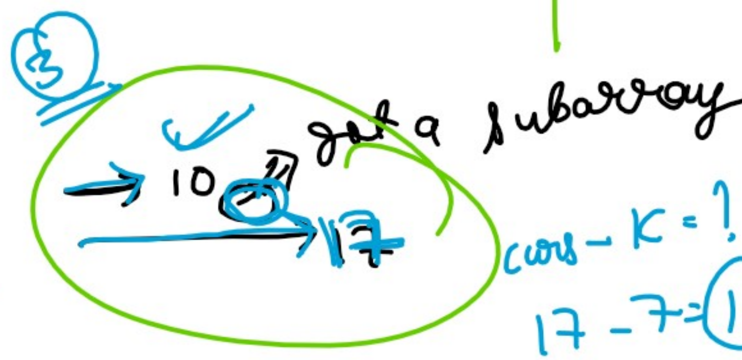
O(N)?

17 - 7 = 10

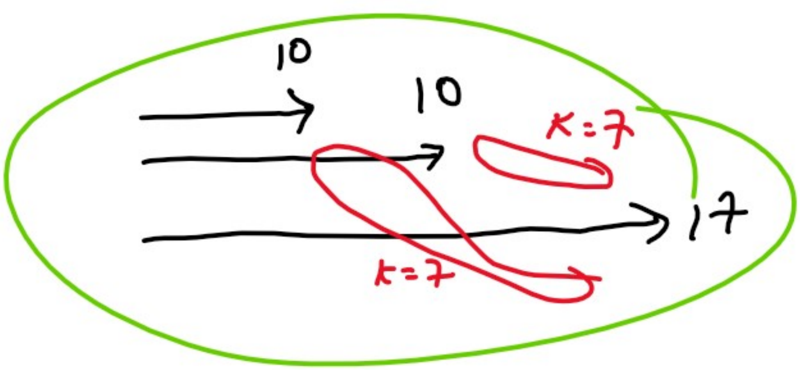
Map

prefixSum	Count
10	2



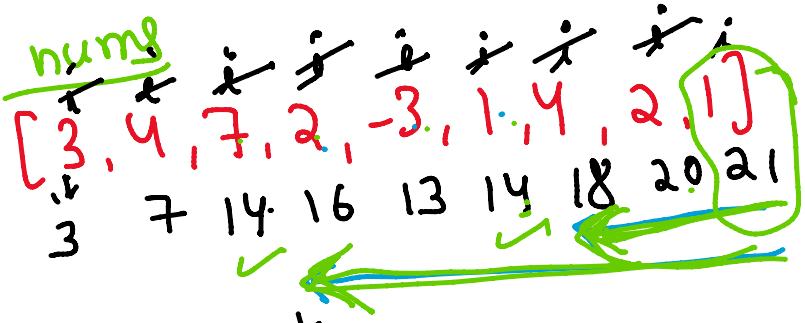


curr - K = ?
17 - 7 = 10



2 subarrays.

5 + 5 + 0 - 5 + 5 + 7



K = 7

Hashmap

prefixSum - count

0 - 1

3 - 1

7 - 1

14 - 2

16 - 1

13 - 1

18 - 1

20 - 1

prefixSum - K

3 - 7 = -4

7 - 7 = 0

14 - 7 = 7

16 - 7 = 9

13 - 7 = 6

14 - 7 = 7

18 - 7 = 11

20 - 7 = 13

prefixSum = K

21 - 7 = 14

count = 0 1 2 3 4 6

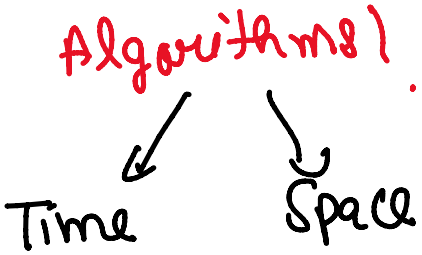
[3,4], [7], [7,2,-3,1], [1,4,2], [4,2,1], [2,-3,1,4,2,1]

Time: O(N)

Space: O(N)

Time and space complexity

27 August 2023 15:21



$f(n) = 7n^2 + 5n$

$= O(N^2)$

$f(n) = \frac{n}{5} + \log N$

$= O(N)$

$f(n) = 5n^4 + 2n^3 + n^2$

$= O(N^4)$

$f(n) = 2n + 100 \log N$

$= O(N)$

→ Bubble sort $O(N^2)$

→ Binary search $O(\log N)$

→ Recursion

→ Merge sort $O(N \log N)$

Bubble sort

27 August 2023 15:25

Time : $O(N^2)$

Space : $O(1)$

pairwise comparison and move the largest element to the end

(N)

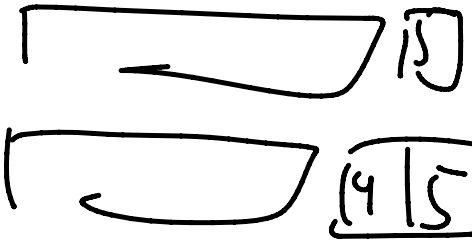
$N-1$

$N-2$

$N-3$

⋮

1



k.

$= ((1+2+3+ \dots (N-2)+(N-1))k$

$= \left(\frac{N(N-1)}{2}\right)k$

$= \frac{N^2 k}{2} - \frac{N}{2}k$

$= O(N^2)$

Recurrence relation

$T(n) = k.(n-1) + T(n-1)$

$T(n-1) = k.(n-2) + T(n-2)$

$T(n-2) = k.(n-3) + T(n-3)$

⋮

$T(1) = k + 0$

$T(n) = k\left((n-1) + (n-2) + \cdots + 2 + 1\right)$

$= \frac{n(n-1)}{2} \times k$

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

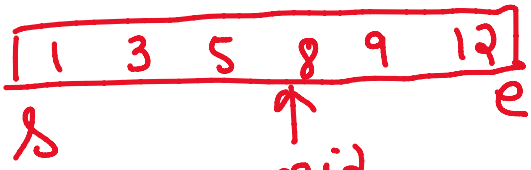
$= O(N^2)$

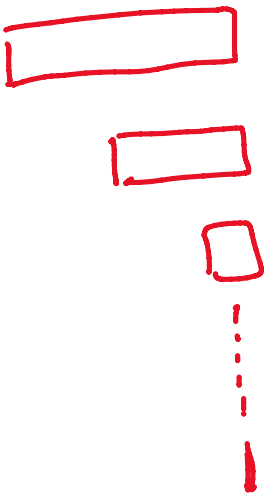
Binary Search

27 August 2023 15:34

Time : $O(\log N)$

Space : $O(1)$





(n) $(n/2)$ $(n/4)$ $(n/8)$ $\cdots$ n $n/2$ $n/4$ $n/8$ $\cdots$ 1

$\frac{N}{2^0}$ $\frac{N}{2^1}$ $\frac{N}{2^2}$ $\frac{N}{2^3}$ $\cdots$ $\frac{N}{2^k}$

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

$$2^k = N$$

$$\log_2 2^k = \log_2 N$$

$$k = \log N$$

Time : $O(\log N)$

Recurrence

$$T(n) = k + T(n/2)$$

$$T(n/2) = k + T(n/4)$$

$$T(n/4) = k + T(n/8)$$

$T(n/4) = k + T(1/8)$

$\|$

$\vdots$

$T(1) = k + 0$

$$T(n) = \sum_{i=0}^{\log N} k \quad \Rightarrow \quad k \log N$$

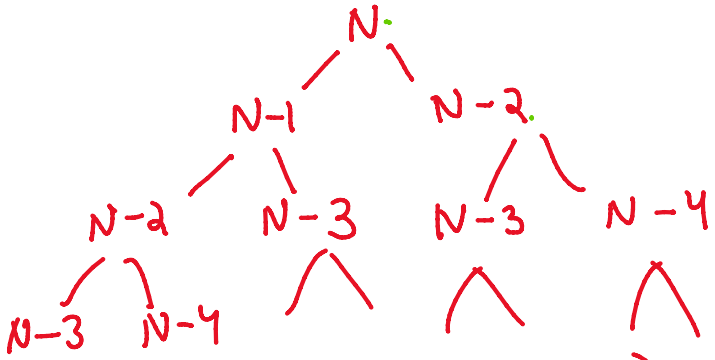
$T(n) = \log N$ $\qquad$ $O(\log N)$

Fibonacci series

27 August 2023 15:40

$fib(n) = fib(n-1) + fib(n-2)$

0 1 1 2 3 5 8 13 .....



1	2^0
2	2^1
4	2^2
8	2^3
	2^4
16	
⋮	
	2^{N-1}

$= K(1+2+4+8+ \cdots + 2^{N-1})$

$= K\frac{(2^N-1)}{2-1}$

$= K(2^N) = O(2^N)$ ← DP / 3 variables.

$O(N) \rightarrow$ space

① 3 variables

~~a~~	~~b~~	~~a~~	~~b~~	~~a~~	~~b~~	c
~~a~~	~~b~~	~~c~~	~~a~~	~~b~~	~~c~~	c
0	1	1	2	3	5	

$c = a + b$

Time → $O(N)$

Space → $O(1)$

② DP (extra array)

Time → $O(N)$

Space → $O(N)$

$dp[i] = dp[i-1] + dp[i-2]$

~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	~~i~~	i		
0	1	1	2	3	5	8	13	21
0	1	2	3	4	5	6	7	8

③ Using recursion.

Time → $O(2^N)$

Space → $O(N)$

Recurrence

$$T(N) = T(N-1) + T(N-2) + K$$

$$T(N) = 2T(N-1) + K$$

$$2T(N-1) = 4T(N-2) + 2K$$

$$4T(N-2) = 8T(N-3) + 4K$$

$$\vdots$$

$$T(1) = \cdots + 2^{N-1}K$$

$$T(N) = K + 2K + 4K + 8K + \cdots + 2^{N-1}K$$

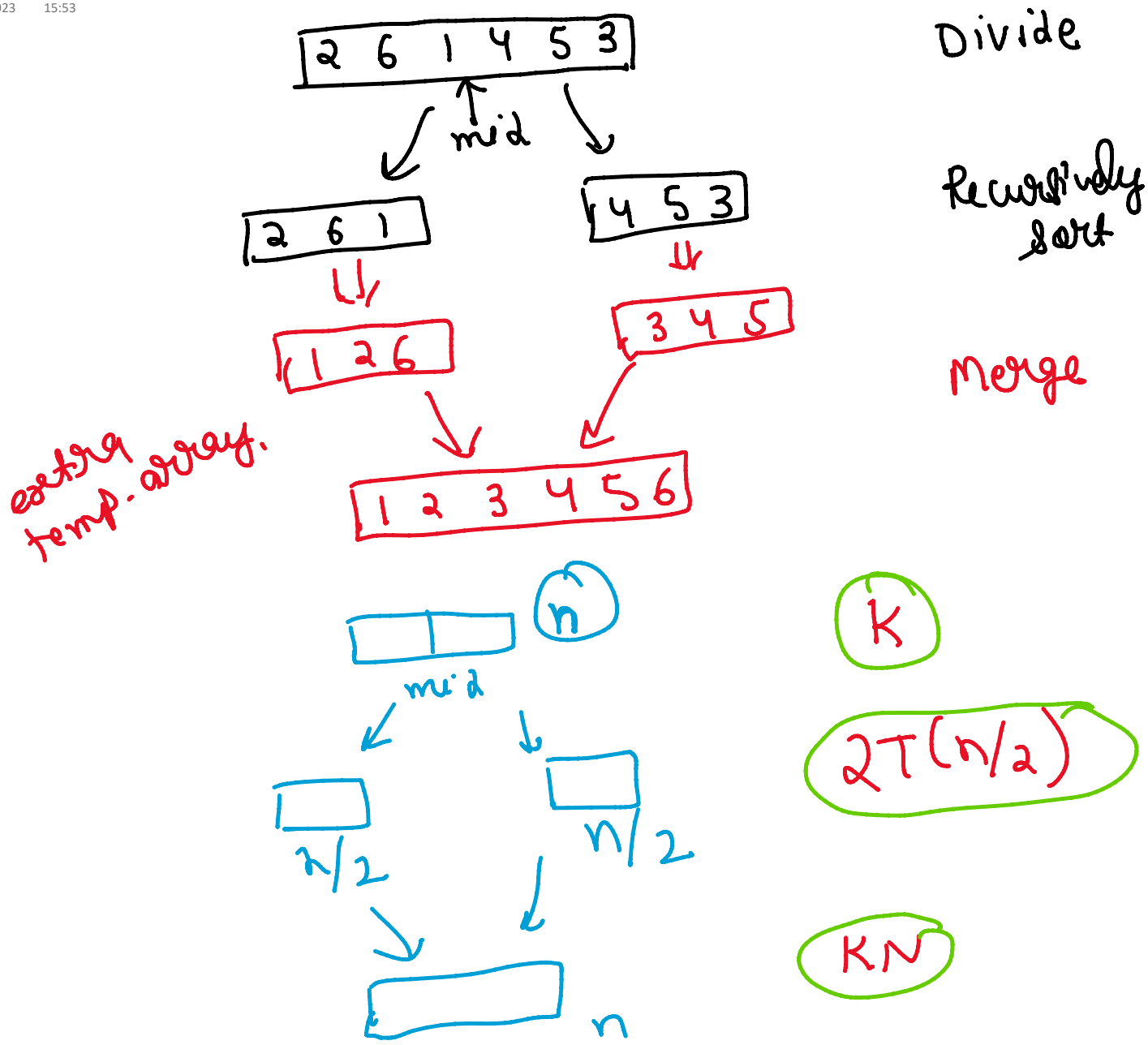
$$= K(1 + 2 + 4 + 8 + \cdots + 2^{N-1})$$

$$= K\left(\frac{2^N - 1}{2 - 1}\right)$$

$$= O(2^N)$$

Merge sort

27 August 2023 15:53



$$T(n) = K + 2T(n/2) + KN$$

$$T(n) = K'N + 2T(n/2) \rightarrow n/2$$

$$2T(n/2) = 2K'\frac{N}{2} + 4T(n/4) \rightarrow n/4$$

$$4T(n/4) = 4K'\frac{N}{4} + 8T(n/8) \rightarrow n/8$$

$4T(n/4) = 4k'\frac{N}{4} + 8T(n/8)$ → n/8

→

$T(1) = k'N + 0$ — 1

$$T(n) = \sum_{l=0}^{\log N} k'n$$

$$= k'N \log N$$

$$= O(N \log N)$$

$O(N)$

Space

Recursive → $O(\log N)$

merging (temp array)

$O(N)$

$\log N + N$ = $O(N)$