

## SORTING I

### Contents

- Basics of sorting { stable / unstable  
Inplace sorting }
- $k^{\text{th}}$  smallest element
- Selection sort
- Merge two sorted arrays.
- Merge sort
- Inversion Count.

**Sorting** : Arranging the data in a defined order  
using some parameter

1	2	3	5	7	→ Asc	→ para → value
9	6	3	2	1	→ DSC	—  —
1	7	2	9	24	→ Yes	→ para →
↓	↓	↓	↓	↓		no. of factor .
1	2	2	3	8		

Why sorting ?

- It makes data easier to find
- —||— organised.

Stable / Unstable sorting

If two data points have same parameter value.

Stable sort → Order is preserved.

Unstable sort → Order is not preserved.

$$\begin{aligned} A[] &= 1 \ 5 \ 2 \ 6 \ 2 \ \{GR\} \\ &= 1 \ 2 \ 2 \ 5 \ 6 \quad 1 \ 2 \ 2 \ 5 \ 6 \\ &\qquad \{RG\} \qquad \{GR\} \\ &\qquad \text{unstable} \qquad \text{stable} \end{aligned}$$

Sort the data in the increasing order of their marks.

Name	Marks	Name	Marks	Name	Marks
mohit	0	mohit	0	mohit	0
aman	100	kajal	32	kajal	32
shreesh	45	shreesh	45	ayuh	45
kajal	32	ayuh	45	shreesh	45
ayuh	45	aditya	92	aditya	92
aditya	92	aman	100	aman	100

↓  
Stable

↓  
unstable .

### Inplace Sorting Algo

Sorting in the same array without extra space

$O(1)$

2 1 5 6 8

→ No additional space → inplace

## $k^{\text{th}}$ Smallest

Given  $A[N]$  elements, find  $k^{\text{th}}$  smallest element.

$$K=3$$

$$A[8] = \begin{matrix} 2 & 8 & 4 & -4 & 6 & 7 & 5 & 10 & -1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix}$$

Idea 1 : Sort the given input and return  $K-1$

$$TC : O(n \log n)$$

$$SC : O(1)$$

Idea 2 :

2	8	4	-4	6	7	5	10	-1
0	1	2	3	4	5	6	7	8

1

	-4	8	4	2	6	7	5	10	-1
1	0	1	2	3	4	5	6	7	8

2

	-4	-1	4	2	6	7	5	10	8
2	0	1	2	3	4	5	6	7	8

3

	-4	-1	2	4	6	7	5	10	8
3	0	1	2	3	4	5	6	7	8

## Pseudo code

```
int kthsmallest ( A[ ] , k ) {  
    for ( i = 0 ; i < K ; i++ ) {  
        midx = i  
        for ( j = i ; j < n ; j++ ) {  
            if ( A[j] < A[midx] ) {  
                midx = j  
            }  
        }  
        swap i and midx  
    }  
}
```

TC :  $O(nk)$

SC :  $O(1)$

## Selection sort

```
int selection ( A[ ] ) {  
    for ( i = 0 ; i < n - 1 ; i++ ) {  
        midx = i  
        for ( j = i ; j < n ; j++ ) {  
            if ( A[j] < A[midx] ) {  
                midx = j  
            }  
        }  
        swap i and midx  
    }  
}
```

Selection sort does sorting in min no. of swaps

$$= n-1$$

TC :  $O(N^2)$

SC :  $O(1)$

2 2 1 { GP }  
1 2 2 { PG }

Eg

→ 2 1 → 3 1 2  
    1 2     1 3 2  
              1 2 3

Given  $A[N]$ , we can only swap adjacent elements, sort the array in increasing order

$$\begin{aligned} A[8] &= \begin{matrix} 2 & 8 & 4 & -4 & 6 & 7 & 5 & 10 & -1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ &= \begin{matrix} 2 & 4 & -4 & 6 & 7 & 5 & 8 & -1 & 10 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \end{aligned}$$

HW. Read more on bubble sort.

Given two sorted array .  $A[N]$  ,  $B[M]$  . Create  $C[]$  which contains overall sorted data.

$$A[4] : \{ 7 \ 10 \ 11 \ 14 \} \quad A[3] : \{ 3 \ 6 \ 10 \}$$
$$B[3] : \{ 3 \ 8 \ 9 \} \quad B[2] : \{ 5 \ 9 \} .$$

Idea : Merge the array A and B into C.  
then apply sorting

$$A[4] : \{ 7 \ 10 \ 11 \ 14 \}$$
$$B[3] : \{ 3 \ 8 \ 9 \}$$
$$C[7] = 7 \ 10 \ 11 \ 14 \ 3 \ 8 \ 9$$

sort

$$C[7] = 3 \ 7 \ 8 \ 9 \ 10 \ 11 \ 14$$

Idea 2 : Take two pointers .

$$A[4] : \{ 7 \ 10 \ 11 \ 14 \}$$
$$B[3] : \{ 3 \ 8 \ 9 \}$$
$$C[7] : [ 3 \ 7 \ 8 \ 9 \ 10 \ 11 \ 14 ]$$

$\uparrow$   
 $P_1$

$\uparrow$   
 $P_2$

$\uparrow$   
 $k$

$A[3] : \{ 3 \ 6 \ 10 \}$

$B[2] : \{ 5 \ 9 \}$

$\downarrow p_1$   
 $\uparrow p_2$

$C[7] = 3 \ 5 \ 6 \ 9 \ 10$

```
int[] merge ( A[N] , B[M] ) {
```

```
    C[M+N] // init
```

```
    p1 = 0
```

```
    p2 = 0
```

```
    k = 0
```

```
    while ( p1 < N && p2 < M ) {
```

```
        if ( A[p1] <= B[p2] ) {
```

```
            C[k] = A[p1]
```

```
            k++
```

```
            p1++
```

```
}
```

```
else {
```

```
    C[k] = B[p2]
```

```
    k++
```

```
    p2++
```

```
}
```

```
}
```



```
while (p1 < N) {  
    C[k] = A[p1]  
    k++  
    p1++  
}  
  
while (p2 < M) {  
    C[k] = B[p2]  
    k++  
    p2++  
}  
}  
  
return C
```

$TC : O(N+M)$   
 $SC : O(M+N)$

Given  $A[N]$  elements & 3 indices  $s, m, e$  &  
 subarray  $[s \ m]$  is sorted  
 subarray  $[m+1 \ e]$  is sorted  
 sort subarray from  $[s \ e]$

$$A[12] = \{ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \\ 4 \ 8 \ -1 \ 2 \ 6 \ 9 \ 11 \ 3 \ 4 \ 7 \ 13 \ 0 \ 3 \}$$

$$\begin{matrix} s & m & e & C[] \\ 2 & 6 & 9 & \end{matrix} \quad \begin{matrix} -1 & 2 & 3 & 4 & 6 & 7 & 9 & 11 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$

`int[] merge ( A[N], s, m, e ) {`

`C[e-s+1] // init`

`p1 = s`

`p2 = m+1`

`k = 0`

`while ( p1 ≤ m && p2 ≤ e ) {`

`if ( A[p1] ≤ A[p2] ) {`

`C[k] = A[p1]`

`k++`

`p1++`

`}`

`else {`

`C[k] = A[p2]`

`k++`

```

    }           p2 ++
}
}

while (p1 ≤ m) {
    C[k] = A[p1]
    k ++
    p1 ++
}

while (p2 ≤ e) {
    C[k] = A[p2]
    k ++
    p2 ++
}

for (i = 0 ; i < C.length ; i++) {
    A[i+s] = C[i]
}

```

TC: O(e-s)

SC: O(e-s)

Break : 8: uu

Given  $A[N]$ , sort it without inbuilt function.

$$N = 100$$

Case I

$A[N]$



TC  
 $O(N^2)$

iterations  
 $10^4$

Case II

$A[N]$



$n/2$

$n/2$

$$\left(\frac{n^2}{2}\right) \times 2 + n = \left(\frac{100^2}{2}\right) \times 2 + 100 = 5100$$

Case III

$A[N]$



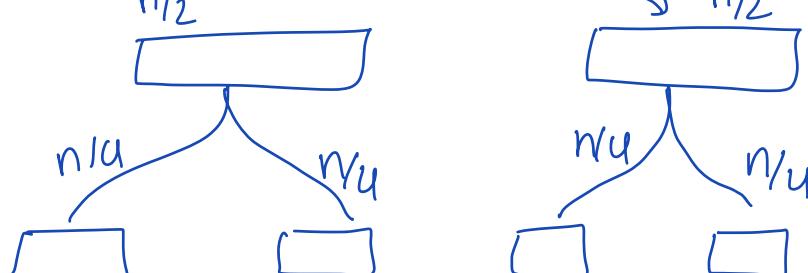
$n/2$

$n/2$

$$\left(\frac{n}{4}\right)^2 \times 4 + 2 \times n$$

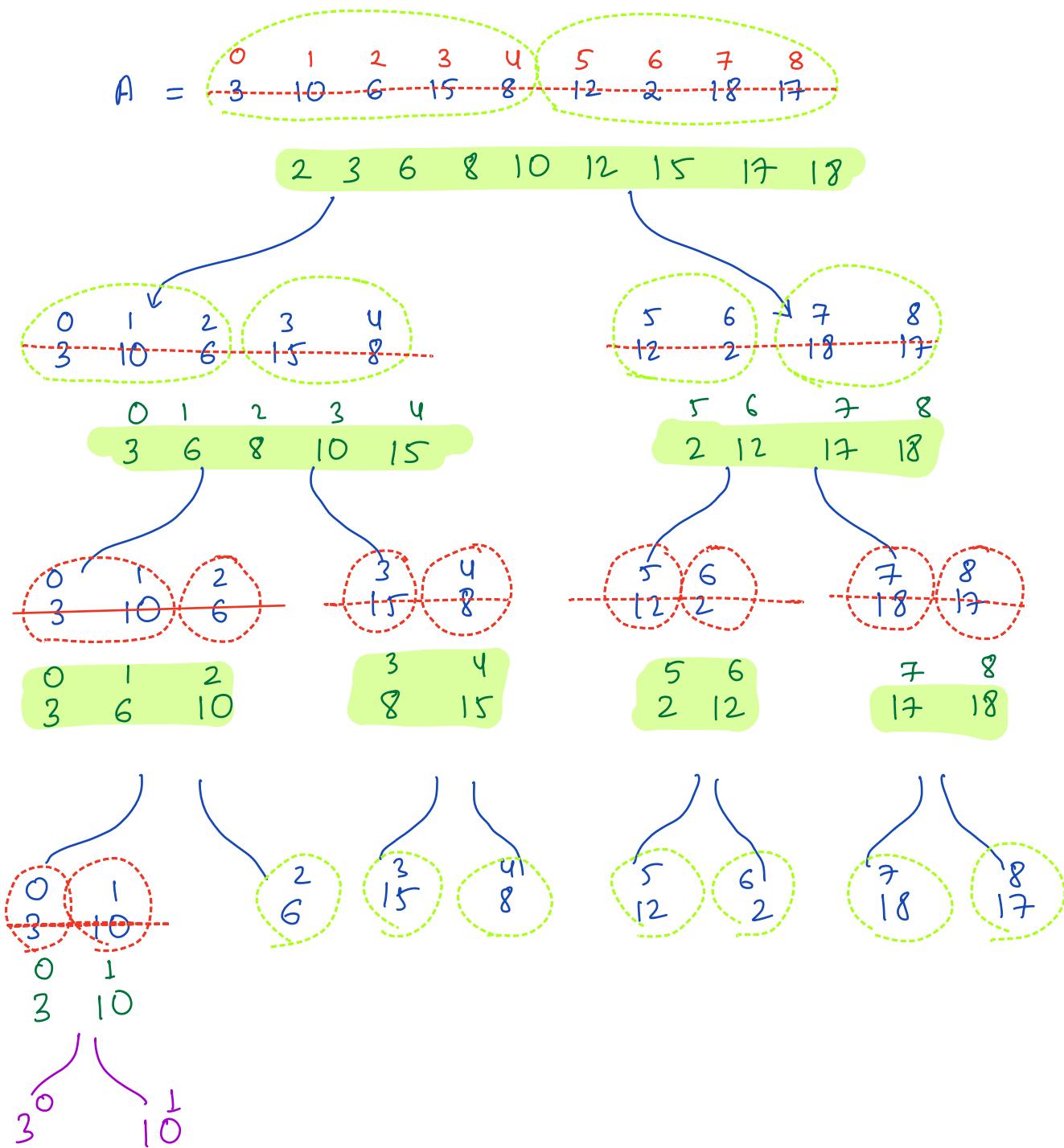
= iterations

$$= 2700$$



## Idea for merge sort

Keep on dividing the array till you have individual elements, then you merge.



## Pseudo code

```
void mergeSort( A[], s , e ) {  
    if( s == e ) return  
  
    m = (s+e)/2  
    mergeSort( A, s , m )  
    mergeSort( A, m+1 , e )  
    merge( A, s , m , e )  
}
```

$$T(n) = 2 * T(n/2) + n \quad \text{HW}$$

TC :  $O(n \log n)$

SC :  $O(N)$

## Inversion Count

Given an integer array, count the number of inversion pairs in the array.

Inversion pair  $\rightarrow (i, j)$  where  $(i < j \text{ & } A[i] > A[j])$

$$A = \begin{matrix} 4 & 5 & 1 & 2 & 6 & 3 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ \downarrow & \downarrow & & & \downarrow & \\ 3 & 3 & & & 1 & \end{matrix} = \textcircled{7}$$

$$A = \begin{matrix} 8 & 3 & 4 \\ 0 & 1 & 2 \end{matrix} \rightarrow 2$$

$(8, 3)$   
 $(8, 4)$

$$A = \begin{matrix} 10 & 3 & 8 & 15 & 6 \\ 0 & 1 & 2 & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 3 & 0 & 1 & 6 & 0 \end{matrix} \rightarrow 5$$

---

Brute force :  $(i, j)$  where  $(i < j \text{ & } A[i] > A[j])$

Two loops  $i, j$  and count the times above condition is true.

$$A = \begin{matrix} 6 & 2 & 4 & 5 & 2 & 3 & 1 & 2 & 1 \\ (10 & 3 & 8 & 15 & 6 & 12 & 2 & 18 & 7 & 1) \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} = (26),$$

$$\begin{matrix} 10 & 3 & 8 & 15 & 6 \\ 0 & 1 & 2 & 3 & 4 \end{matrix}$$

$P_1$

$$\begin{matrix} 12 & 2 & 18 & 7 & 1 \\ 5 & 6 & 7 & 8 & 9 \end{matrix}$$

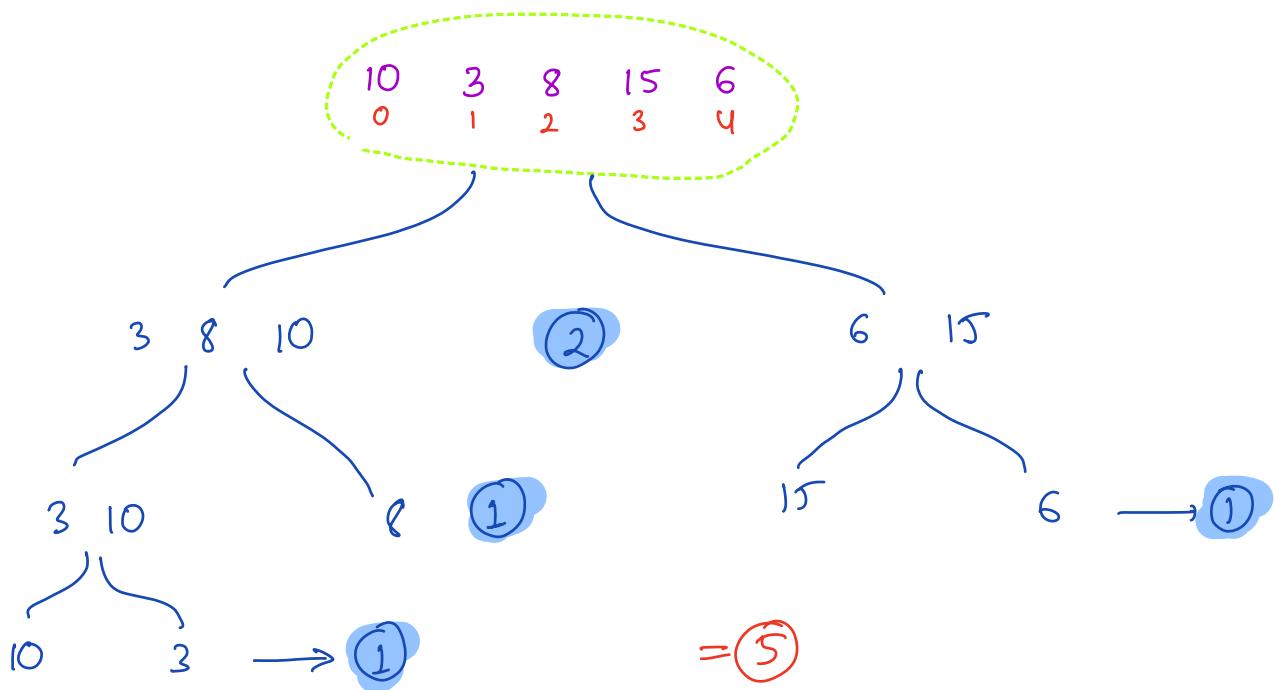
$P_2$

$$\begin{matrix} 3 & 6 & 8 & 10 & 15 \\ 0 & 1 & 2 & 3 & 4 \end{matrix}$$

$$\begin{matrix} 1 & 2 & 7 & 12 & 18 \\ 5 & 6 & 7 & 8 & 9 \end{matrix}$$

$$\begin{matrix} 5 \\ \downarrow \\ 5 \end{matrix} \quad \begin{matrix} 6 \\ \downarrow \\ 5 \end{matrix} \quad \begin{matrix} 7 \\ \downarrow \\ 3 \end{matrix} \quad \begin{matrix} 8 \\ \downarrow \\ 1 \end{matrix} = 14$$

$$\{(3,1), (6,1), (8,1), (10,1), (15,1)\}$$



count = 0

```
int[] merge ( A[N] , s , m , e ) {
```

```
    C[e-s+1] // init
```

```
    p1 = s
```

```
    p2 = m+1
```

```
    k = 0
```

```
    while ( p1 ≤ m && p2 ≤ e ) {
```

```
        if ( A[p1] ≤ A[p2] ) {
```

```
            C[k] = A[p1]
```

```
            k++
```

```
            p1++
```

```
}
```

```
        else {
```

```
            C[k] = A[p2]
```

```
            k++
```

```
            p2++
```

```
}
```

```
        count += m-p1+1
```

s      m  
↓  
p1

m+1    e

```
}
```

while ( p1 ≤ m ) {

```

    C[k] = A[p1]
    k++
    p1++
}

while (p1 <= e) {
    C[k] = A[p2]
    k++
    p2++
}

for (i = 0 ; i < C.length ; i++) {
    A[i+s] = C[i]
}

void mergeSort(A[], s, e) {
    if(s == e) return

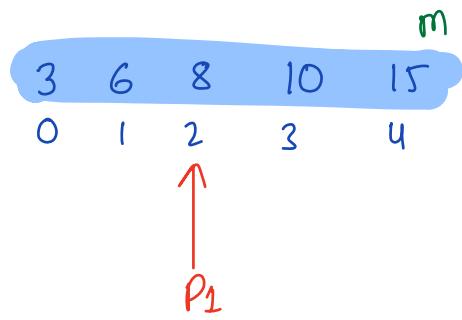
    m = (s+e)/2
    mergesort(A, s, m)
    mergesort(A, m+1, e)
    merge(A, s, m, e)
}

```

TC :  $O(n \log N)$

SC :  $O(N)$ .

Doubt



# elements

$$m - p_1 + 1$$

$$\begin{matrix} & & 4 \\ 2 & \curvearrowright & 4 & \curvearrowright & 6 \\ & 2 & & 2 \end{matrix} = 8$$

$$\begin{array}{r} 0 \ 1 \ 0 \\ \underline{1 \ 0 \ 0} \end{array}$$

$$\begin{matrix} 2 & 4 \\ 2 & 6 \\ 4 & 6 \end{matrix} \quad \left\{ \begin{matrix} 2 & 4 \\ 2 & 6 \\ 4 & 6 \end{matrix} \right\}$$

## Magical Number 5

Flag Question

### Problem Description

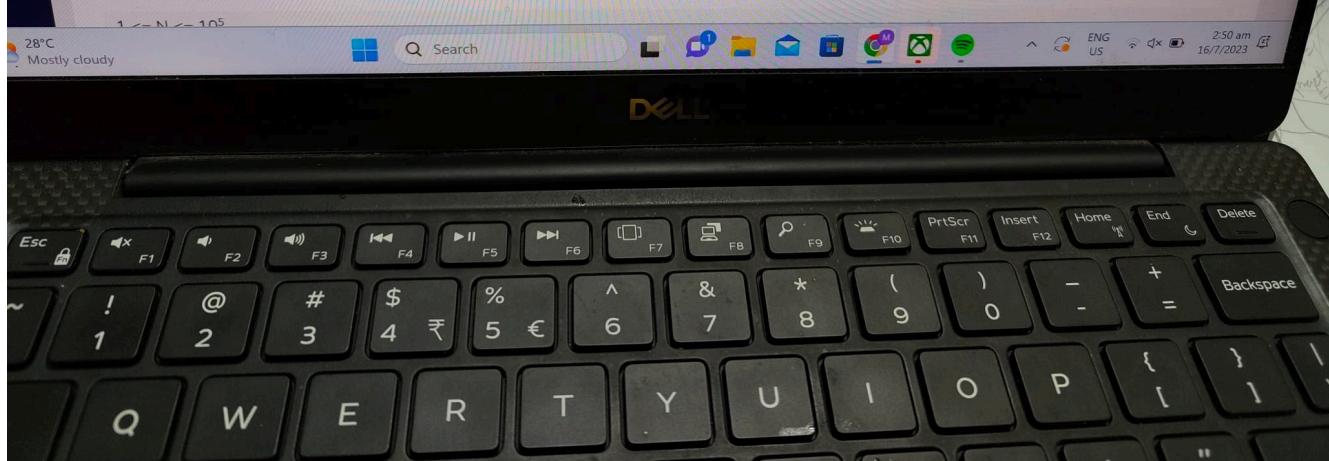
In a magical world, there is a group of wizards who can cast spells by reciting binary numbers. Each time a wizard recites a binary number, a powerful spell is cast. The more powerful the spell, the larger the binary number recited by the wizard.

The wizards have a tradition of casting spells only on numbers that are divisible by 5. To make their spells more powerful, the wizards want to know the power of the spells they can cast by reciting binary numbers from the most-significant-bit to the least-significant-bit.

Given a binary array `nums` (0-indexed), we define  $xi$  as the number whose binary representation is the subarray  $A[0..i]$  (from most-significant-bit to least-significant-bit). For example, if  $A = [1, 0, 1]$ , then  $x_0 = 1$ ,  $x_1 = 2$ , and  $x_2 = 5$ .

Write a function that returns an array of booleans where `answer[i]` is true if  $xi$  is divisible by 5.

### Problem Constraints



$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad 1 \quad 2 \quad 5$$

$$\begin{aligned} x[0] &= A[0:0] = 1 &= 1 \% 5 &= \text{No} \\ x[1] &= A[0:1] = 10 &= 2 \% 5 &= \text{No} \\ x[2] &= A[0:2] = 101 &= 5 \% 5 &= \text{No} \\ x[i] &= A[0:i] \end{aligned}$$

num = 0

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$i = 0 \quad 2 * \text{num} + A[i] = 1 \% 5$$

$$i = 1 \quad = 2 \% 5$$

$$= 5 \% 5$$

↑  
i