18) Count Pairs "ag" }

char []

Given a string, calculate no of pairs i, j such that → index values

i < j && S[i] = 'a' && S[j] = 'g' All characters are lowercase

{a, b .. z}

Ex1:

0	1	2	3	4	5	6	7
b	a	a	g	d	c	a	g

Pairs: <1,3> <1,7> <2,3> <2,7>, <6,7> → 5

Ex2:

0	1	2	3	4	5	6	7
b	c	a	g	g	a	a	g

Pairs: <2,3> <2,4> <2,7> <5,7>, <6,7> → 5

Ex3:

0	1	2	3	4	5	6
a	c	g	d	g	a	g

Pairs: <0,2> <0,4> <0,6> <5,6> → 4

Sol: Check all pairs & inc count

TC: O(N²)

SC: O(1)

```
int c = 0
i = 0; i < N; i++) {
    j = i+1; j < N; j++) {
        if(S[i] == 'a' && S[j] == 'g') {
            c++
        }
    }
}
return c;
```

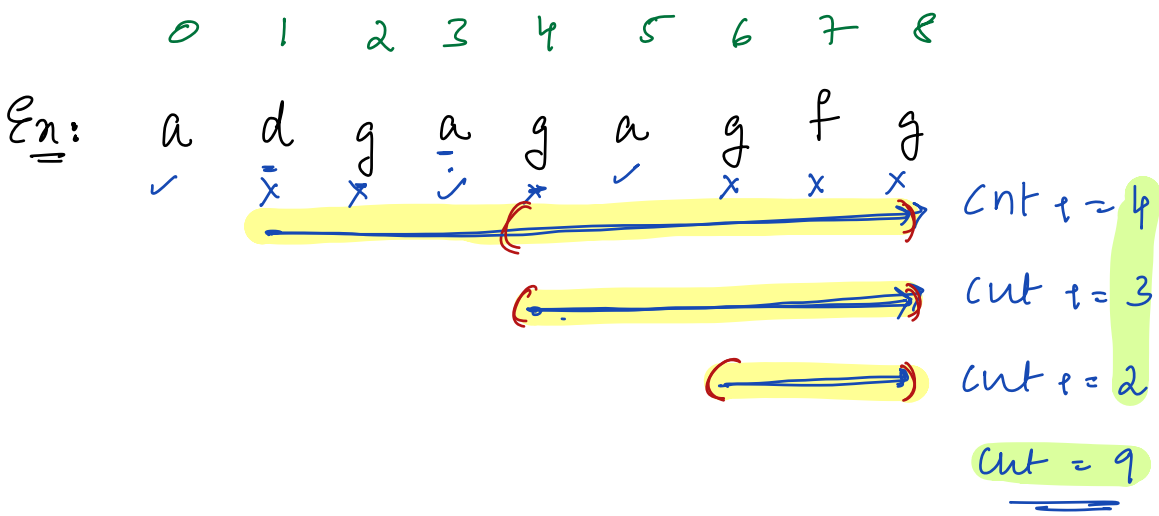
```
int c=0
for(i=0; i<N; i++){
    if(s[i]=='a'){
        for(j=i+1; j<N; j++){
            if(s[j]=='g'){
                c++
            }
        }
    }
}
```

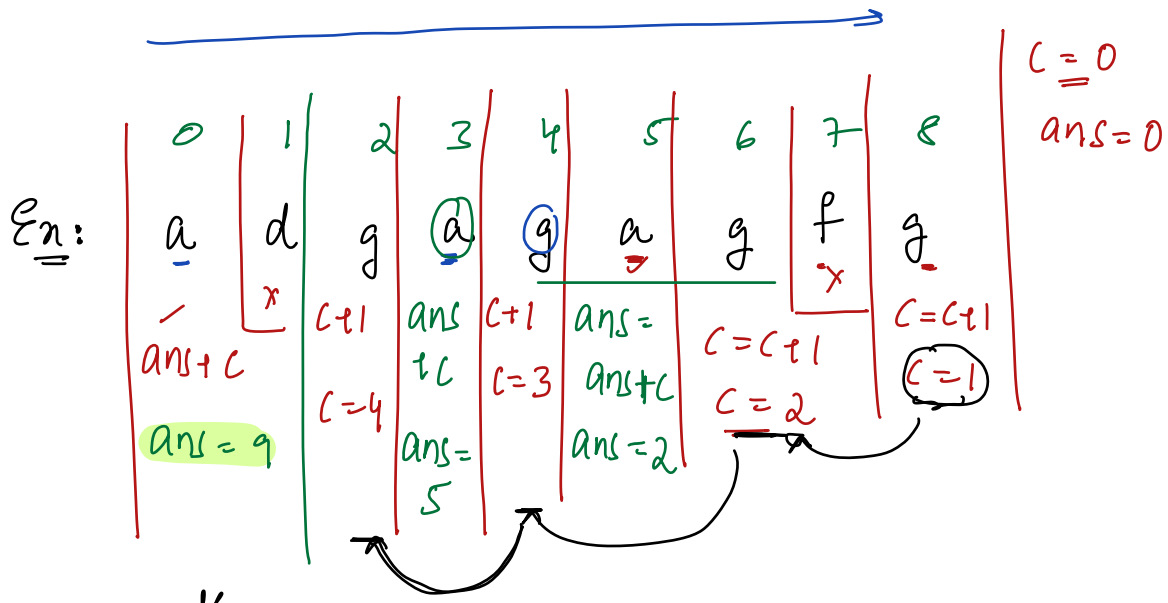
if s[i] == 'a'

We are getting no. of g's on right side.

return c;

TC: O(N²) SC: O(1)





Pseudo code

Carry count of g's

ans = 0, c = 0

```
for i = N-1; i >= 0; i--) {
    if (s[i] == 'g') {
        c+1
    }
    else if (s[i] == 'a') {
        ans = ans + c
    }
}
return ans
```

TC: O(N) SC: O(1)

TODO

Carry forward no. of a's

if s[i] == 'g', no. of a's on left side

28) Leaders in a Array

Given an ar[N], you have to find all leaders in ar[]

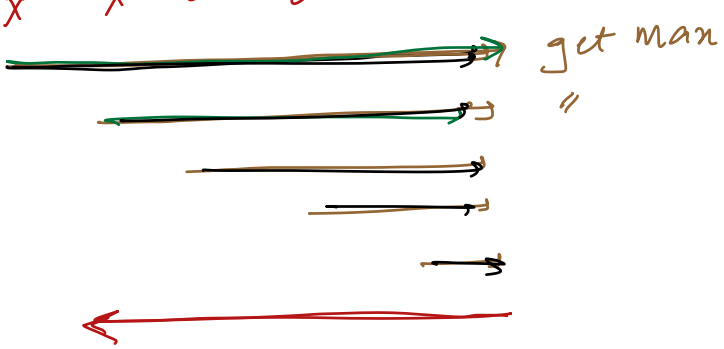
An ele is leader, if it is strictly greater (>) than all ele on it's right side.

Note: ar[N-1] is always considered as leader

	0	1	2	3	4	5	6	7	
Ex1:	15	-1	7	2	5	4	2	3	leaders: 5
	✓	x	✓	x	✓	✓	x	✓	

	0	1	2	3	4	5	
Ex2:	10	7	9	3	2	4	leaders: 3
	✓	x	✓	x	x	✓	

	0	1	2	3	4	5	6	
Ex3:	8	-2	4	7	6	5	1	leaders: 5
	✓	x	x	✓	✓	✓	✓	



Pseudocode

```
c = 1;
max = ar[N-1];
for(i = N-2; i >= 0; i--){
    if(ar[i] > max){
        c++ // ar[i] is leader
        max = ar[i]
    }
}
return c;
```

TC: O(N)

SC: O(1)

Doubt

ar[1] = {8} →

output: 1

Doubt

ar[3] = { 10, 8, 8 } leaders = 2

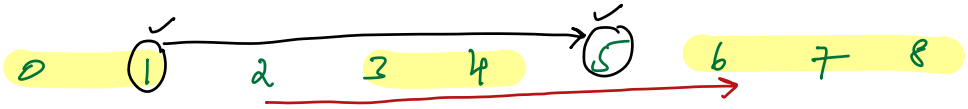
✓ ✗ ✓

Subarray Basics : Continous part of an array is called Subarray

→ Single element is subarray ✓

→ Entire array is subarray ✓

→ Empty is not subarray ✓



ar[ ] : 3 2 8 -3 0 9 10 8 6

Eg: [2 - 5] ✓

[2 - 4] [6 - 8] ✗ ? [Index 5 is skipped]

[0 1 ↑3 4 ↑6 7 8] ✗ ? [2, 5 are skipped]

[2 - 2] ✓

// Note: If start of subarray fixed = s

If end of subarray fixed = e

Subarray = [s e], len = e - s + 1

sort( ) → To Sort N Elements → TC: O(N log N)

max(a, b) {→ it return max of a, b : O(1)

min(a, b) {→ it return min of a, b : O(1)

3(8) **Closest Min Max**

Given an array find the length of (Smallest) Subarray which contains Both Min & Max of array?

	0	1	2	3	4	5	6	7	8	9
Ex1:	1	2	3	1	3	4	6	4	6	3

[a b] → b - a + 1

Min: 1
Max: 6

[3 - 8] : len → 6
[3 - 6] : len → 4

Ex2:

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	1

Min: 1
Max: 6

[2 - 5] : len → 4
[8 - 10] : len → 3

Ex3:

0	1	2	3	4	5	6	7	8	9	10	11
1	6	4	2	7	7	5	1	3	1	1	5

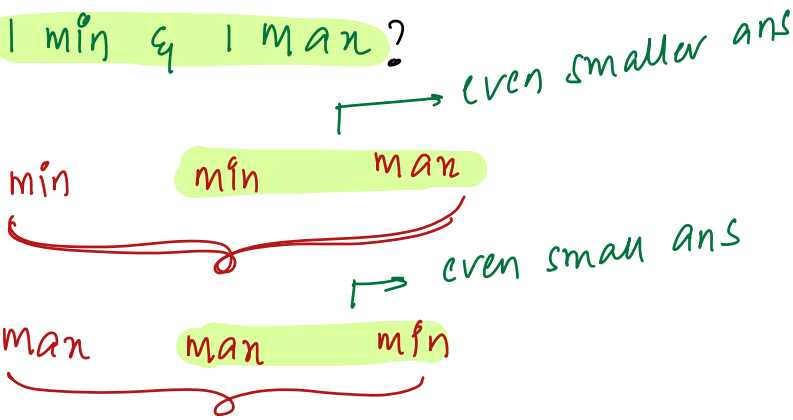
Min: 1
Max: 7

[0 - 4] : len → 5
[5 - 7] : len → 3

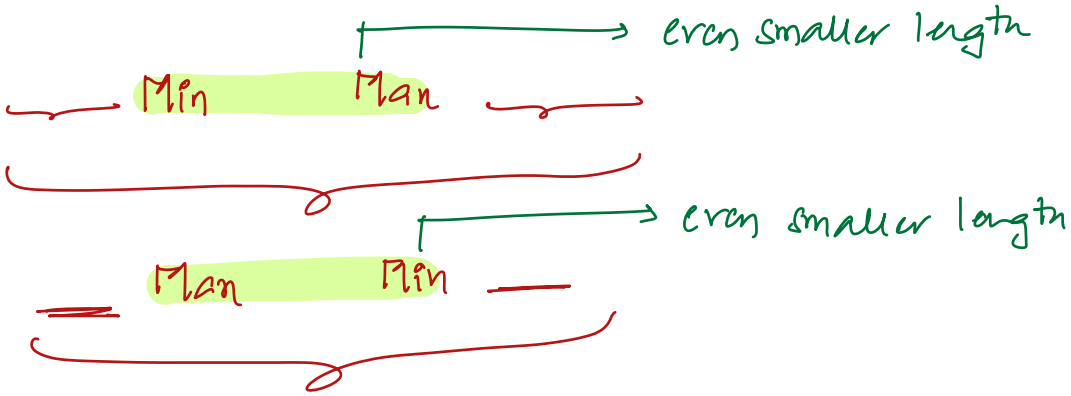
observations

→ subarray length is min.

1) In your ans subarray it should only contain 1 min & 1 max?



2) Min & Max should present in extremes of subarray?



Cases.

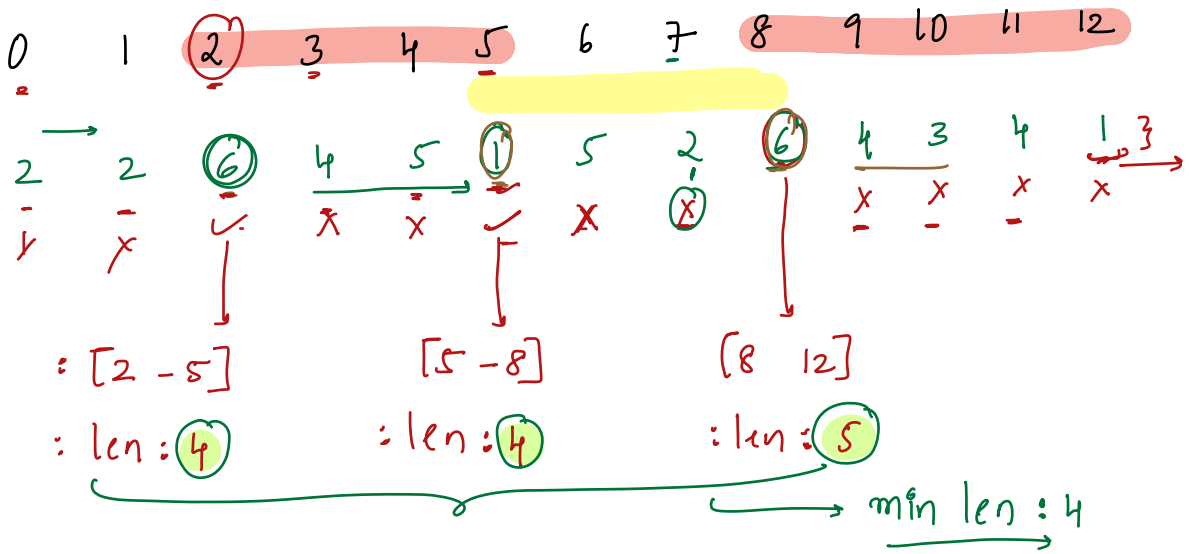
Final Subarray?

1) Min arr[] Max

2) Max arr[] Min

Min : 1

Max : 6


0 1 2 3 4 5 6 7 8 9 10 11 12
2 2 6 4 5 1 5 2 6 4 3 4 1 3
: [2 - 5]
[5 - 8]
[8 12]
: len : 4
: len : 4
: len : 5
min len : 4

Pseudo Code

ans = N → iterate & get min_val, iterate & get max_val

```
ans = N
iterate & get min_val
iterate & get max_val
i = 0; i < N; i++) {
    if( ar[i] == min_val) {
        j = i; j < N; j++) {          iterate & get 1st max on right side
            if( ar[j] == max_val){
                ans = min (ans, j-i+1)   => { [i - j] -> {j-i+1}   (length)
                break;
            }
        }
    }
    else if( ar[i] == max_val) {      iterate & get 1st min on right side
        j = i; j < N; j++) {
            if (ar[j] == min_val) {
                ans = min (ans, j-i+1)
                break
            }
        }
    }
}
return ans;
```

TC: $O(N^2)$ SC: $O(1)$

ans → current; j-i+1 (0, 1)

Note: j = i; j < N; j++ it will handle edge case where both min & max are same.

Doubts:

```
min_val = ar[0]

for(i = 0; i < N; i++){
    min_val = min(min_val, ar[i])
}
```

min(current min val, ar[i])

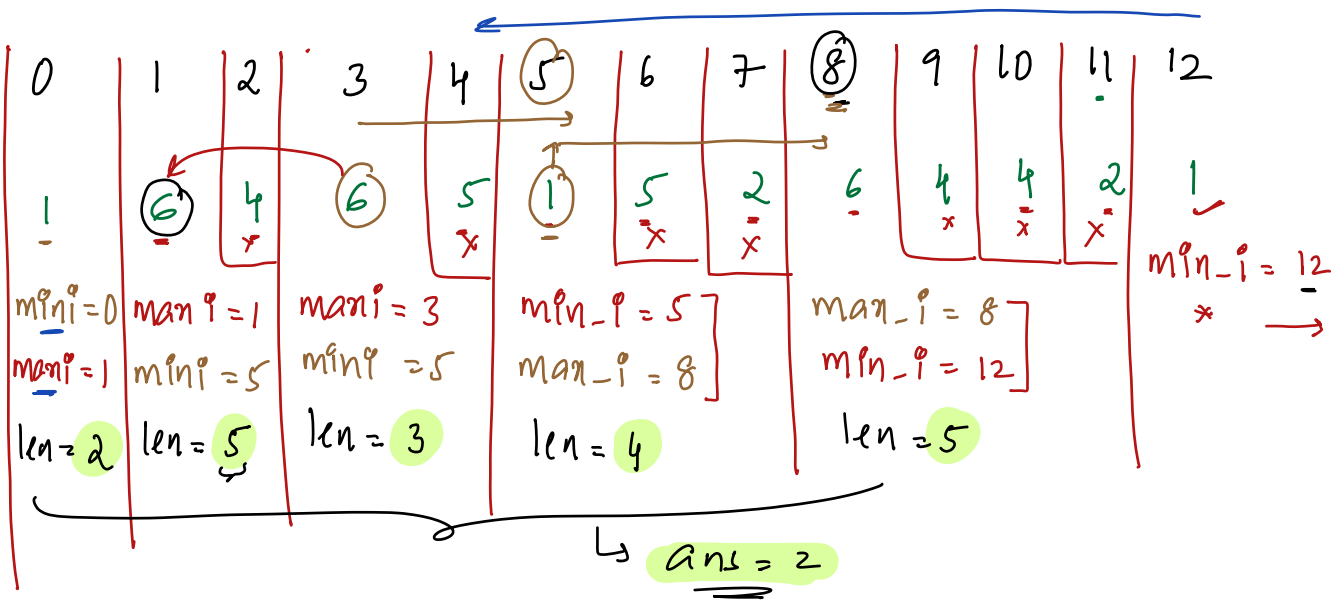
max_value = 6 ✓

min_value = 1 ✓

ans = N

min_i = -1

max_i = -1



Ex: 0 1 2 3

ans = N, mini = -1, maxi = -1

8 8 8 8

mini=0 mini=1 mini=2 mini=3

maxi=-1 maxi=-1 max=-1 maxi=-1

Pseudocode

1) Iterate & find max_val

2) iterate & find min_val

if (max_val == min_val) return 1

ans = N, mini = -1, maxi = -1

```
for (i = N-1; i >= 0; i--) {
    if (ar[i] == min_value) {          // 1st min on right side
        mini = i;
        if (maxi != -1) {              // there is max on right
            // subarray [mini, maxi]
            ans = min(ans, maxi - mini + 1)
        }
    }
    else if (ar[i] == max_value) {     // 1st max on right side
        maxi = i;
        if (mini != -1) {              // there is min on right
            // subarray [maxi mini]
            ans = min(ans, mini - maxi + 1)
        }
    }
}
return ans;
```

TC: O(N)

SC: O(1)

Why TLE?

TLE - Time Limit Exceed

→ Hiring Challenges → observations → logic → ✓ → Code ← TLE ← TLE OS

Google: 1hr → Q1: → Q2:

Run on their Servers

→ Online Editors → execution time 1sec → 1 GHz

$1\,GHz = 10^9$ instructions/sec

obs1: Our code can atmax have 10^9 instructions

```
for( i = 1 ; i <= 100 ; i++ ){
    print(i)
}
```

① i = 1; ① i <= 100; ② i > ②, i = i + 1; ④ print(i)

approx 4 instruction per iteration

Total 400 instructions

$\approx 10^9$ instru

$\approx 10^7 \times 10^2$ instr

$\approx$

In our code

1 iteration = 10 instructions $\approx = 10^8$ iterations

1 iteration = 100 instructions $\approx = 10^7$ iterations

Our code iterations → $[10^7 - 10^8]$ iterations

Constraints

$1 \le N \le 10^5$ ← array size

$1 \le ar[i] \le 10^9$ ← array value

Question: No: of leaders

Sol1: $O(N^2)$ →

$N = 10^5 \approx 10^{10}$ iterations (TLE)

Sol2: $O(N)$

$N = 10^5 \approx 10^5$ iterations (✓)

Constraints

$1 \le N <= 10^3$ } → Approach → TC: $O(N^2)$ ✓

Min Picks → Monday ✓

Constraints

$1 \le T <= 100$

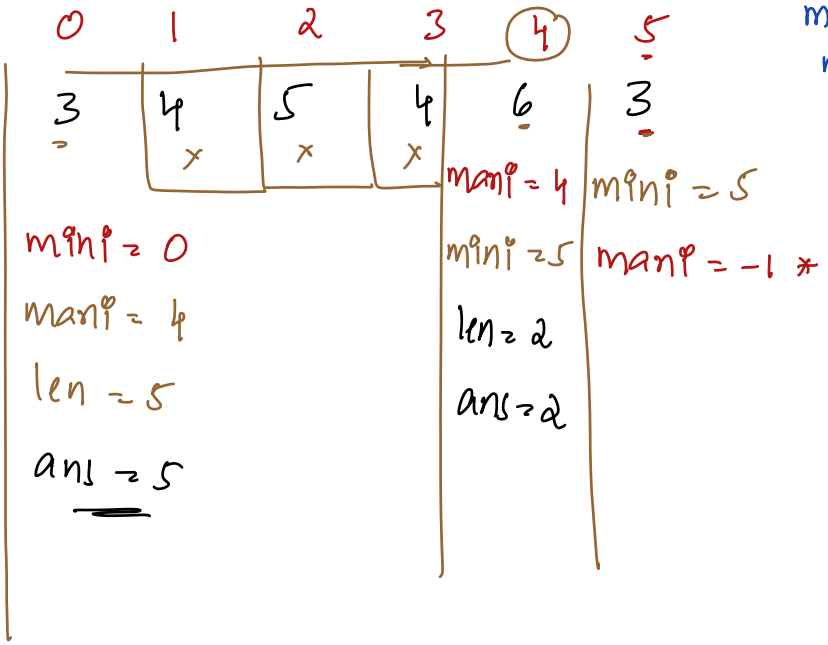
$1 <= N <= 10^3$

} { T * TC of your code } $<= 10^8$ iterations

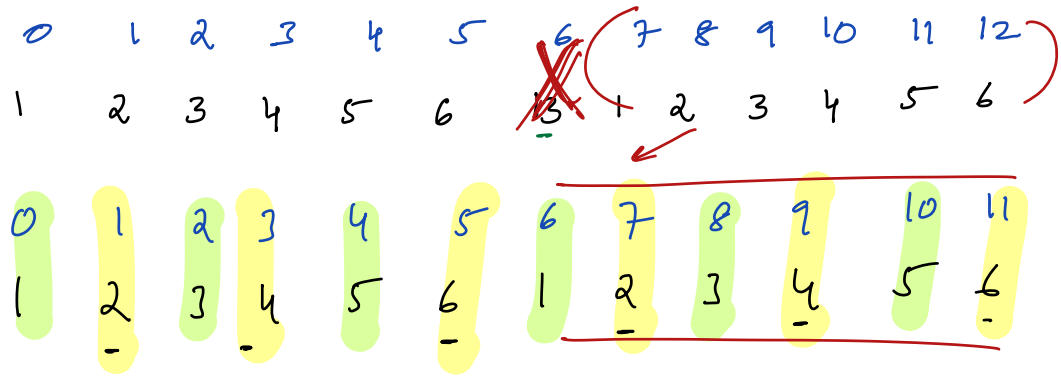
Doubts

ans = N
mini = -1
mani = -1

ar[ ] →



ar[ ] :



even sum = 21 } →
odd sum = 24 }

Prefix Sum → pf[i] → Sum of all elements [0 i]

Suffix Sum → sf[i] → Sum of all elements [i N-1]

	0	1	2
A :	67,	104,	99

Pfeven[] :

Pfodd[] :

Pf[] : 67 ~~171~~ 270

[2 2] → Pf[2] - Pf[1]

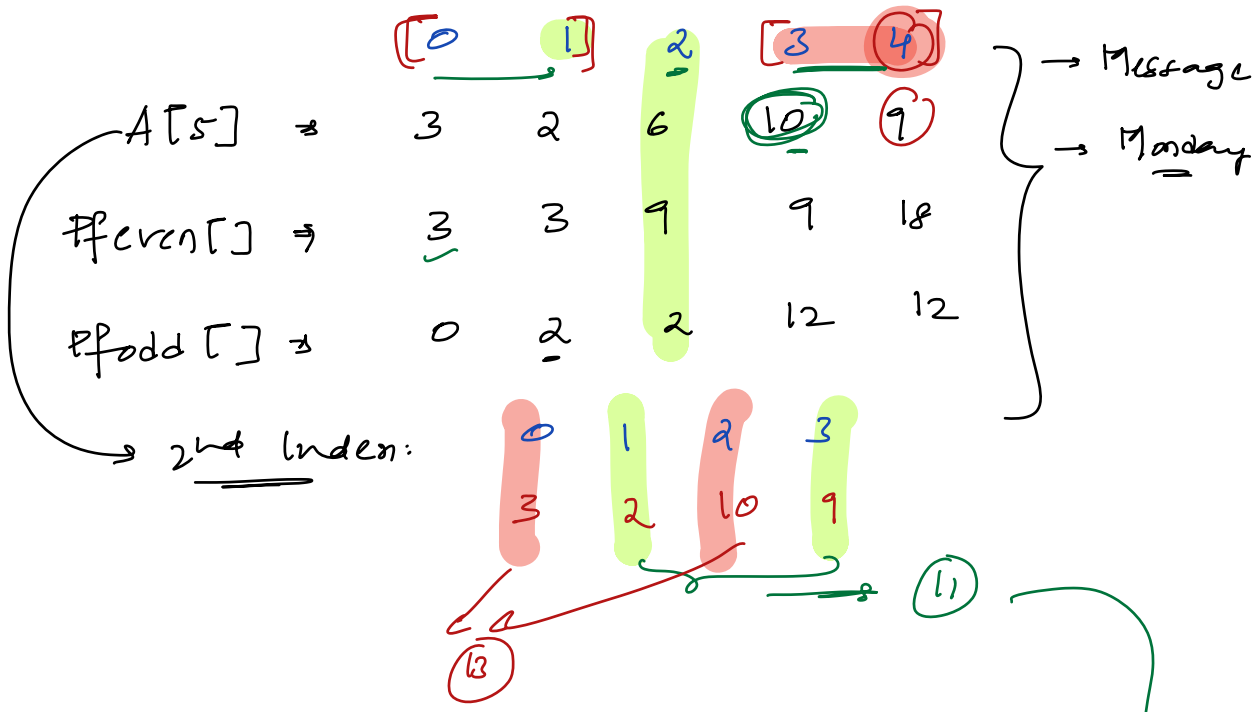
→ 270 - 171 = 99

	0	1	2	3	4	5
A[] :	3	2	6	8	10	-1
Pf[] :	3	5	11	19	29	28

[1 1] + [3 4] → Pf[4] - Pf[2] = 29 - 11 = 18

→ Pf[1] - Pf[0] = 2 → 20

	0	1	2
A :	67,	104,	99
Pfeven[] :	67,	67,	166



Delete 2nd Index

10 → 13

Total even sum = sumeven[0 1] + sumodd[3 4]

Total odd Sum = sumodd[0 1] + sumeven[3 4]

2

9 → 11

508) N Bulbs → {Interview}

N light bulbs are connected by wire

Initial state of each bulb is given → 0 on
→ 1 off

Note: Each bulb has a switch associated with it.
but if we press i^{th} switch all bulbs index >= i
flip their status

Goal: Minimize no: of switch press, to on all bulbs

Eg1: [0 1 0 0 1]