TLE and constraints
Carry Forward Question
Intro to subarrays
Iterating subarray
Question

Contest after Intermidiate module
↓
1.5 hrs → 3 Ques

TLE (Time Limit Exceeded Error)

Online Platforms → 1 GHz

↓

$10^9$ instructions /sec

Algo → iterations

```
bool countFactors (N) {
    int cnt = 0          → 1
    for (i=1 ; i ≤ N ; i++) {
        if (N % i == 0) {
            c = c+1
        }
    }
    return c
}
```

+1 +1 +1 +1 +1 +1 +1 +1

Ass 1 : 10 instructions → 1 iteration

1 instr → 1/10 iteration

$10^9$ instr → $\frac{10^9}{10}$ iteration

$= 10^8$ iter

$\boxed{1 \text{ sec}}$ $10^9$ instr → $10^8$ iter

Ass 2 :

100 instr → 1 iteration

1 instr → $\frac{1}{100}$ itr

$10^9$ instr → $\frac{10^9}{10^2} = 10^7$ itr

1 sec → $10^9$ instr → $10^7$ iter

Conclusion → $10^7 - 10^8$ iter $\boxed{1 \text{ sec}}$

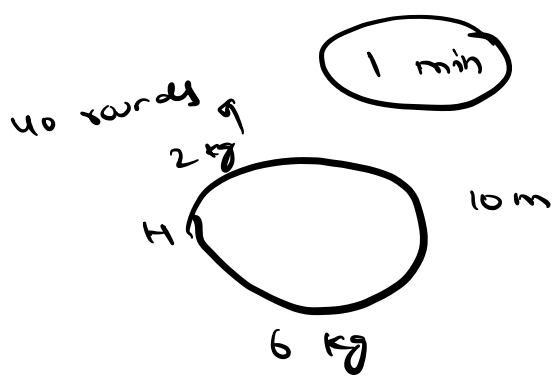
1 iter

↙ ↘

10 instr — $10^8$ iter

100 instr — $10^7$ itr

Better processor



$10^7 - 10^8$ iter / sec

1. Read problem
2. Identify constraints
3. Examples → observe sol
4. Idea → TC
5.

$1 < N < 10^5$

$O(N \log_2 N)$

$10^5 \log_2 10^5$ ✓

$10^5 \times 15$ ✓

$O(N)$ ↓ $10^5$ ✓

$O(N^2)$ ↓ $10^{10}$ TLE

$1 < N < 10^7$

$O(N^2) \rightarrow 10^{14}$ ✗

$O(N) \rightarrow 10^7$ ✓

$1 < N < 100$

$O(N)$ ✓

$N^2$ ✓

$O(N^3)$ ✓

$N = 10^2$

$N^3 = 10^6$

$N^4 \rightarrow 10^8$

Doubtful

1. Given a string s of lowercase characters, return count of pairs (i, j) such that $i < j$ and $s[i] = 'a'$, $s[j] = 'g'$.

L R

s = abegag  ans = 3

i j
L R
a g

s = acgdgag  ans = 4

$1 < N < 10^5$

a a c g d g a g

s = bcaggaag  ans = 5

Brute Force :

For every 'a', look for 'g' on right

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

TC : $O(N^2)$

SC : $O(1)$

S = a a a a a a a a

a → g
a ← g

## Optimized Approach

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| | a | c | b | a | g | k | a | g | g |
| count A = 0 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| ans = 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 5 | 8 |

① When 'g' comes, look for 'a' on left

② Store the count of 'a' and whenever 'g' is encountered, ans += count of a

a [a a g]
a 3

```
ans = 0        cnta = 0
for (i=0 ; i<n ; i++)
    if (S[i] == 'a')
        cnta ++
    else if(S[i] == 'g')
        ans += cnta
return ans
```

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

$10^5$ iter ✓

S = g g g g
ans = 0  0 0 0 0
cnta = 0  0

10:22

# Introduction to subarrays → part of array

A subarray is a contiguous part of an array. It is formed by selecting a range of elements from the array. A subarray can have one or more elements and must be a contiguous part of the original array.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2 | 3 | -1 | 6 | 9 | 8 | 12 |

[4, 1, 2] ✓

[2, 3, -1, 6] ✓

[9] ✓

[4, 1, 2, 3, -1, 6, 9, 8, 12] ✓

[3, 6, 8] ✗

[4, 12] ✗

[3, 2, 1, 4] ✗

subarray → left to right

A = <2, 4, 1, 6, -3, 7, 8, 4>

which of following is a valid subarray?

<1, 6, 8> ✗

<1, 4> ✗

<6, 1, 4, 2> ✗

<7, 8, 4> ✓

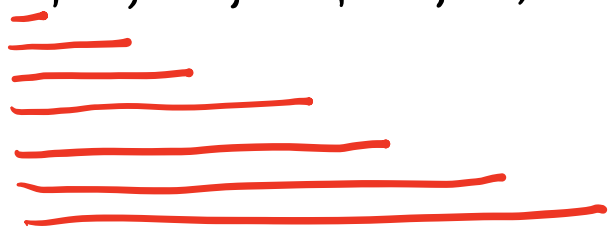
# Representation of subarray

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2 | 3 | -1 | 6 | 9 | 8 | 12 |

1. Start idx and end index (2,5)
2. Start idx and len of subarray (2,4)

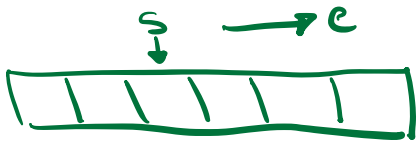
How many subarrays of following array start from index 0?

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 4, | 2, | 10, | 3, | 12, | -2, | 15 |



ans = 7

s e
0,0
0,1
0,2
0,3
0,4
0,5
0,6

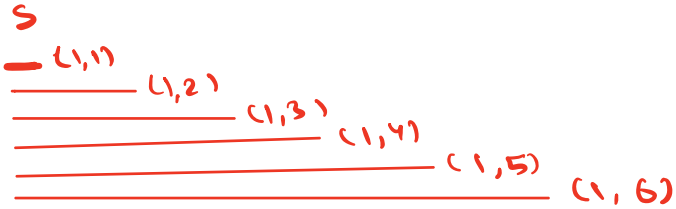



$s \le e$

How many subarrays of following array start from index 1?

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 4, | 2, | 10, | 3, | 12, | -2, | 15 |

s




s e
1 1 to 6

ans = 6

Formula to count total no. of subarrays

$[a_0 \quad a_1 \quad a_2 \quad \ldots\ldots \quad a_{n-2} \quad a_{n-1}]$ len = n

Subarrays start from $0^{th}$ idx = $N$

$s = 0$ $\boxed{s \quad e}$ +

$e = 0, 1, 2 \ldots\ldots N-1$

Subarrays start from $1^{st}$ idx = $N-1$

$s = 1$

$e = 1, 2, 3, \ldots\ldots N-1$

+

Subarrays start from $2^{nd}$ idx = $N-2$

$s = 2$ +

$e = 2, 3, 4 \ldots N-1$

⋮ ⋮

Subarrays start from $N-2^{nd}$ idx = $2$

$s = N-2$

$e = N-2, N-1$ +

Subarrays start from $(N-1)^{th}$ idx = $1$

$s = N-1$ $e = N-1$

Total subarrays $= \frac{N(N+1)}{2}$

$$\begin{array}{ccc} 0 & 1 & 2 \\ [10, & 20, & 30] \end{array}$$

$N = 3$

subarray cnt $= \frac{3 \times 4}{2} = 6$

| s | e | |
|---|---|---|
| 0 | 0 | 10 |
| 0 | 1 | 10, 20 |
| 0 | 2 | 10, 20, 30 |
| 1 | 1 | 20 |
| 1 | 2 | 20, 30 |
| 2 | 2 | 30 |

$s \le e$

Q. Given an array of integers, 2 indexes $\underset{s}{\downarrow}$ $\underset{e}{\downarrow}$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2 | 3 | -1 | 6 | 9 | 8 | 12 |

$\underset{2}{s}$ $\underset{5}{e}$

```
// arr [ ], int s, int e
for (i = s ; i ≤ e ; i++) {
    print (arr [i])
}
```

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

$\underset{0}{s}$ $\underset{N-1}{e}$

Size of subarr [s, e] $= e - s + 1$

Print all subarrays

| 0 | 1 | 2 |
|---|---|---|
| 10 | 20 | 30 |

[10, 20, 30]  N=3

| s | e |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 0 | 2 |
| 1 | 1 |
| 1 | 2 |
| 2 | 2 |

output

10
10, 20
10, 20, 30
20
20, 30
30

| s | e | |
|---|---|---|
| 0 | 0 | 0, 0 |
| 0 | 1 | 0, 1 |
| 0 | 2 | 0, 2 |
| 1 | 1 | 1, 1 |
| 1 | 2 | 1, 2 |
| 2 | 2 | 2, 2 |

$N^2$ subarrays

```
for ( s=0 ; s < n ; s++) {
    for (e = s ; e < n ; e++) {
        // s e
        for (i=s ; i <= e ; i++) {
            print (arr[i])
        }
        print ("\n")
    }
}
```

| s | e |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 0 | 2 |

1 subarr → $O(N)$

$N^2$ subarr → $N^2 * O(N)$

TC → $O(N^3)$

SC : $O(1)$

Given an array of N integers, return length of smallest subarray which contains both minimum & maximum of the array.

2 2 6 4 5 1 5 2 6 4 1 ans = 3

Max = 6
Min = 1

1 2 3 1 3 4 6 4 6 3

Max = 6
Min = 1

ans = 4

Brute force : Check all subarrays

1. First find min and max → O(N)
2. Check all subarrays TC : $O(N^3)$

SC : O(1)

Optimised solution

Max = 6
Min = 1

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 3 | 4 | 6 | 4 | 6 | 3 |

Obs 1 : In the ans subarray, there will be 1 min and 1 max present

Max . . . Max . . . . Min

Max ... Min ..... Max

Edge case : 2 2 2 2 2 2

Min = 2
Max = 2

Obs 2 : Max and Min will be corner elements of ans subarray

... Max .... Min ..

... Min .... Max ...

Subarray → Max ... Min

→ Min ... Max

So, basically we are looking for subarray **which starts with maximum value and ends with closest minimum value or which starts with minimum value and ends with closest maximum value.**

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

6 ← 1

2 5

$5 - 2 + 1 = 4$

Max = 6

Min = 1

last_min_idx = ~~-1~~ ~~5~~ 10

last_max_idx = ~~-1~~ ~~7~~ 8

ans = ~~0~~ ~~4~~ 3

INT_MAX / Integer.MAXVAL

1. Calculate min and max of arr
2. lastMinIdx = -1
   lastMaxIdx = -1
   ans = INT_MAX / ∞

```
for (i=0 ; i<n ; i++) {
    if (A[i] == min) {
        if (lastMaxIdx != -1) {
            len = i - lastMaxIdx + 1
            ans = min (ans, len)
        }
        lastMinIdx = i
    }
    else if (A[i] == max) {
        if (lastMinIdx != -1) {
            len = i - lastMinIdx + 1
            ans = min (ans, len)
        }
        lastMaxIdx = i
    }
}
```

TC: O(N)

SC: O(1)

```
for ( s=0 ; s < n ; s++) {
    for (e = s ; e < n ; e++) {
            flag1 = false, flag2 = false
        for (i=s ; i ≤ e ; i++) {
            if (a[i] == max)
                flag1 = true
            if (a[i] == min)
                flag2 = true
        }
        if (flag1 && flag2) {
            ans = min (ans, e-s+1)
        }
    }
}

for (i=0 ; i<n ; i++) {
        if (s[i] == 'a') {
            for(j= i+1 ; j<n ; j++) {
                if (s[j] == 'g')
                    ans++
            }
        }
}
return ans
```

[a b] = b-a+1

[2 N-1]

N-x-2+x

| i | j | Iterations |
|---|---|---|
| 0 | [1 N-1] | N-1 + |
| 1 | [2 N-1] | N-2 + |
| 2 | [3 N-1] | N-3 + |
| ⋮ | | ⋮ |
| N-2 | [N-1 N-1] | 1 + |
| N-1 | [N | 0 |

$$\text{sum} = \frac{(N-1) \times N}{2}$$

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

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