N. Satya Sai Siva Rama krishna 2019

Teaching exp: 3+

Work Experience in Scaler: 2+

Sessim will start sharp 7:05 AM

Todays Content:

a) Factors Optimization

b) Rotate arr[] from right to left

c) logarithmetic basics

d) Total DSA Content

Q1) Given N, return no: of factrs of N?

↳ factor? Is 4 a factr of 24 = Yes? 24%4 = 0

1 is a factr of N = N%1 == 0

Countfactrs:

N = 10 factrs = {1 2 5 10} : ans = 4

N = 12 factrs = {1 2 3 4 6 12} : ans = 6

N = 6 factrs = {1 2 3 6} : ans = 4

```
int countfactrs(int N){   Ass: 10^8 iteratns code 1 sec
    int c = 0
    for(int i = 1; i <= N; i++){ ⟶ i : {1 ... N} iteratns = N
        // Check if i is a factr of N?
        if(N%i == 0) { c = c+1 }
    }
    return c;
}
```

What is $10^{10}$ sec?

: Sec → Years

: $10^{10}$ sec ≈ 317 years

You → 1st → 2nd → 3rd → 4th → 5th → 6th

| Input N | iteratns | Exectn time |
|---|---|---|
| $10^9$ | $10^9$ | $x = 10$ sec |
| | $10^8$ ite = 1 sec | |
| | $x \times 10^8 = 10^9$ | |
| | $x = \frac{10^9}{10^8} = 10$ | |
| $10^{18}$ | $10^{18}$ | $x = 10^{10}$ sec |
| | $10^8$ = 1 sec | |
| | $10^{18} = x \times 10^8$ | |
| | $x = \frac{10^{18}}{10^8} = \frac{10^{10} \times 10^8}{10^8} = 10^{10}$ | |

## Observations:

1. Say $i * j = N$
   - : Both i & j are factors of N
   - : $j = N/i$
   - : Both i & N/i are factors of N

   Obs: If i is factor of N
   N/i is a factor of N

   N = 24
   : i = 3 factor
   N/i is a factor = $24/3 = 8$

   N = 24
   : i = 6 factor
   N/i is factor = $24/6 = 4$

2. Relational Operators:

   < > <= >= == !=

3. a <= 10 max value of a = 10
   a <= n max value of a = n

## Optimization:

N = 24

| i | <= | N/i |
|---|---|---|
| Part 1: | | |
| i = 1 | <= | 24 |
| i = 2 | <= | 12 |
| i = 3 | <= | 8 |
| i = 4 | <= | 6 |
| 6 | | 4 |
| 8 | | 3 |
| 12 | | 2 |
| 24 | | 1 |

Obs1: Iterate only in Part1
We can get all factors

Part A: min i max i

1 i <= N/i

i = N/i

i * i = N

$i^2 = N$

Part A: 1 $i = \sqrt{N}$

N = 36

| i | <= | N/i | c |
|---|---|---|---|
| Part 1: | | | |
| i = 1 | <= | 36 | +2 |
| i = 2 | <= | 18 | +2 |
| i = 3 | <= | 12 | +2 |
| i = 4 | <= | 9 | +2 |
| i = 6 | <= | 6 | +1 |
| 9 | | 4 | |
| 12 | | 3 | |
| 18 | | 2 | |
| 36 | | 1 | |

## Optimize Code

```
int count_factrs(int N){
    int c = 0
    for(int i=1; i*i<=N; i++){
        Check if i is factr
        if( N%i==0){ // i & N/i are factr of N
            if(i==N/i){ c=c+1 }
            else { c=c+2 }
        }
    }
    return c;
}
```

i <= √N : square on both sides

i * i <= N

Ass: $10^8$ iterations code 1 sec

approx: i : [1, √N] : √N iterations :

| Input N | iterations | Execution time |
|---|---|---|
| $10^{18}$ | $\sqrt{10^{18}} = 10^9$ | 10sec |
| | $10^9$ = x | x = 10sec |
| | $10^8$ ite = 1sec | |
| | $x * 10^8 = 10^9$ | |
| | $x = \frac{10^9}{10^8} = 10$ | |

**Final Con:** $N = 10^{18}$ : 317 years → 10sec

### Dry Run: N = 36

c = 0

i = 1; i*i <= N; i++; if N%i==0, {i, == N/i} c=c+1 else c=c+2

| | | | | |
|---|---|---|---|---|
| 1 | 1 <= 36 | 36%1==0 | {1 ≠ 36} | c = c+2 = 2 |
| 2 | 4 <= 36 | 36%2==0 | {2 ≠ 18} | c = c+2 = 4 |
| 3 | 9 <= 36 | 36%3==0 | {3 ≠ 12} | c = c+2 = 6 |
| 4 | 16 <= 36 | 36%4==0 | {4 ≠ 9} | c = c+2 = 8 |
| 5 | 25 <= 36 | 36%5 ≠ 0 | | |
| 6 | 36 <= 36 | 36%6==0 | {6 == 6} | c = c+1 = 9 |
| 7 | 49 <= 36 ✗ | come out of loop & return ans = 9 | | |

Target: Code → obsskills, Libraries, DataStructures & Algorithms → ProblemSolver

8:25 → 8:35am

## log Basics:

$\log_b^a$ = { To what power we need to raise b to get a }

$\log_b^a = \log_2^8 = 3 \quad 2^3 = 8$

$\log_3^{27} = 3 \quad 3^3 = 27$

$\log_5^{25} = 2, \quad 5^2 = 25$

$\log_2^{32} = 5, \quad 2^5 = 32$

$\log_4^{16} = 2 \quad 4^2 = 16$

$\log_2^{10}$ = $2^3 = 8$, $2^4 = 16$ → 3 (nyq) → $2^3$ = 10, Integral part = 3

$\log_2^{20}$ = $2^4 = 16$, $2^5 = 32$ → 4 (nyq) → $2^4$ = 20, Integral part = 4

$\log_2^{33}$ = $2^5 = 32$, $2^6 = 64$ → 5 (nyq) → $2^5$ = 33 Integral part = 5

→ $\log_b^a = \log_2^{2^{10}} = 10$ =

Doubts:

$\log_3^{3^4} = 4$

$\log_5^{5^6} = 6$

$\log_5^{34} = \log_5^{81}$ = $5^2 = 25$, $5^3 = 125$ → 2 (nyq) → $5^2$ = 81 Integral = 2

$\log_a^{a^x} = x$

$\log_2^{2^N} = N$

28) Given ar[N] elements & index s & e

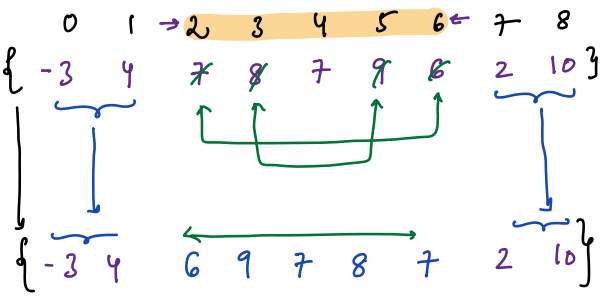
Reverse ar[] from index [s e] Note: s < = e

Eg1:

ar[9] = { -3 4 7 8 7 9 6 2 10 }

s = 2 e = 6

After rev




P1 P2

2 < 6 : swap(ar[2], ar[6])

3 < 5 : swap(ar[3], ar[5])

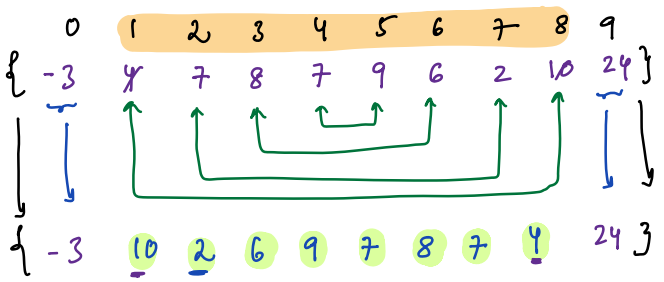
4 = 4 : Stop

Eg2:

ar[10] = { -3 4 7 8 7 9 6 2 10 24 }

s = 1 e = 8

After rev




P1 P2

1 < 8 : swap(ar[1], ar[8])

2 < 7 : swap(ar[2], ar[7])

3 < 6 : swap(ar[3], ar[6])

4 < 5 : swap(ar[4], ar[5])

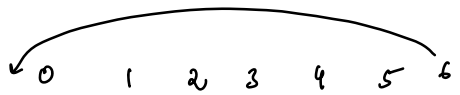
5 > 4 : Break.

```
void RevenRange(int ar[], int s, int e){
    int p1 = s, p2 = e;
    while(p1 < p2){
        // Swap ar[p1] & ar[p2]
            int t = ar[p1]
            ar[p1] = ar[p2]
            ar[p2] = t
        p1 = p1+1  p2 = p2-1
    }
}
```

38) Given ar[N] & k : Rotate array from last to first by k times

Note: You cannot create another array

Eg 1: ar[7] = { 3 -2 1 4 6 9 8 }, k = 3

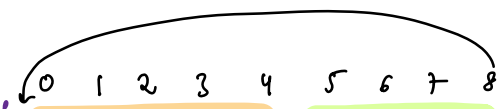



Rotate 1 time : 8 3 -2 1 4 6 9

Rotate 2 time : 9 8 3 -2 1 4 6

Rotate 3rd time : 6 9 8 3 -2 1 4

Eg 2: ar[9] = { 4 1 6 9 2 14 7 8 3 }, k = 4




Rotate 1st time : { 3 4 1 6 9 2 14 7 8 }

Rotate 2nd time : { 8 3 4 1 6 9 2 14 7 }

Rotate 3rd time : { 7 8 3 4 1 6 9 2 14 }

Rotate 4th time : { 14 7 8 3 4 1 6 9 2 }

Eg 3: ar[10] : -2 3 1 4 6 2 8 7 9 3, k = 3


Rotate 1st time : 3 -2 3 1 4 6 2 8 7 9

Rotate 2nd time : 9 3 -2 3 1 4 6 2 8 7

Rotate 3rd time : 7 9 3 -2 3 1 4 6 2 8

k = 5 ar[13] : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$

Step1:

Reverar[] : $a_{12}$ $a_{11}$ $a_{10}$ $a_9$ $a_8$ $a_7$ $a_6$ $a_5$ $a_4$ $a_3$ $a_2$ $a_1$ $a_0$

Reverm first 5 ele  Reven last 8 ele

ar[13] = $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$

ar[13] : $a_8$ $a_9$ $a_{10}$ $a_{11}$ $a_{12}$ $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$

Steps: Rotate ar[N] by k times:

1. Reverm full arr[] : reverse(ar[], 0, N-1)
2. Reven first k ele : reverm(ar[] 0 k-1)
3. Reven last N-k ele : reverm(ar[], k, N-1)

```
void rotate(int ar[], int k){
    int N = ar.length
    k = k%N
    reverserange(ar, 0, N-1)
    reverserange(ar, 0, k-1)
    reverm range(ar, k, N-1)
}
```

Doubts: ar[5] k=7
reverm range(ar, 0, 6) // reversing from 0..6
is not even possible it will go outside array bounds.

```
void Revenrange(int ar[], int s, int e){
    int p1 = s, p2 = e;
    while(p1 < p2){
        // Swap ar[p1] & ar[p2]
        int t = ar[p1]
        ar[p1] = ar[p2]
        ar[p2] = t
        p1 = p1+1  p2 = p2-1
    }
}
```

Obs:

ar[5] : { $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ }

Rotate

0 : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$

1 : $a_4$ $a_0$ $a_1$ $a_2$ $a_3$

2 : $a_3$ $a_4$ $a_0$ $a_1$ $a_2$

3 : $a_2$ $a_3$ $a_4$ $a_0$ $a_1$

4 : $a_1$ $a_2$ $a_3$ $a_4$ $a_0$

5 : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$

| Rotate | k | rotation |
|---|---|---|
| 5 | 10 | rotation = 0 |
| 6 | 11 | rotation = 1 |
| 7 | 12 | rotation = 2 |
| 8 | 13 | rotation = 3 |
| 9 | 14 | rotation = 4 |

k

20 rotation = 0 — 20 % 5 = 1

36 rotation = 1 — 36 % 5 = 1

32 rotation = 2 — 32 % 5 = 1

24 rotation = 4 — 24 % 5 = 1

↳ 5 + 5 + 5 + 5 + 4

(5 * 4 + 4) % 5