- Welcome to PS + DSA Module
- Manisha Pawar
- Graduated from MSIT Delhi (IT department)
- Pepcoding (DSA Instuctor + Content creator)
- 2+ years of Teaching Experience

Programming constructs
Problem solving (efficient)

Q-1 count of factors

12 → 1, 2, 3, 4, 6, 12
24 → 1, 2, 3, 4, 6, 8, 12, 24
10 → 1, 2, 5, 10

N, factors → 1 to N

```
int countfactors (int N) {
    int count = 0;
    for (int i=1; i<= N; i++) {
        if (N % i == 0) {
            // i is a factor of N
            count++;
        }
    }
    return count;
}
```

N = 10

i → 1 2 3 4 5 6 7 8 9 10

count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

iterations : N

Execution time

→ value of N

→ system configuration

Assumption

10^8 iterations per seconds

N	iteration(N)	time
10^8	10^8	1 sec
10^9	10^9	10 sec
10^{18}	10^{18}	10^{10} sec
		$\approx$ 317 years

10^8 itr → 1 sec

1 itr → $\frac{1}{10^8}$ sec

10^9 itr → $\frac{1}{10^8} \times 10^9$

= 10 sec

1 itr → $\frac{1}{10^8}$ sec

10^{18} itr → $\frac{1}{10^8} \times 10^{18}$

= 10^{10} sec

Improvisation

$i * j = N$ (both i and j are factors of N)

$j = \frac{N}{i}$ (both i and N/i are factors of N)

$N = 24$

i	N/i
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

$$i <= \frac{N}{i}$$

$$i * i <= N$$

$$i^2 <= N$$

$$\boxed{i <= \sqrt{N}}$$

$N = 100$

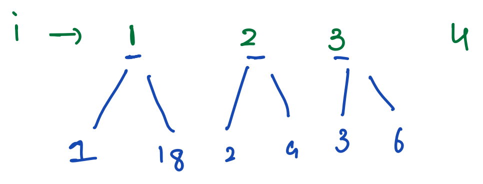
i	N/i
1	100
2	50
4	25
5	20
10	10
20	5
25	4
50	2
100	1

Observation: a) All factors of N are present in first half.

b) we are in first half till $i <= \sqrt{N}$

$N = 18$

loop i: 1 to $\sqrt{N}$



```
int countfactors (int N) {
    int count = 0;
        for( int i = 1; i <= √N ; i++) {
            if (N % i == 0) {
                if ( i == N/i ) {
                    count ++;
                }
                else {
                    // both i and N/i are factors of N
                    count += 2;
                }
            }
        }
    return count;
}
```

N = 24

$\sqrt{24} = 4$

i	count	
1	2	(1, 24)
2	4	(2, 12
3	6	(3, 8
4	8	(4, 6
5	end of loop	

```
int countfactors (int N) {
    int count = 0;
    for( int i= 1; i <= √N ; i++) {
        if (N % i == 0) {
            if ( i == N/i ) {
                count ++;
            }
            else {
                // both i and N/i are factors of N
                count += 2;
            }
        }
    }
    return count;
}
```

N = 36

$\sqrt{36} = 6$

i	count	
1	2	1,36
2	4	2,18
3	6	3,12
4	8	4,9
5		
6	9	6,6

```
int countfactors (int N) {
    int count = 0;
    for(int i=1; i <= √N ; i++) {
        if (N%i == 0) {
            if (i == N/i) {
                count++;
            }
            else {
                // both i and N/i are factors of N
                count += 2;
            }
        }
    }
    return count;
}
```

$N = 100$

$\sqrt{100} = 10$

i	count	i, N/i
1	2	1, 100
2	4	2, 50
3		
4	6	4, 25
5	8	5, 20
6		
7		
8		
9		
10	9	10, 10

iterations : $\sqrt{N}$

N	iterations ($\sqrt{N}$)	time
10^{18}	10^9	10 sec

10^8 itr $\rightarrow$ 1 sec

10^9 itr $\rightarrow \frac{1}{10^8} \times 10^9$

$= 10$ sec

Q.2 Check whether given no. is prime or not.

Prime no. → only two factors (1 and no. itself)

{ 10 , 11 , 23 , 2 , 25 , 27 , 31 }

↳ prime no. : 11 23 2 31

```
boolean isPrime(int N) {
    if (countfactors(N) == 2) {
        return true;
    }
    else {
        return false;
    }
}
```

$\sqrt{N}$ iterations

```
int temp = (int) Math.sqrt(N);
```

$\downarrow$

$\sqrt{N}$

Q-1 Reverse an array.

A = { 10 20 30 40 50 }

0	1	2	3	4

s	e
0	4
1	3
2	2

50	40		20	10
{ ~~10~~	~~20~~	30	~~40~~	~~50~~ }
0	1	2	3	4
		s		
		e		

A = { 10 20 30 40 50 60 }

0	1	2	3	4	5

s	e
0	5
1	4
2	3
3	2

60	50	40	30	20	10
{ ~~10~~	~~20~~	~~30~~	~~40~~	~~50~~	~~60~~ }
0	1	2	3	4	5
		e	s		

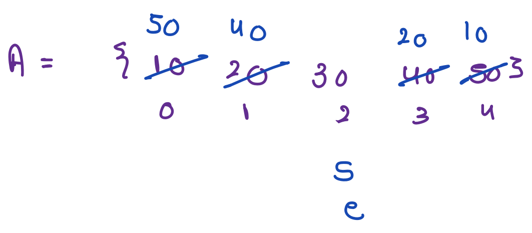
do swap till $s < e$

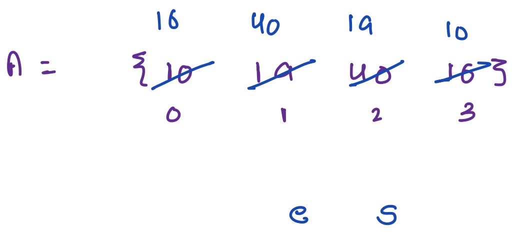
```
void   reverse (int [] A) {
       int  n= A. length;
       int  s = 0;
       int  e = n-1;

       while (s<e) {
            // swap  A[s]  and  A[e]
            int  temp = A[s];
            A[s] = A[e];

            A[e] = temp;

            s++;
            e--;
       }
}
```





Q-4 Reverse part of an Array.

A =	{10	20	19	14	50	60	80	12}
	0	1	2	3	4	5	6	7

s = 2

e = 5

{10	20	60	50	14	19	80	12}
0	1	2	3	4	5	6	7

```
void reverseParst (int[] A, int s, int e) {

    while (s<e) {
        // swap A[s] and A[e]
        int temp = A[s];
        A[s] = A[e];
        A[e] = temp;
        s++;
        e--;
    }
}
```

s = 2

e = 5

			7	6	4	19	
A = {10	20	~~19~~	~~14~~	~~8~~	~~7~~	133
0	1	2	3	4	5	6

Q.6 Given an array, rotate it K times from last to first.

Google, amazon

Note → i) don't create extra array

ii) do it efficient

$$\begin{tikzcd}[column sep=small]
A = & 10 & 20 & 30 & 40 & 50 \arrow[d] \\
& 50 & 10 & 20 & 30 & 40 \arrow[d] \\
& 40 & 50 & 10 & 20 & 30 \arrow[d] \\
& 30 & 40 & 50 & 10 & 20
\end{tikzcd}$$

K = 3

A = 19 18 16 48 36 44 39

K = 4

48 36 44 39 19 18 16

A = 19 13 14 18 16 27 5 K=3

16 27 5 19 13 14 18

rev

$\underbrace{5 \quad 27 \quad 16}_{rev} \quad \underbrace{18 \quad 14 \quad 13 \quad 19}_{rev}$

16 27 5 19 13 14 18

A = 19 18 4 6 8 17 27 k=4

6 8 17 27 19 18 4

rev

$\underbrace{27 \quad 17 \quad 8 \quad 6}_{rev} \quad \underbrace{4 \quad 18 \quad 19}_{rev}$

6 8 17 27 19 18 4

obs:

i) reverse complete array

ii) reverse first k elements

iii) reverse the remaining array.

```
void    rotate (int[] A, int k) {
    int n= A.length;

    reversePart (A, 0, n-1);
    reversePart (A, 0, K-1);
    reversePart (A, k, n-1);

}
```

K=4

A =	10	20	30	19	14	16	18
	0	1	2	3	4	5	6

$\downarrow$

19	14	16	18	10	20	30
~~18~~	~~16~~	~~14~~	~~19~~	~~30~~	~~20~~	~~10~~
0	1	2	3	4	5	6

19	14	16	18	10	20	30

K > A.length

```
void    rotate (int[] A, int k) {
    int n= A.length;
    reversePart (A, 0, n-1); ✓
    reversePart (A, 0, K-1);  }  Array index
    reversePart (A, k, n-1);  }     out of bound
}
```

K= 12

A = 10 20 30 40 50

n= 5

$K = 4$

$n = 4$

$$\begin{tikzcd}
10 \quad 20 \quad 30 \quad 40 \arrow[d] \\
40 \quad 10 \quad 20 \quad 30 \arrow[d] \\
30 \quad 40 \quad 10 \quad 20 \arrow[d] \\
20 \quad 30 \quad 40 \quad 10 \arrow[d] \\
10 \quad 20 \quad 30 \quad 40
\end{tikzcd}$$

$n = 5$

$K = 12$

nr : 2

$n = 6$

$K = 27$

nr : 3

```
void rotate (int[] A, int k) {
    int n = A.length;

    K = k % n;

    reversePart (A, 0, n-1);
    reversePart (A, 0, k-1);
    reversePart (A, k, n-1);
}
```

$n = 5$

$k = 3$

$k = 3 \% 5$

$= 3$

$n = 5$

$k = 29$

$k = 29 \% 5$

$= 4$

log basics

$\log_b a$ (b power what is equals to a)

$$\log_b a = c$$

$$b^c = a$$

$\log_2 8 = 3$

$\log_2 64 = 6$

$\log_{10} 10000 = 4$

$\log_3 81 = 4$

$\log_2 31 = \underline{4.954}$

↳ int $\Rightarrow$ 4

$\log_4 16 = 2$

$\log_3 27 = 3$

* * *

i) $N = 2^k$

$k = \log_2 N$

ii) $\log_b b^N = N$

Problem solving and DSA

- Time complexity
- Arrays : Prefix sum, subarrays, sliding window, 2D matrices.
- Bit manipulation
- Hashing
- Recursion
- Sorting
- Searching
- 2 pointer technique
- Strings
- Linked list
- Trees, BST, heaps
- Dynamic programming
- Graphs

52 classes

Doubts

12 % 5 $\rightarrow$ 2

$$\begin{array}{r} 2 \\ 5\overline{)12} \\ \underline{10} \\ \underline{2} \end{array}$$

3 % 5 $\rightarrow$ 3

$$5\overline{)\;3\;}$$

19	15	16	18	13	35	47
0	1	2	3	4	5	6

k = 5

Print : $\underbrace{n-k \quad \text{to} \quad n-1}_{\text{last k values}}$

Print : $\underbrace{0 \quad \text{to} \quad n-k-1}_{\text{first rem. values}}$