344. Reverse String

Easy 👍 8239 👎 1147 ♡ Add to List ⌷ Share

Write a function that reverses a string. The input string is given as an array of characters s .

You must do this by modifying the input array in-place with O(1) extra memory.

```
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
```
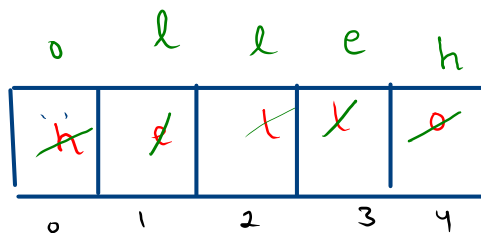
1. no need to mange i/p

2. no need to print o/p.

given.

```java
public void reverseString(char[] s) {

}
```


while (i < j)
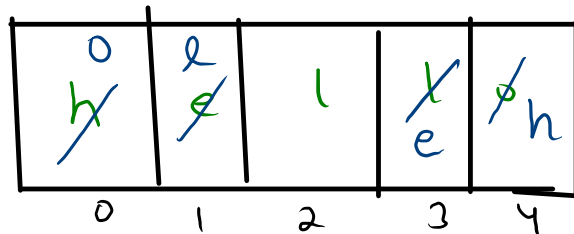
(i ≤ j)

```java
class Solution {
    public void reverseString(char[] s) {
        int i = 0;
        int j = s.length-1;

        while(i < j){
            //swap and manage i and j
            char tmp = s[i];
            s[i] = s[j];
            s[j] = tmp;
            i++;
            j--;
        }
    }
}
```

$s \rightarrow$

h	e	l	e	h

0   1   2   3   4

i

j

$0 < 4$ ✓

$\boxed{2 < 2}$

$1 < 3$ ✓ $tmp = e$

$tmp = h[j]$

$s[i] = s[j]$

$s[j] =$

Pg 16 Q2 $nC_r = \boxed{\dfrac{n!}{r!\,(n-r)!}}$

$5C_3 = \dfrac{5!}{3!\,2!}$

$= \dfrac{5 \times 4 \times 3!}{3! \times 2}$

$= \boxed{10}$

$\rightarrow$ factorial ? $\rightarrow$ ans = factorial of n

n

```java
import java.io.*;
import java.util.*;

public class Solution {
    public static int factorial(int n){
        int fact = 1;
        for(int i = 1; i <= n; i++){
            fact *= i;
        }
        return fact;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int r = scn.nextInt();

        int nFact = factorial(n);      // 120
        int rFact = factorial(r);      // 6
        int nmrFact = factorial(n-r);  // 2

        int ans = (nFact)/(nmrFact * rFact);
        System.out.println(ans);
    }
}
```

$$^{5}C_{3} = \boxed{10} \ ?$$

$$n = 5$$
$$r = 3.$$

$$fact = 1 \times 1 \times 2 \times 3 \times 4 \quad 120$$

$$i = 1 \qquad\qquad 1 \le 5$$
$$2 \qquad\qquad 2 \le 5$$
$$3 \qquad\qquad 3 \le 5 \checkmark$$
$$\frac{120}{6 \times 2} \qquad 4 \qquad\qquad 4 \le 5 \checkmark$$
$$5 \qquad\qquad 5 \le 5$$
$$6 \qquad\qquad \boxed{6 \le 5}$$

Time Complexity. (TC)

TC $\neq$ time to run your program

⊲ TC: any maths function to represent relationship between number of user's i/p & operations

⑤ i/p $\longrightarrow$ operation.

Time Complexity   Why?

(A) solution

TC

SC

(B) solution

TC

SC

3 Notations.

↳ represent

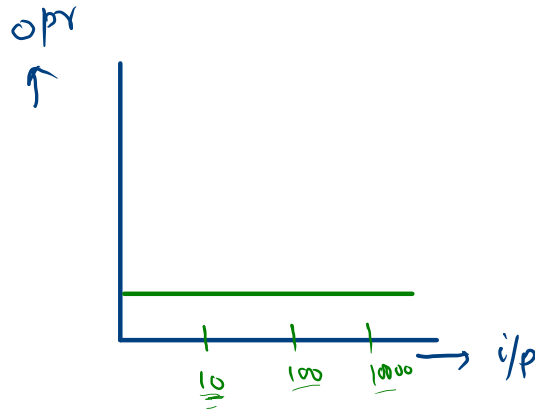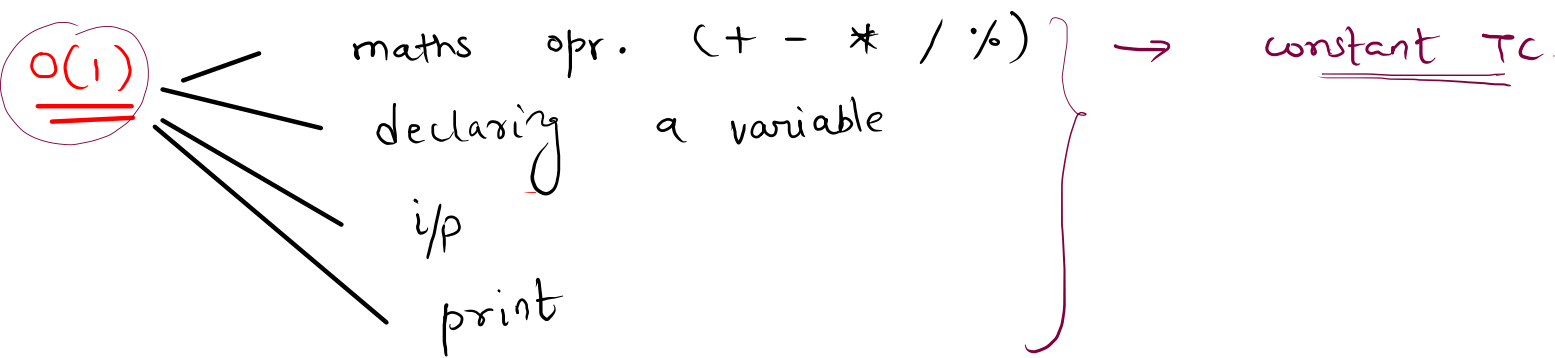ω omega $\longrightarrow$ best case

Θ theta $\longrightarrow$ avg. case

O Oh $\longrightarrow$ worst case.

O(1)

maths opr. (+ - * / %) → constant TC.

declaring a variable

i/p

print

opr
↑

i/p

10   100   10000

$$f(x) = 2x^2 + \underline{\underline{x + 5}}$$

$$\text{degree} = 2.$$

$$Tc = \boxed{n^3} + 2n + 6$$

$$\boxed{n^3}$$

$O(n) \rightarrow$ linear time complexity.

i/p

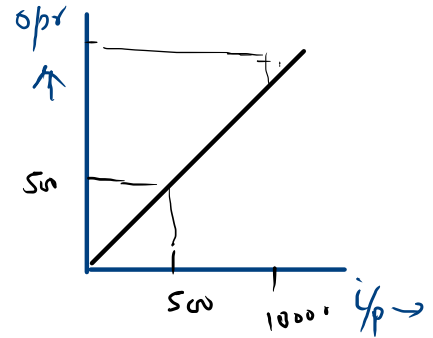$n = 500$ --- 500 opr

$n = 1000$ --- 1000 opr

$n = 10000$ --- 10000 opr.


```java
public static void function(int [] A){
    for(int i = 0; i < A.length; i++){
        System.out.println(A[i]);
    }
}
```

```java
10    public static void function(int n){
11        for(int i = 0; i < n; i++){
12            System.out.println(i);
13        }
14    }
15
```

Que.

```java
public static void function(int n){
    for(int i = 0; i < 10; i++){
        System.out.println(i);
    }
}
```

$O(10) \quad \rightarrow \quad O(1)$

```
10 ▾    public static void function(int n){
11 ▾        for(int i = 0; i < n; i++){
12             int x = scn.nextInt();
13             System.out.println(x);
14         }
15     }
16
```

linear.

$O(1)$ ⎫
         ⎬ $O(1)$
$O(1)$ ⎭

$k_1 + k_2 = \boxed{k}$

1 itr $\longrightarrow$ $O(1)$

n itr $\longrightarrow$ $O(1) \times n$ = $\underline{\underline{O(n)}}$

```java
1  import java.io.*;
2  import java.util.*;
3
4  public class Solution {
5
6      public static void main(String[] args) {
7          Scanner scn = new Scanner(System.in);
8          int x = scn.nextInt();
9          int n = scn.nextInt();
10
11         for(int i = x; i <= n; i++){
12             System.out.println(i);
13         }
14
15     }
```

$$-2^{31} < x \leq 2^{31} - 1$$
$$-2^{31} < n \leq 2^{31} - 1$$

Worst

$$x = -2^{31}$$

$$n = 2^{31} - 1$$

2^n

$2n$

$$n = 2^{31} - 1$$
$$x = -2^{31}$$

$$x = 0$$
$$n = 100$$

$$x = 100$$
$$n = 900$$

$$O(2n)$$

$$O(n)$$

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        int n = scn.nextInt();

        for(int i = 1; i <= 10; i++){
            System.out.println(n+"x" + i + "=" + (n*i));
        }
    }
}
```
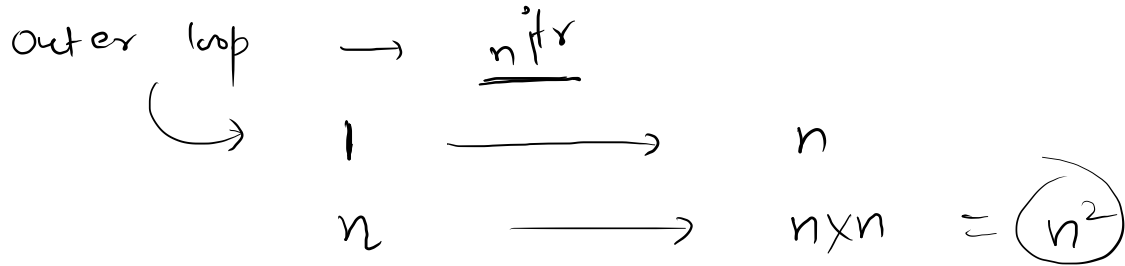
$n = 5$

$n = 50$

$n = 50,000$

10 times. $\longrightarrow$ constant

$O(1)$.

2 loops ?

n^2

```java
public static void function(int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            System.out.println("Geekster");
        }
    }
}
```

Outer loop $\longrightarrow$ n^{itr}

$1 \longrightarrow n$

$n \longrightarrow n \times n = \boxed{n^2}$

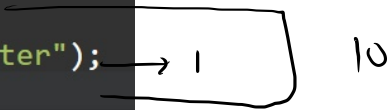$O(n^2)$

```
public static void function(int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 10; j++){
            System.out.println("Geekster");  → 1
        }
    }
}
```

10

1 ——————→ 10

2 ——————→ 10

8 ——————→ 10

⋮

n ——————→ 10

10 + 10 --- + 10
        n terms.

$O(n)$

$10 \times n$

$$1 \leq n \leq 100000$$

```java
public static void function(int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < 10; j++){
            System.out.println("Geekster");
        }
    }
}
```

eg 1.   n = 100   →   opr
                          1000

         n = 10000   →   100000

```java
16   public static void function(int n){
17       for(int i = 0; i < 10; i++){
18           for(int j = 0; j < 10; j++){
19               System.out.println("Geekster");
20           }
21       }
22   }
23
```

n = 100   →   100

n = 10000   →   100

$O(n^2)$ $n = 100$

eg
```java
public static void function(int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            System.out.println("Geekster");
        }
    }
}
```
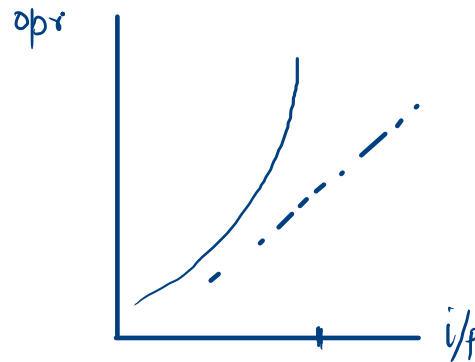
Que.

```java
public static void function(int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            System.out.println("Geekster");
        }
    }
    for(int i = 0; i < n; i++){
        int x = scn.nextInt();
        System.out.println(x);
    }
}
```

$\left.\rule{0pt}{2em}\right\} n^2$

$\left.\rule{0pt}{1em}\right\} n$

$n^2 + n$

$O\left(n^2\right)$

$O(\log n) \longrightarrow O(\log_2 n)$