

2023 Year Sale is Live!



All GeeksforGeeks Courses Upto 25% Off + Additional Cashback Upto 1000 INR [Claim Now!](#)



Courses @SALE Data Structures and Algorithms Array Matrix Strings Hashing Linked List Stack Queue Binary Tree

Practice Questions on Time Complexity Analysis

Difficulty Level : Easy • Last Updated : 21 Dec, 2022

Read

Discuss

Courses @Sale

Practice

Video

Prerequisite: [Analysis of Algorithms](#)

1. What is the time, and space complexity of the following code:

CPP

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

Start Your Coding Journey Now!

[Login](#)[Register](#)

```
a = 0
b = 0
for i in range(N):
    a = a + random()

for i in range(M):
    b = b + random()
```

Options:

1. $O(N * M)$ time, $O(1)$ space
2. $O(N + M)$ time, $O(N + M)$ space
3. $O(N + M)$ time, $O(1)$ space
4. $O(N * M)$ time, $O(N + M)$ space

Output:



Start Your Coding Journey Now!

[Login](#)[Register](#)

3. $O(N + M)$ time, $O(1)$ space

Explanation: The first loop is $O(N)$ and the second loop is $O(M)$. Since **N** and **M** are **independent variables**, so we can't say which one is the leading term. Therefore **Time complexity** of the given problem will be **$O(N+M)$** .

Since variables size does not depend on the size of the input, therefore **Space Complexity** will be **constant or $O(1)$**

2. What is the time complexity of the following code:

CPP

```
int a = 0;
for (i = 0; i < N; i++) {
```

Start Your Coding Journey Now!

Login

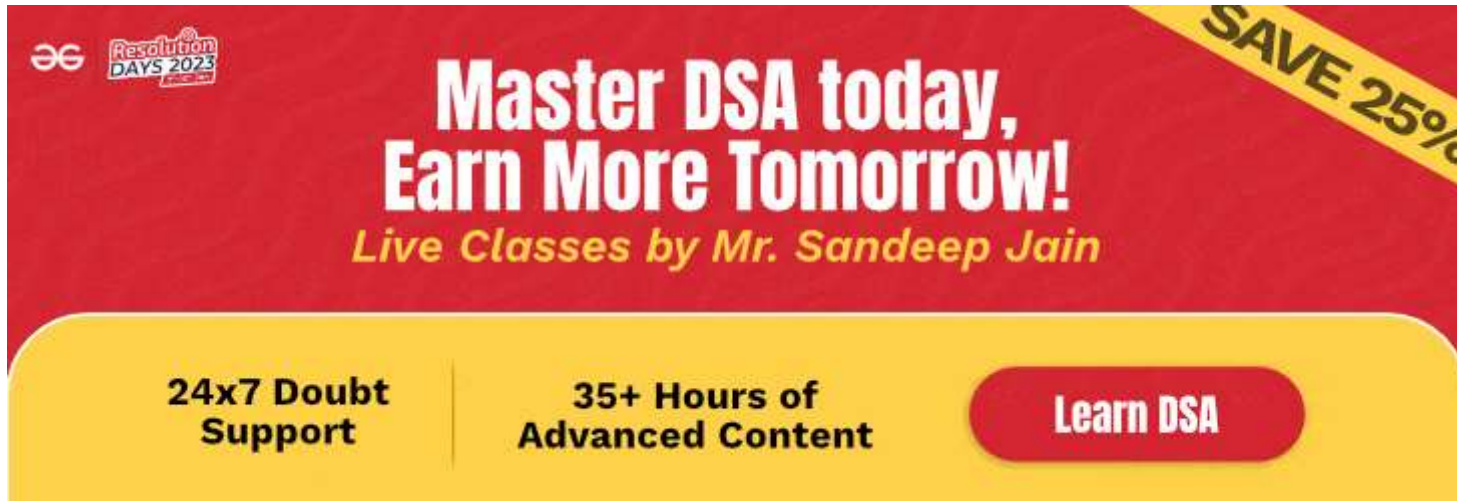
Register

```
}
```

Python3

```
a = 0;
for i in range(N):
    for j in reversed(range(i,N)):
        a = a + i + j;
```

Options:



The advertisement features a red background with white and yellow text. At the top left, there is a logo for 'Resolution DAYS 2023'. The main headline reads 'Master DSA today, Earn More Tomorrow!' in large white letters, followed by 'Live Classes by Mr. Sandeep Jain' in smaller yellow letters. A yellow diagonal banner in the top right corner says 'SAVE 25%'. The bottom section is yellow and contains three elements: '24x7 Doubt Support', '35+ Hours of Advanced Content', and a red button with white text that says 'Learn DSA'.

**Master DSA today,
Earn More Tomorrow!**
Live Classes by Mr. Sandeep Jain

24x7 Doubt Support | **35+ Hours of Advanced Content** | **Learn DSA**

Start Your Coding Journey Now!

Login

Register

3. $O(N * \text{Sqrt}(N))$

4. $O(N*N)$

Output:

4. $O(N*N)$

Explanation:

The above code runs total no of times

$= N + (N - 1) + (N - 2) + \dots 1 + 0$

$= N * (N + 1) / 2$

$= 1/2 * N^2 + 1/2 * N$

$O(N^2)$ times.

3. What is the time complexity of the following code:

CPP

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

```
k = 0;
for i in range(n//2,n):
    for j in range(2,n,pow(2,j)):
        k = k + n / 2;
```

Options:

1. $O(n)$
2. $O(N \log N)$
3. $O(n^2)$
4. $O(n^2 \log n)$

Output:

2. $O(n \log n)$

Explanation: If you notice, j keeps doubling till it is less than or equal to n . Several times, we can double a number till it is less than n would be $\log(n)$.

Let's take the examples here.

for $n = 16$, $j = 2, 4, 8, 16$

for $n = 32$, $j = 2, 4, 8, 16, 32$

So, j would run for $O(\log n)$ steps.



Start Your Coding Journey Now!

Login

Register

4. What does it mean when we say that an algorithm X is asymptotically more efficient than Y?

Options:



- 1. X will always be a better choice for small inputs
- 2. X will always be a better choice for large inputs
- 3. Y will always be a better choice for small inputs
- 4. X will always be a better choice for all inputs

Output:

- 2. X will always be a better choice for large inputs



Start Your Coding Journey Now!

[Login](#)[Register](#)

larger than a value n_0 where $n_0 > 0$.

5. What is the time complexity of the following code:

CPP

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

Python3

```
a = 0
i = N
while (i > 0):
    a += i
    i //= 2
```

Options:



Start Your Coding Journey Now!

Login

Register

3. $O(N / 2)$

4. $O(\log N)$

Output:

4. $O(\log N)$

Explanation: We have to find the smallest x such that ' $(N / 2^x) < 1$ OR $2^x > N$ '

$x = \log(N)$

6. Which of the following best describes the useful criterion for comparing the efficiency of algorithms?

1. Time

2. Memory

3. Both of the above

4. None of the above

3. Both of the above

Explanation: Comparing the efficiency of an algorithm depends on the time and memory taken by an algorithm. **The algorithm which runs in lesser time and** takes less memory even for a large input size is considered a more efficient algorithm.



Start Your Coding Journey Now!

[Login](#)[Register](#)

1. By counting the number of algorithms in an algorithm.
2. By counting the number of primitive operations performed by the algorithm on a given input size.
3. By counting the size of data input to the algorithm.
4. None of the above

2. By counting the number of primitive operations performed by the algorithm on a given input size.

8. What will be the time complexity of the following code?

Javascript

```
for(var i=0;i<n;i++)  
    i*=k
```

C++

```
for(int i=0;i<n;i++){  
    i*=k;  
}
```

Python3

```
# code  
for i in range(n):
```

Start Your Coding Journey Now!

Login

Register

1. $O(n)$
2. $O(k)$
3. $O(\log_k n)$
4. $O(\log_n k)$

Output:

3. $O(\log_k n)$

Explanation: Because loops for the k^{n-1} times, so after taking log it becomes $\log_k n$.

9. What will be the time complexity of the following code?

Javascript

```
var value = 0;
for(var i=0;i<n;i++)
    for(var j=0;j<i;j++)
        value += 1;
```

C++

```
int value = 0;
for(int i=0;i<n;i++)
    for(int j=0;j<i;j++)
```



Python3

```
value = 0;
for i in range(n):
    for j in range(i):
        value=value+1
```

1. n
2. (n+1)
3. n(n-1)
4. n(n+1)

Output:

3. n(n-1)

Explanation: First for loop will run for (n) times and another for loop will be run for (n-1) times as the inner loop will only run till the range i which is 1 less than n , so overall time will be n(n-1).

10. Algorithm A and B have a worst-case running time of $O(n)$ and $O(\log n)$, respectively. Therefore, algorithm B always runs faster than algorithm A.

1. True
2. False



Start Your Coding Journey Now!

Login

Register

Explanation: The Big-O notation provides an asymptotic comparison in the running time of algorithms. For $n < n^0$, algorithm A might run faster than algorithm B, for instance.

Recommended

Solve DSA problems on GfG Practice.

Solve Problems



Like 540

Next

**Sample Practice Problems on Complexity
Analysis of Algorithms**



Start Your Coding Journey Now!

[Login](#)[Register](#)

1. Sample Practice Problems on Complexity Analysis of Algorithms

2. Asymptotic Analysis (Based on input size) in Complexity Analysis of Algorithms

3. Time Complexity and Space Complexity

4. Time Complexity Analysis | Tower Of Hanoi (Recursion)

5. Step Count Method for Time Complexity Analysis

6. What are Asymptotic Notations in Complexity Analysis of Algorithms

7. How to Analyse Loops for Complexity Analysis of Algorithms

8. Complexity Analysis of Binary Search

9. Complexity analysis of various operations of Binary Min Heap

10. Prune-and-Search | A Complexity Analysis Overview



Article Contributed By :