Window Sliding Technique

Difficulty Level : Easy • Last Updated : 10 Jul, 2021

This technique shows how a nested for loop in some problems can be converted to a single for loop to reduce the time complexity.
Let's start with a problem for illustration where we can apply this technique –

```
Given an array of integers of size 'n'.
Our aim is to calculate the maximum sum of 'k'
consecutive elements in the array.

Input  : arr[] = {100, 200, 300, 400}
         k = 2
Output : 700

Input  : arr[] = {1, 4, 2, 10, 23, 3, 1, 0, 20}
         k = 4
Output : 39
We get maximum sum by adding subarray {4, 2, 10, 23}
of size 4.

Input  : arr[] = {2, 3}
         k = 3
Output : Invalid
There is no subarray of size 3 as size of whole
array is 2.
```

So, let's analyze the problem with **Brute Force Approach**. We start with first index and sum till **k-th** element. We do it for all possible consecutive blocks or groups of k elements. This method requires nested for loop, the outer for loop starts with the

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

starting element of the block of k elements and the inner or the nested loop will add up till the k-th element.

Attention reader! Don't stop learning now. Get hold of all the important DSA concepts with the **DSA Self Paced Course** at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course.**

In case you wish to attend **live classes** with experts, please refer **DSA Live Classes for Working Professionals** and **Competitive Programming Live for Students**.

Consider the below implementation :

C++

```
// O(n*k) solution for finding maximum sum of
// a subarray of size k
#include <bits/stdc++.h>
using namespace std;

// Returns maximum sum in a subarray of size k.
int maxSum(int arr[], int n, int k)
{
    // Initialize result
    int max_sum = INT_MIN;

    // Consider all blocks starting with i.
    for (int i = 0; i < n - k + 1; i++) {
        int current_sum = 0;
        for (int j = 0; j < k; j++)
            current_sum = current_sum + arr[i + j];

        // Update result if required.
        max_sum = max(current_sum, max_sum);
    }

    return max_sum;
}
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
    int arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
    int k = 4;
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << maxSum(arr, n, k);
    return 0;
}
```

Java

```
// Java code here
// O(n*k) solution for finding
// maximum sum of a subarray
// of size k
class GFG {
    // Returns maximum sum in
    // a subarray of size k.
    static int maxSum(int arr[], int n, int k)
    {
        // Initialize result
        int max_sum = Integer.MIN_VALUE;

        // Consider all blocks starting with i.
        for (int i = 0; i < n - k + 1; i++) {
            int current_sum = 0;
            for (int j = 0; j < k; j++)
                current_sum = current_sum + arr[i + j];

            // Update result if required.
            max_sum = Math.max(current_sum, max_sum);
        }

        return max_sum;
    }

    // Driver code
    public static void main(String[] args)
    {
        int arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
        int k = 4;
        int n = arr.length;
        System.out.println(maxSum(arr, n, k));
    }
}

// This code is contributed
// by  prerna saini.
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

Python

```python
# code
import sys
print "GFG"
# O(n * k) solution for finding
# maximum sum of a subarray of size k
INT_MIN = -sys.maxsize - 1

# Returns maximum sum in a
# subarray of size k.


def maxSum(arr, n, k):

    # Initialize result
    max_sum = INT_MIN

    # Consider all blocks
    # starting with i.
    for i in range(n - k + 1):
        current_sum = 0
        for j in range(k):
            current_sum = current_sum + arr[i + j]

        # Update result if required.
        max_sum = max(current_sum, max_sum)

    return max_sum


# Driver code
arr = [1, 4, 2, 10, 2,
       3, 1, 0, 20]
k = 4
n = len(arr)
print(maxSum(arr, n, k))

# This code is contributed by mits
```

C#

```csharp
// C# code here O(n*k) solution for
// finding maximum sum of a subarray
// of size k
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
    // Returns maximum sum in a
    // subarray of size k.
    static int maxSum(int[] arr, int n, int k)
    {
        // Initialize result
        int max_sum = int.MinValue;

        // Consider all blocks starting
        // with i.
        for (int i = 0; i < n - k + 1; i++) {
            int current_sum = 0;
            for (int j = 0; j < k; j++)
                current_sum = current_sum + arr[i + j];

            // Update result if required.
            max_sum = Math.Max(current_sum, max_sum);
        }

        return max_sum;
    }

    // Driver code
    public static void Main()
    {
        int[] arr = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
        int k = 4;
        int n = arr.Length;
        Console.WriteLine(maxSum(arr, n, k));
    }
}

// This code is contributed by anuj_67.
```

PHP

```
<?php
    // code
?>
<?php
// O(n*k) solution for finding maximum sum of
// a subarray of size k

// Returns maximum sum in a subarray of size k.
function maxSum($arr, $n, $k)
{
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
    // Consider all blocks
    // starting with i.
    for ( $i = 0; $i < $n - $k + 1; $i++)
    {
        $current_sum = 0;
        for ( $j = 0; $j < $k; $j++)
            $current_sum = $current_sum +
                                $arr[$i + $j];

        // Update result if required.
        $max_sum = max($current_sum, $max_sum );
    }

    return $max_sum;
}

    // Driver code
    $arr = array(1, 4, 2, 10, 2, 3, 1, 0, 20);
    $k = 4;
    $n = count($arr);
    echo maxSum($arr, $n, $k);

// This code is contributed by anuj_67.
?>
```

Javascript

```
<script>

// O(n*k) solution for finding maximum sum of
// a subarray of size k

// Returns maximum sum in a subarray of size k.
function maxSum( arr, n, k){
    // Initialize result
    let max_sum = Number.MIN_VALUE;

    // Consider all blocks starting with i.
    for (let i = 0; i < n - k + 1; i++) {
        let current_sum = 0;
        for (let j = 0; j < k; j++)
            current_sum = current_sum + arr[i + j];

        // Update result if required.
        max_sum = Math.max(current_sum, max_sum);
    }
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
}

// Driver code
let arr = [ 1, 4, 2, 10, 2, 3, 1, 0, 20 ];
let k = 4;
let n = arr.length;
document.write(maxSum(arr, n, k));

</script>
```

Output

```
24
```

It can be observed from the above code that the time complexity is **O(k*n)** as it contains two nested loops.

Window Sliding Technique

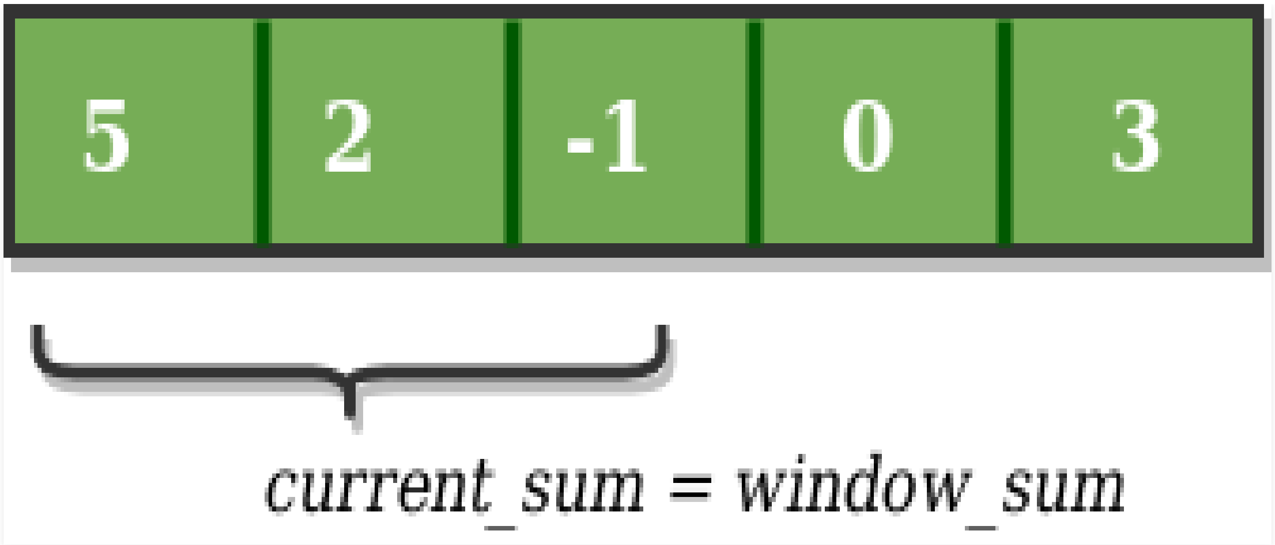
The technique can be best understood with the window pane in bus, consider a window of length **n** and the pane which is fixed in it of length **k**. Consider, initially the pane is at extreme left i.e., at 0 units from the left. Now, co-relate the window with array arr[] of size n and pane with current_sum of size k elements. Now, if we apply force on the window such that it moves a unit distance ahead. The pane will cover next **k** consecutive elements.
Consider an array **arr[]** = {5, 2, -1, 0, 3} and value of **k** = 3 and **n** = 5
Applying sliding window technique :

1. We compute the sum of first k elements out of n terms using a linear loop and store the sum in variable window_sum.
2. Then we will graze linearly over the array till it reaches the end and simultaneously keep track of maximum sum.
3. To get the current sum of block of k elements just subtract the first element from the previous block and add the last element of the current block .

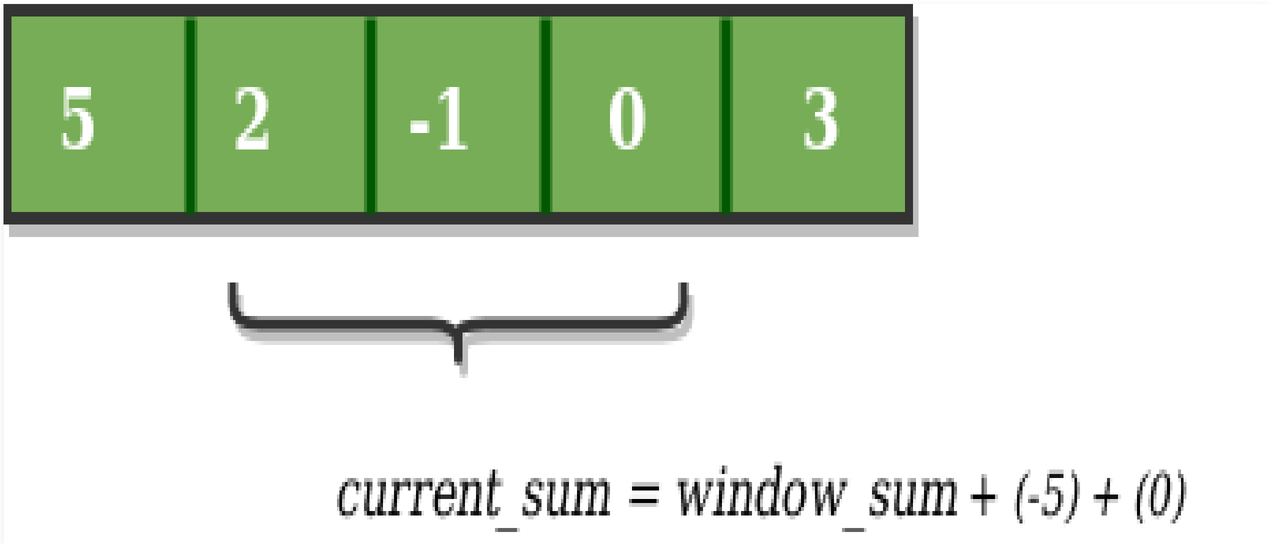
The below representation will make it clear how the window slides over the array.
This is the initial phase where we have calculated the initial window sum starting from index 0 . At this stage the window sum is 6. Now, we set the maximum_sum as current window i.e 6.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !



Now, we slide our window by a unit index. Therefore, now it discards 5 from the window and adds 0 to the window. Hence, we will get our new window sum by subtracting 5 and then adding 0 to it. So, our window sum now becomes 1. Now, we will compare this window sum with the maximum_sum. As it is smaller we wont the change the maximum_sum.

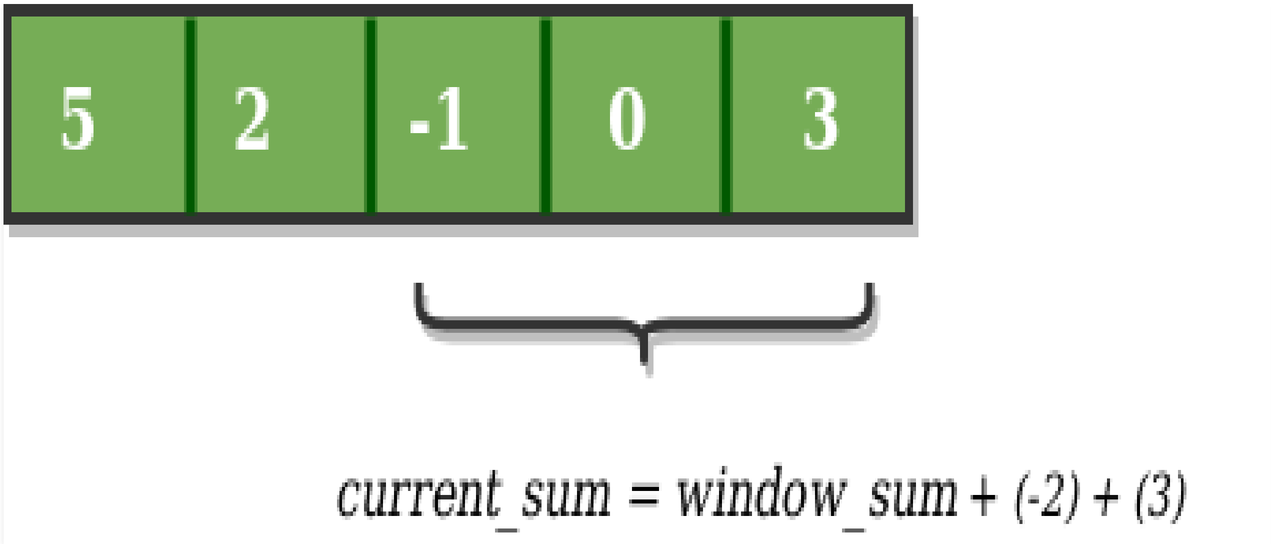


Similarly, now once again we slide our window by a unit index and obtain the new window sum to be 2. Again we check if this current window sum is greater than the maximum_sum till now. Once, again it is smaller so we don't change the maximum_sum.

Therefore, for the above array our maximum_sum is 6.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !



code for the above description :

C++

```
// O(n) solution for finding maximum sum of
// a subarray of size k
#include <iostream>
using namespace std;

// Returns maximum sum in a subarray of size k.
int maxSum(int arr[], int n, int k)
{
    // n must be greater
    if (n < k) {
        cout << "Invalid";
        return -1;
    }

    // Compute sum of first window of size k
    int max_sum = 0;
    for (int i = 0; i < k; i++)
        max_sum += arr[i];

    // Compute sums of remaining windows by
    // removing first element of previous
    // window and adding last element of
    // current window.
    int window_sum = max_sum;
    for (int i = k; i < n; i++) {
        window_sum += arr[i] - arr[i - k];
        max_sum = max(max_sum, window_sum);
    }
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
}

// Driver code
int main()
{
    int arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
    int k = 4;
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << maxSum(arr, n, k);
    return 0;
}
```

Java

```
// Java code for
// O(n) solution for finding
// maximum sum of a subarray
// of size k
class GFG {

    // Returns maximum sum in
    // a subarray of size k.
    static int maxSum(int arr[], int n, int k)
    {
        // n must be greater
        if (n < k) {
            System.out.println("Invalid");
            return -1;
        }

        // Compute sum of first window of size k
        int max_sum = 0;
        for (int i = 0; i < k; i++)
            max_sum += arr[i];

        // Compute sums of remaining windows by
        // removing first element of previous
        // window and adding last element of
        // current window.
        int window_sum = max_sum;
        for (int i = k; i < n; i++) {
            window_sum += arr[i] - arr[i - k];
            max_sum = Math.max(max_sum, window_sum);
        }

        return max_sum;
    }
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
    public static void main(String[] args)
    {
        int arr[] = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
        int k = 4;
        int n = arr.length;
        System.out.println(maxSum(arr, n, k));
    }
}

// This code is contributed
// by  prerna saini.
```

Python

```
# O(n) solution for finding
# maximum sum of a subarray of size k


def maxSum(arr, k):
    # length of the array
    n = len(arr)

    # n must be greater than k
    if n < k:
        print("Invalid")
        return -1

    # Compute sum of first window of size k
    window_sum = sum(arr[:k])

    # first sum available
    max_sum = window_sum

    # Compute the sums of remaining windows by
    # removing first element of previous
    # window and adding last element of
    # the current window.
    for i in range(n - k):
        window_sum = window_sum - arr[i] + arr[i + k]
        max_sum = max(window_sum, max_sum)

    return max_sum


# Driver code
arr = [1, 4, 2, 10, 2, 3, 1, 0, 20]
k = 4
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
# This code is contributed by Kyle McClay
```

C#

```
// C# code for O(n) solution for finding
// maximum sum of a subarray of size k
using System;

class GFG {

    // Returns maximum sum in
    // a subarray of size k.
    static int maxSum(int[] arr, int n, int k)
    {

        // n must be greater
        if (n < k) {
            Console.WriteLine("Invalid");
            return -1;
        }

        // Compute sum of first window of size k
        int max_sum = 0;
        for (int i = 0; i < k; i++)
            max_sum += arr[i];

        // Compute sums of remaining windows by
        // removing first element of previous
        // window and adding last element of
        // current window.
        int window_sum = max_sum;
        for (int i = k; i < n; i++) {
            window_sum += arr[i] - arr[i - k];
            max_sum = Math.Max(max_sum, window_sum);
        }

        return max_sum;
    }

    // Driver code
    public static void Main()
    {
        int[] arr = { 1, 4, 2, 10, 2, 3, 1, 0, 20 };
        int k = 4;
        int n = arr.Length;
        Console.WriteLine(maxSum(arr, n, k));
    }
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

```
// This code is contributed by anuj_67.
```

PHP

```php
<?php
// O(n) solution for finding maximum sum of
// a subarray of size k

// Returns maximum sum in a
// subarray of size k.
function maxSum( $arr, $n, $k)
{
    // n must be greater
    if ($n < $k)
    {
        echo "Invalid";
        return -1;
    }

    // Compute sum of first
    // window of size k
    $max_sum = 0;
    for($i = 0; $i < $k; $i++)
    $max_sum += $arr[$i];

    // Compute sums of remaining windows by
    // removing first element of previous
    // window and adding last element of
    // current window.
    $window_sum = $max_sum;
    for ($i = $k; $i < $n; $i++)
    {
        $window_sum += $arr[$i] - $arr[$i - $k];
        $max_sum = max($max_sum, $window_sum);
    }

    return $max_sum;
}

    // Driver code
    $arr = array(1, 4, 2, 10, 2, 3, 1, 0, 20);
    $k = 4;
    $n = count($arr);
    echo maxSum($arr, $n, $k);

// This code is contributed by anuj_67
?>
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

Javascript

```javascript
// Javascript code for
// O(n) solution for finding
// maximum sum of a subarray
// of size k
function maxSumofK(arr, k) {
let max = 0;
let sum = 0;
//find initial sum of first k elements
for(let n = 0; n <  k ; n++) {
    sum +=  arr[n];
}
//iterate the array once and increment the right edge
 for(let i = k; i < arr.length; i++) {
        sum += arr[i] - arr[i-k];
        //compare if sum is more than max, if yes then replace max with new sum
           if(sum > max) {
               max = sum;
           }
        }
    return max;
}

let arr = [1, 4, 2, 10, 2, 3, 1, 0, 20 ];
console.log(maxSumofK(arr, 4))
//output 28
```

Output

```
24
```

Now, it is quite obvious that the Time Complexity is linear as we can see that only one loop runs in our code. Hence, our Time Complexity is **O(n)**.
We can use this technique to find max/min k-subarray, XOR, product, sum, etc. Refer sliding window problems for such problems.
https://youtu.be/9-3BXsfrpbY?list=PLqM7alHXFySEQDk2MDfbwEdjd2svVJH9p

This article is contributed by **Kanika Thakral**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Like 0

Previous

StAX XML Parser in Java

Next

Finding sum of digits of a number until sum becomes single digit

RECOMMENDED ARTICLES

Page : 1 2 3

01 Sliding Window Maximum (Maximum of all subarrays of size k)
27, May 11

02 Sliding Window Maximum (Maximum of all subarrays of size k) using stack in O(n) time
25, Apr 19

03 Median of sliding window in an array
13, Jan 20

04 Sliding Window Maximum : Set 2
05, Apr 20

05 Median of sliding window in an array | Set 2
13, Jul 20

06 Maximum possible sum of a window in an array such that elements of same window in other array are unique
04, Aug 17

07 Difference between window.onkeypress and window.document.body.onkeypress
16, Jan 21

08 Sliding Up Screen in Android
07, Feb 21

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : vt_m, AnnieAlford, Mithun Kumar, KyleMcClay, Ekta Arora, Parimal7, maartenjjacobs, rohitsingh07052, GauravMaheshwari1

Article Tags : sliding-window, Arrays, Technical Scripter

Practice Tags : sliding-window, Arrays

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !

@geeksforgeeks , Some rights reserved

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our Cookie Policy & Privacy Policy

Got It !