









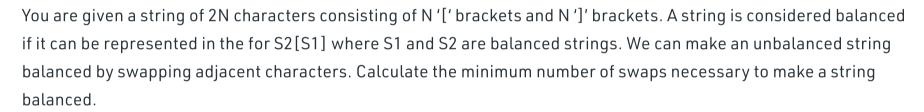


Related Articles

Save for later

## Minimum Swaps for Bracket Balancing

Difficulty Level: Medium • Last Updated: 02 Jun, 2021



#### **Examples:**

Input : []][][

Output : 2

First swap: Position 3 and 4

[][]][

Second swap: Position 5 and 6

[][][]

Input : [[][]]



Output: 0

The string is already balanced.

Recommended: Please solve it on "PRACTICE" first, before moving on to the solution.

We can solve this problem by using greedy strategies. If the first X characters form a balanced string, we can neglect these characters and continue on. If we encounter a ']' before the required '[', then we must start swapping elements to balance the string.

#### **Naive Approach**

Initialize sum = 0 where **sum** stores result. Go through the string maintaining a **count** of the number of '[' brackets encountered. Reduce this count when we encounter a ']' character. If the count hits negative, then we must start balancing the string.

Let index 'i' represent the position we are at. We now move forward to the next '[' at index j. Increase sum by j - i. Move the '[' at position j, to position i, and shift all other characters to the right. Set the count back to 0 and continue traversing the string. In the end, 'sum' will have the required value.

Time Complexity =  $O(N^2)$ 

Extra Space = O(1)









#### Optimized approach

We can initially go through the string and store the positions of '[' in a vector say '**pos**'. Initialize 'p' to 0. We shall use p to traverse the vector 'pos'. Similar to the naive approach, we maintain a count of encountered '[' brackets. When we encounter a '[' we increase the count and increase 'p' by 1. When we encounter a ']' we decrease the count. If the count ever goes negative, this means we must start swapping. The element pos[p] tells us the index of the next '['. We increase the sum by pos[p] - i, where i is the current index. We can swap the elements in the current index and pos[p] and reset the count to 0 and increment p so that it pos[p] indicates to the next '['.

Since we have converted a step that was O(N) in the naive approach, to an O(1) step, our new time complexity reduces.

Time Complexity = O(N)Extra Space = O(N)

#### **C++**





// C++ program to count swaps required to balance string



#include <iostream>
#include <vector>
#include <algorithm>

```
using namespace std;
long swapCount(string s)
    vector<int> pos;
    for (int i = 0; i < s.length(); ++i)</pre>
        if (s[i] == '[')
             pos.push back(i);
    int count = 0; // To count number of encountered '['
    int p = 0; // To track position of next '[' in pos
    long sum = 0; // To store result
    for (int i = 0; i < s.length(); ++i)</pre>
        if (s[i] == '[')
             ++count;
             ++p;
         else if (s[i] == ']')
             --count;
        if (count < 0)</pre>
             sum += pos[p] - i;
             swap(s[i], s[pos[p]]);
             ++p;
             count = 1;
```



```
}
    return sum;
}

// Driver code
int main()
{
    string s = "[]][][";
    cout << swapCount(s) << "\n";

    s = "[[][]]";
    cout << swapCount(s) << "\n";
    return 0;
}</pre>
```

#### Java

```
// Java program to count swaps
// required to balance string
import java.util.*;

class GFG{

// Function to calculate swaps required
public static long swapCount(String s)
{

// Keep track of '['
    Vector<Integer> pos = new Vector<Integer>();
    for(int i = 0; i < s.length(); ++i)
        if (s.charAt(i) == '[')
            pos.add(i);

// To count number of encountered '['</pre>
```



```
int count = 0;
int p = 0;
long sum = 0;
char[] S = s.toCharArray();
for(int i = 0; i < s.length(); ++i)</pre>
    if (S[i] == '[')
        ++count;
        ++p;
    else if (S[i] == ']')
        --count;
    if (count < 0)</pre>
        sum += pos.get(p) - i;
        char temp = S[i];
        S[i] = S[pos.get(p)];
        S[pos.get(p)] = temp;
        ++p;
```



## Python3

```
# Python3 Program to count
# swaps required to balance
# string

# Function to calculate
# swaps required
def swapCount(s):

# Keep track of '['
pos = []

for i in range(len(s)):
    if(s[i] == '['):
```



```
pos.append(i)
count = 0
p = 0
# To store result
sum = 0
s = list(s)
for i in range(len(s)):
    if(s[i] == '['):
        count += 1
        p += 1
    elif(s[i] == ']'):
        count -= 1
    # We have encountered an
    if(count < 0):</pre>
        sum += pos[p] - i
        s[i], s[pos[p]] = (s[pos[p]],
                           s[i])
        p += 1
```



```
# Reset count to 1
    count = 1
return sum

# Driver code
s = "[]][]["
print(swapCount(s))

s = "[[][]]"
print(swapCount(s))

# This code is contributed by avanitrachhadiya2155
```

#### C#



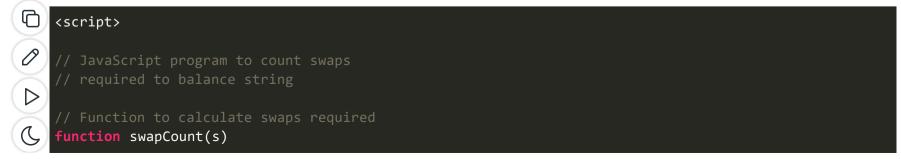


```
int count = 0;
int p = 0;
long sum = 0;
char[] S = s.ToCharArray();
for(int i = 0; i < S.Length; i++)</pre>
    if (S[i] == '[')
        ++count;
        ++p;
    else if (S[i] == ']')
        --count;
    if (count < 0)</pre>
```



```
sum += pos[p]-i;
            char temp = S[i];
            S[i] = S[pos[p]];
           S[pos[p]] = temp;
            ++p;
            count = 1;
   return sum;
static void Main()
   string s = "[]][][";
   Console.WriteLine(swapCount(s));
   s = "[[][]]";
   Console.WriteLine(swapCount(s));
```

## **Javascript**





```
let pos = [];
for(let i = 0; i < s.length; ++i)</pre>
    if (s[i] == '[')
        pos.push(i);
let count = 0;
let p = 0;
let sum = 0;
let S = s.split('');
for(let i = 0; i < s.length; ++i)</pre>
    if (S[i] == '[')
        ++count;
         ++p;
    else if (S[i] == ']')
         --count;
    if (count < 0)</pre>
```



```
sum += pos[p] - i;
            let temp = S[i];
            S[i] = S[pos[p]];
            S[pos[p]] = temp;
            ++p;
            count = 1;
   return sum;
   let s = "[]][][";
    document.write(swapCount(s) + "<br/>");
   s = "[[][]]";
    document.write(swapCount(s));
</script>
```

#### **Output:**

2

0



#### **Another Method:**

```
Time Complexity = O(N)
Extra Space = O(1)
We can do without having to store the positions of '['.
```

Below is the implementation:

#### C++

```
#include <bits/stdc++.h>
using namespace std;
long swapCount(string chars)
    int countLeft = 0, countRight = 0;
    int swap = 0 , imbalance = 0;
    for(int i = 0; i < chars.length(); i++)</pre>
        if (chars[i] == '[')
             countLeft++;
```

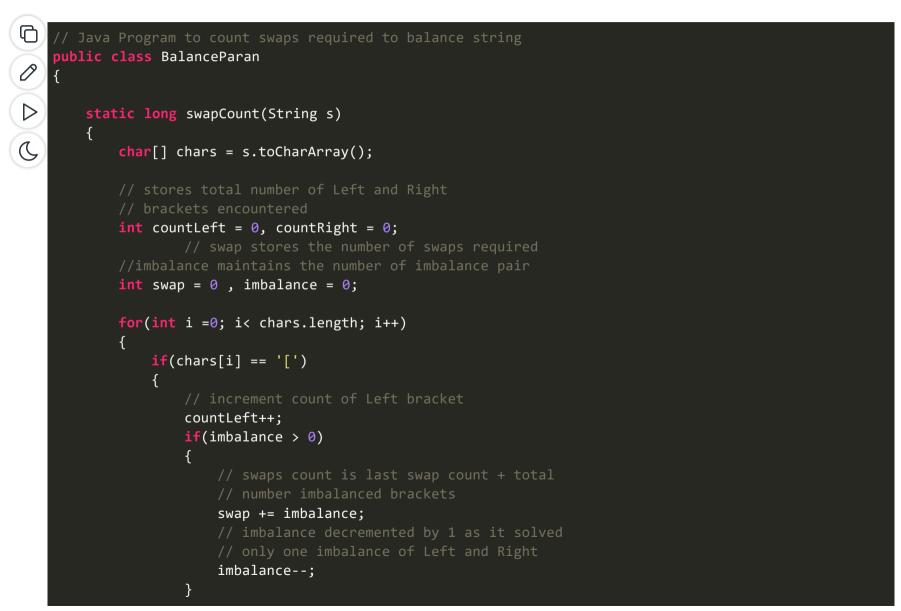


```
(imbalance > 0)
                // number imbalanced brackets
                 swap += imbalance;
                 imbalance--;
        else if(chars[i] == ']' )
             countRight++;
            imbalance = (countRight - countLeft);
    return swap;
int main()
    string s = "[]][][";
    cout << swapCount(s) << endl;</pre>
    s = "[[][]]";
    cout << swapCount(s) << endl;</pre>
    return 0;
```



```
// This code is contributed by divyeshrabadiya07
```

#### Java





```
} else if(chars[i] == ']' )
            countRight++;
            imbalance = (countRight-countLeft);
    return swap;
public static void main(String args[])
    String s = "[]][][";
    System.out.println(swapCount(s) );
    s = "[[][]]";
    System.out.println(swapCount(s) );
```

## Python3

```
# Python3 program to count swaps required to
# balance string
def swapCount(s):

Chars = s

# Stores total number of left and
# right brackets encountered
```



```
countLeft = 0
countRight = 0
swap = 0
imbalance = 0;
for i in range(len(chars)):
    if chars[i] == '[':
        # Increment count of left bracket
        countLeft += 1
        if imbalance > 0:
            # imbalanced brackets
            swap += imbalance
            imbalance -= 1
    elif chars[i] == ']':
        countRight += 1
        # difference between left and
        imbalance = (countRight - countLeft)
```



```
return swap

# Driver code
s = "[]][][";
print(swapCount(s))

s = "[[][]]";
print(swapCount(s))

# This code is contributed by Prateek Gupta
```

#### C#

```
using System;
class GFG
public static long swapCount(string s)
   char[] chars = s.ToCharArray();
   int countLeft = 0, countRight = 0;
   int swap = 0, imbalance = 0;
   for (int i = 0; i < chars.Length; i++)</pre>
```



```
(chars[i] == '[')
            countLeft++;
           if (imbalance > 0)
               swap += imbalance;
               imbalance--;
       else if (chars[i] == ']')
            countRight++;
           imbalance = (countRight - countLeft);
   return swap;
public static void Main(string[] args)
   string s = "[]][][";
   Console.WriteLine(swapCount(s));
   s = "[[][]]";
   Console.WriteLine(swapCount(s));
```



```
}
// This code is contributed by Shrikant13
```

### **Javascript**

```
<script>
    function swapCount(s)
        let chars = s.split('');
        let countLeft = 0, countRight = 0;
        let swap = 0, imbalance = 0;
        for (let i = 0; i < chars.length; i++)</pre>
            if (chars[i] == '[')
                countLeft++;
                if (imbalance > 0)
                    // number imbalanced brackets
                    swap += imbalance;
```



```
imbalance--;
            else if (chars[i] == ']')
                countRight++;
                imbalance = (countRight - countLeft);
        return swap;
     let s = "[]][][";
   document.write(swapCount(s) + "</br>");
   s = "[[][]]";
   document.write(swapCount(s));
</script>
```

#### **Output:**

2

0



This article is contributed by **Aditya Kamath**. If you like GeeksforGeeks and would like to contribute, you can also write an article using <u>write.geeksforgeeks.org</u> or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

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 industry experts, please refer **Geeks Classes Live** 





Next >

**Fitting Shelves Problem** 



#### RECOMMENDED ARTICLES

Page: 1 2 3

Find index of closing bracket for a given opening bracket in an expression 23, Nov 17

Minimum Cost required to generate a balanced Bracket Sequence
21, Aug 20

Check if the bracket sequence can be balanced with at most one change in the position of a bracket

- Minimum number of bracket reversals needed to make an expression balanced 27, Oct 15
- Find numbers of balancing positions in string
- Number of closing brackets needed to complete a regular bracket sequence 15, Nov 18
- Minimum number of bracket reversals needed to make an expression balanced | Set 2
- Print Bracket Number 26, Feb 18

# C

#### **Article Contributed By:**



#### Vote for difficulty

Current difficulty: Medium

Easy Normal Medium Hard Expert

17, Sep 19

Improved By: JANMEJAYA DAS, shrikanth13, PrateekGupta10, sanchayang98, avanitrachhadiya2155,

divyeshrabadiya07, divyesh072019, rag2127, suresh07, susmitakundugoaldanga

Article Tags: Greedy, Strings

Practice Tags: Strings, Greedy

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













Company	Learn	Practice	Contribute
About Us	Algorithms	Courses	Write an Article
Careers	Data Structures	Company-wise	Write Interview Experience
Privacy Policy	Languages	Topic-wise	Internships
Contact Us Copyright Policy	CS Subjects Video Tutorials	How to begin?	Videos

@geeksforgeeks , Some rights reserved

