Find minimum number of merge operations to make an array palindrome

Given an array of positive integers. We need to make the given array a 'Palindrome'. Only allowed operation on array is merge. Merging two adjacent elements means replacing them with their sum. The task is to find minimum number of merge operations required to make given array a 'Palindrome'.

To make an array a palindromic we can simply apply merging operations n-1 times where n is the size of array (Note a single element array is alway palindrome similar to single character string). In that case, size of array will be reduced to 1. But in this problem we are asked to do it in minimum number of operations.

Example:

```
Input : arr[] = {15, 4, 15}
Output : 0
Array is already a palindrome. So we
do not need any merge operation.

Input : arr[] = {1, 4, 5, 1}
Output : 1
We can make given array palindrome with
minimum one merging (merging 4 and 5 to
make 9)

Input : arr[] = {11, 14, 15, 99}
Output : 3
We need to merge all elements to make
a palindrome.
```

Expected time complexity is O(n).

We strongly recommend that you click here and practice it, before moving on to the solution.

Let f(i, j) be minimum merging operations to make subarray arr[i..j] a palindrome. If i == j answer is 0. We start i from 0 and j from n-1.

1. If arr[i] == arr[j], then there is no need to do any merging operations at index i or index j. Our answer in this case will be f(i+1, j-1).
2. Else, we need to do merging operations. Following cases arise.
   - If arr[i] > arr[j], then we should do merging operation at index j. We merge index j-1 and j, and update arr[j-1] = arr[j-1] + arr[j]. Our answer in this case will be 1 + f(i, j-1).
   - For the case when arr[i] < arr[j], update arr[i+1] = arr[i+1] + arr[i]. Our answer in this case will be 1 + f(i+1, j).
3. Our answer will be f(0, n-1), where n is size of array arr[].

Therefore this problem can be solved iteratively using two pointers (first pointer pointing to start of the array and second pointer pointing to last element of the array) method and keeping count of total merging operations done till now.

Below is C++ implementation of above idea.

```
// C++ program to find number of operations
// to make an array palindrome
#include <bits/stdc++.h>
using namespace std;

// Returns minimum number of count operations
// required to make arr[] palindrome
int findMinOps(int arr[], int n)
{
   int ans = 0; // Initialize result

   // Start from two corners
   for (int i=0,j=n-1; i<=j;)
   {
      // If corner elements are same,
      // problem reduces arr[i+1..j-1]
      if (arr[i] == arr[j])
      {
         i++;
         j--;
      }

      // If left element is greater, then
      // we merge right two elements
      else if (arr[i] > arr[j])
      {
         // need to merge from tail.
         j--;
         arr[j] += arr[j+1] ;
         ans++;
      }

      // Else we merge left two elements
      else
      {
         i++;
         arr[i] += arr[i-1];
         ans++;
      }
   }

   return ans;
}

// Driver program to test above
int main()
{
   int arr[] = {1, 4, 5, 9, 1};
   int n = sizeof(arr)/sizeof(arr[0]);
   cout << "Count of minimum operations is "
        << findMinOps(arr, n) << endl;
   return 0;
}
```

Output :

```
Count of minimum operations is 1
```

Time complexity for the given program is : O(n)