GeeksforGeeks

A computer science portal for geeks

Practice IDE Q&A GeeksQuiz

Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed

Given an array, only rotation operation is allowed on array. We can rotate the array as many times as we want. Return the maximum possbile of summation of i*arr[i].

Example:

```
Input: arr[] = {1, 20, 2, 10}
Output: 72
We can 72 by rotating array twice.
{2, 10, 1, 20}
20*3 + 1*2 + 10*1 + 2*0 = 72

Input: arr[] = {10, 1, 2, 3, 4, 5, 6, 7, 8, 9};
Output: 330
We can 330 by rotating array 9 times.
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
0*1 + 1*2 + 2*3 ... 9*10 = 330
```

We strongly recommend you to minimize your browser and try this yourself first.

A **Simple Solution** is to find all rotations one by one, check sum of every rotation and return the maximum sum. Time complexity of this solution is $O(n^2)$.

We can solve this problem in O(n) time using an **Efficient Solution**.
Let R_j be value of i*arr[i] with j rotations. The idea is to calculate next rotation value from previous rotation, i.e., calculate R_j from R_{j-1}. We can calculate initial value of result as R_0, then keep calculating next rotation values.

How to efficiently calculate R_j from R_{j-1}?
This can be done in O(1) time. Below are details.

```
Let us calculate initial value of i*arr[i] with no rotation
R₀ = 0*arr[0] + 1*arr[1] +...+ (n-1)*arr[n-1]

After 1 rotation arr[n-1], becomes first element of array,
arr[0] becomes second element, arr[1] becomes third element
```

```
and so on.
$R_1$ = 0*arr[n-1] + 1*arr[0] +...+ (n-1)*arr[n-2]

$R_1$ - $R_0$ = arr[0] + arr[1] + ... + arr[n-2] - (n-1)*arr[n-1]

After 2 rotations arr[n-2], becomes first element of array,
arr[n-1] becomes second element, arr[0] becomes third element
and so on.
$R_2$ = 0*arr[n-2] + 1*arr[n-1] +...+ (n?1)*arr[n-3]

$R_2$ - $R_1$ = arr[0] + arr[1] + ... + arr[n-3] - (n-1)*arr[n-2] + arr[n-1]

If we take a closer look at above values, we can observe
below pattern

$R_j$ - $R_{j-1}$ = arrSum - n * arr[n-j]

Where arrSum is sum of all array elements, i.e.,

arrSum = ∑ arr[i]
        i<=0<=n-1
```

Below is complete algorithm:

```
1) Compute sum of all array elements. Let this sum be 'arrSum'.

2) Compute $R_0$ by doing i*arr[i] for given array.
   Let this value be currVal.

3) Initialize result: maxVal = currVal // maxVal is result.

// This loop computes $R_j$ from  $R_{j-1}$
4) Do following for j = 1 to n-1
......a) currVal = currVal + arrSum-n*arr[n-j];
......b) If (currVal > maxVal)
            maxVal = currVal

5) Return maxVal
```

Below are C++ and Python implementations of above idea.

C++

```
// C++ program to find max value of i*arr[i]
#include <iostream>
using namespace std;

// Returns max possible value of i*arr[i]
int maxSum(int arr[], int n)
{
    // Find array sum and i*arr[i] with no rotation
    int arrSum = 0;  // Stores sum of arr[i]
    int currVal = 0;  // Stores sum of i*arr[i]
    for (int i=0; i<n; i++)
    {
        arrSum = arrSum + arr[i];
        currVal = currVal+(i*arr[i]);
    }

    // Initialize result as 0 rotation sum
    int maxVal = currVal;

    // Try all rotations one by one and find
    // the maximum rotation sum.
    for (int j=1; j<n; j++)
    {
        currVal = currVal + arrSum-n*arr[n-j];
        if (currVal > maxVal)
            maxVal = currVal;
    }

    // Return result
    return maxVal;
}

// Driver program
int main(void)
{
    int arr[] = {10, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "\nMax sum is " << maxSum(arr, n);
    return 0;
}
```

Run on IDE

Python

```
'''Python program to find maximum value of Sum(i*arr[i])'''

# returns max possible value of Sum(i*arr[i])
def maxSum(arr):

    # stores sum of arr[i]
    arrSum = 0

    # stores sum of i*arr[i]
    currVal = 0

    n = len(arr)

    for i in range(0, n):
        arrSum = arrSum + arr[i]
        currVal = currVal + (i*arr[i])

    # initialize result
    maxVal = currVal

    # try all rotations one by one and find the maximum
    # rotation sum
    for j in range(1, n):
        currVal = currVal + arrSum-n*arr[n-j]
        if currVal > maxVal:
            maxVal = currVal

    # return result
    return maxVal

# test maxsum(arr) function
arr = [10, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print "Max sum is: ", maxSum(arr)
```

Run on IDE

Output:

```
Max sum is 330
```

Time Complexity: O(n)

Auxiliary Space: O(1)

This article is contributed by **Nitesh Singh**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

23 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1’s is maximized

- Reorder an array according to given indexes
- Find maximum average subarray of k length
- Convert array into Zig-Zag fashion

(Login to Rate and Mark)

3 Average Difficulty : **3/5.0**
Based on **4** vote(s)

□ Add to TODO List
□ Mark as DONE

Like Share 22 people like this. Be the first of your friends.

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