# Max sum in the configuration

Given an array(0-based indexing), you have to find the max sum of i*A[i] where A[i] is the element at index i in the array. The only operation allowed is to rotate(clock-wise or counter clock-wise) the array any number of times.

**Example 1:**

```
Input: N = 4
A[] = {8,3,1,2}
Output: 29 Explanation: Above the configuration
possible by rotating elements are
3 1 2 8 here sum is 3*0+1*1+2*2+8*3 = 29
1 2 8 3 here sum is 1*0+2*1+8*2+3*3 = 27
2 8 3 1 here sum is 2*0+8*1+3*2+1*3 = 17
8 3 1 2 here sum is 8*0+3*1+1*2+2*3 = 11
Here the max sum is 29
```

**Your Task:**
Your task is to complete the function **max_sum** which takes two arguments which is the array A [ ] and its size and returns an integer value denoting the required max sum.

**Expected Time Complexity:** O(N).
**Expected Auxiliary Space:** O(1).

**Constraints:**

$1<=N<=10^4$
1<=A[]<1000

next_val = curr_val - (cum_sum - arr[i-1]) + arr[i-1] * (n-1);

next_val = Value of ∑i*arr[i] after one rotation.*

- 

curr_val = Current value of ∑i*arr[i]

cum_sum = Sum of all array elements, i.e., ∑arr[i].

Lets take example {1, 2, 3}. Current value is 1*0+2*1+3*2
= 8. Shifting it by one will make it {2, 3, 1} and next value
will be 8 - (6 - 1) + 1*2 = 5 which is same as 2*0 + 3*1 + 1*2

```python
def max_sum(a,n):
    #code here
    arrSum = sum(a)
    currSum = 0
    for j in range(n):
        currSum+=j*a[j]
    maxSum = 0
    for i in range(n):
        if i==0:
            maxSum = max(maxSum,currSum)
        else:
            currSum = currSum -arrSum+n*a[i-1]
            maxSum = max(maxSum,currSum)
    return maxSum
```