## 396. Rotate Function

Medium 239 70 Favorite Share

Given an array of integers `A` and let *n* to be its length.

Assume $B_k$ to be an array obtained by rotating the array `A` *k* positions clock-wise, we define a "rotation function" `F` on `A` as follow:

$F(k) = 0 * B_k[0] + 1 * B_k[1] + ... + (n-1) * B_k[n-1]$.

Calculate the maximum value of `F(0), F(1), ..., F(n-1)`.

**Note:**
*n* is guaranteed to be less than $10^5$.

**Example:**

```
A = [4, 3, 2, 6]

F(0) = (0 * 4) + (1 * 3) + (2 * 2) + (3 * 6) = 0 + 3 + 4 + 18 = 25
F(1) = (0 * 6) + (1 * 4) + (2 * 3) + (3 * 2) = 0 + 4 + 6 + 6 = 16
F(2) = (0 * 2) + (1 * 6) + (2 * 4) + (3 * 3) = 0 + 6 + 8 + 9 = 23
F(3) = (0 * 3) + (1 * 2) + (2 * 6) + (3 * 4) = 0 + 2 + 12 + 12 = 26

So the maximum value of F(0), F(1), F(2), F(3) is F(3) = 26.
```

```java
package Algorithm;

public class L396 {

    public int maxRotatrFunction(int [] A) {
        if (A == null || A.length == 0) {
            return 0;
        }
        if(A.length == 1)
            return 0;

        int max = Integer.MIN_VALUE;
        //找规律
        for(int i = 0; i < A.length; i ++) {
            int sum = 0;
            for(int j = A.length - i, k = 0; j < 2 * A.length - i; j ++, k ++) {
                sum += A[j % A.length] * k;
            }
            System.out.println(sum);
            if(sum > max)
                max = sum;
        }
        return max;
    }

    public static void main(String[] args) {
        int [] A = new int [] {4, 3, 2, 6};
        System.out.println(new L396().maxRotatrFunction(A));
    }
}
```