// https://leetcode.com/problems/sum-of-absolute-differences-in-a-sorted-array/submissions/

In the problem we are asked, given an already sorted array *A*, to find an array containing the values, for all indices, of the value corresponding to the sum of the absolute values of the differences between each element and all other array elements. A brute force approach, having a time complexity of *O(N^2)*, would involve two loops, where in the outer we loop for all array elements, and in the inner one we accumulate the sum of the absolute differences between the given array element and all other array elements. This approach is certainly not optimum. In fact, if *N* is the size of the array, *i* the index of the array element for which we want to calculate the sum, mathematically the equation can be expressed as follows:

$$y = \sum_{j=0}^{j=N-1} |A[i] - A[j]| \qquad [1]$$

Equation [1] can be further simplified if we exploit the fact that the array is already sorted: if we are at element with index *i*, looking to the left (for all indices < *i*) all elements will be smaller than the given element, whereas, looking to the right (for all indices > *i*), all elements will be greater. Hence, we can rewrite the above equation as follows:

$$y = \sum_{j=0}^{j=i-1} (A[i] - A[j]) + \sum_{j=i+1}^{j=N-1} (A[j] - A[i]) \qquad [2]$$

Equation [2] can be further simplified, given that the element at the given index *i* can be taken out of the summation:

$$y = i \times A[i] - \sum_{j=0}^{j=i-1} (A[j]) + \sum_{j=i+1}^{j=N-1} (A[j]) - (N - 1 - i) \times A[i] \qquad [3]$$

If the prefix sum of the given array A is pre-computed (with linear time complexity, i.e. O(N)), equation [3] can then subsequently be computed scanning the array only once (linear time complexity, i.e. O(N)). The devised algorithm, written below in C++, will have an overall linear time and space complexity (because we need to allocate an array, where we store the values of the prefix sum).

```
class Solution {
public:
  vector<int> getSumAbsoluteDifferences(vector<int>& nums) {
    const int l=static_cast<int>(std::size(nums));
    if(!l)return{};
    if(l==1)return{0};
    vector<int> px(l+1,0);
    int i{0};
    while(i<l){
      px[i+1]=px[i]+nums[i];
      ++i;
    }
    vector<int> sol(l,0);
```

```
        i=0;
        while(i<l){
            int left{i},right{l-1-i};
            int sumleft{left*nums[i]-px[i]};
            int sumright{px[l]-px[i+1]-right*nums[i]};
            sol[i]=sumleft+sumright;
            ++i;
        }
        return sol;
    }
};
```