1769. Minimum Number of Operations to Move All Balls to Each Box

You have `n` boxes. You are given a binary string `boxes` of length `n`, where `boxes[i]` is `'0'` if the `i`th box is **empty**, and `'1'` if it contains **one** ball.

In one operation, you can move **one** ball from a box to an adjacent box. Box `i` is adjacent to box `j` if `abs(i - j) == 1`. Note that after doing so, there may be more than one ball in some boxes.

Return an array `answer` of size `n`, where `answer[i]` is the **minimum** number of operations needed to move all the balls to the `i`th box.

Each `answer[i]` is calculated considering the **initial** state of the boxes.

Example 1:

```
Input: boxes = "110"
Output: [1,1,3]
Explanation: The answer for each box is as follows:
1) First box: you will have to move one ball from the second box to the first box in one operation.
2) Second box: you will have to move one ball from the first box to the second box in one operation.
3) Third box: you will have to move one ball from the first box to the third box in two operations, and move one ball from the second box to the third box in one operation.
```

Example 2:

```
Input: boxes = "001011"
Output: [11,8,5,4,3,4]
```

Constraints:

- `n == boxes.length`
- `1 <= n <= 2000`
- `boxes[i]` is either `'0'` or `'1'`.

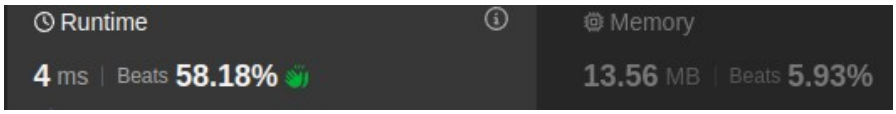
1769. Minimum Number of Operations to Move All Balls to Each Box

```
/*
   Brute force
   Time compelxity: O(n^2)
   Space complexity: O(1)
*/
```


```
typedef std::vector<int> vi;
class Solution {
   public:
      vi minOperations(std::string boxes){
         int n=boxes.size();
         vi ans;
         // For each box
         for(int i=0;i<n;++i){
            // Compute the number of steps to get the 1 at position j
            int cnt=0;
            for(int j=0;j<n;++j){
               if(boxes[j]=='1') cnt+=abs(i-j);
            }
            // Add the result to the answer
            ans.push_back(cnt);
         }
         return ans;
      }
};
```

1769. Minimum Number of Operations to Move All Balls to Each Box

/*
Prefix/Suffix sum
Time complexity: O(7n)
Space complexity: O(4n)
*/




```
typedef std::vector<int> vi;

class Solution {
   public:
      vi minOperations(std::string boxes){
         int n=boxes.size();

         // For each box:

         // Preprocess #moves from left
         vi balls_in_left(n,0);
         balls_in_left[0]=int(boxes[0]=='1');
         vi moves_from_left(n,0);
         for(int i=1;i<n;++i){
            balls_in_left[i]=balls_in_left[i-1]+int(boxes[i]=='1');
            moves_from_left[i]=moves_from_left[i-1]+balls_in_left[i-1];
         }

         // Preprocess #moves from the right
         vi balls_in_right(n,0);
         balls_in_right[n-1]=int(boxes[n-1]=='1');
         vi moves_from_right(n,0);
         for(int i=n-2;i>=0;--i){
            balls_in_right[i]=balls_in_right[i+1]+int(boxes[i]=='1');
            moves_from_right[i]=moves_from_right[i+1]+balls_in_right[i+1];
         }

         vi ans;
         for(int i=0;i<n;++i){
            ans.push_back(moves_from_left[i]+moves_from_right[i]);
         }
         return ans;
      }
};
```

1769. Minimum Number of Operations to Move All Balls to Each Box

```
/*
   Prefix/Suffix sum: One passe and space optimization
   Time complexity: O(n)
   Space complexity: O(1)
*/

typedef std::vector<int> vi;

class Solution {
   public:
      vi minOperations(std::string boxes){
         int n=boxes.size();

         vi ans(n,0);
         int balls_in_left=0,moves_from_left=0;
         int balls_in_right=0,moves_from_right=0;
         for(int i=0,j=n-1;i<n && j>=0;++i,--j){
            ans[i]+=moves_from_left;
            balls_in_left+=int(boxes[i]=='1');
            moves_from_left+=balls_in_left;


            ans[j]+=moves_from_right;
            balls_in_right+=int(boxes[j]=='1');
            moves_from_right+=balls_in_right;
         }
         return ans;
      }
};
```