6.01.2025

Problem Description

You have `n` boxes. You are given a binary string `boxes` of length `n`, where `boxes[i]` is `'0'` if the `ith` 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 `ith` 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]
```

2 Approaches

In first approach, I created a `set` data structure which stores the indices in the sorted manner.

After that, I traversed through the set to get the indices of ‘1’ in the given string so that I can find the absolute difference between each index in string and indices

the summation of each difference between i and all indices at which 1 is present would be stored in resultant vector. Hence, the result.

time complexity : O(N^2)

space complexity: O(2 * N)

---

the O(n^2) solution would be accepted by leetcode as the constraints are low but for learning I have found a new approach, i.e. left right summation or left right prefix sum.

here, first we traverse through the array left to right and do the required operations and moving from right to left and do the required operations.

code:

```
 class Solution {
public:
    vector<int> minOperations(string s) {
      int n = s.length();
      int count = 0;
      int operations = 0;
      vector<int>res(n,0);
      // left to right ja rhe hai
      for(int i = 0; i < n; i++){
        res[i] =  operations;
        if(s[i] == '1'){
            count = count + 1;
        }
        operations = operations + count;
      }

      count = 0, operations =0;
      // right to left ja rhe hai
      for(int i = n-1; i>= 0; i--){
        res[i] += operations;
        if(s[i] == '1'){
            count = count + 1;
        }
        operations = operations + count;
      }
      return res;
    //   TC: O(n), SC: O(N)
    }
};

// O(n^2) Tc AND sC : o(2n)
//   set<int>storeOneIndices;
//         for(int i = 0 ; i < boxes.length(); i++){
//             if(boxes[i] == '1'){
//                 storeOneIndices.insert(i);
//             }
//         }

//         vector<int>res(boxes.length(), 0);
//         for(int i = 0; i < boxes.length(); i++){
//             int sum = 0;
//             for(auto& it: storeOneIndices){
//                 sum = sum + abs(i - it);
//             }
//              res[i] = res[i] + sum;
//          }
//          return res;
```