如果一个由 `'0'` 和 `'1'` 组成的字符串，是以一些 `'0'`（可能没有 `'0'`）后面跟着一些 `'1'`（也可能没有 `'1'`）的形式组成的，那么该字符串是*单调递增*的。

我们给出一个由字符 `'0'` 和 `'1'` 组成的字符串 `S`，我们可以将任何 `'0'` 翻转为 `'1'` 或者将 `'1'` 翻转为 `'0'`。

返回使 `S` 单调递增的最小翻转次数。

示例 1：

输入： `"00110"`
输出： `1`
解释： 我们翻转最后一位得到 `00111.`

示例 2：

输入： `"010110"`
输出： `2`
解释： 我们翻转得到 `011111`，或者是 `000111`。

示例 3：

输入： `"00011000"`
输出： `2`
解释： 我们翻转得到 `00000000`。

提示：

1. `1 <= S.length <= 20000`
2. `S` 中只包含字符 `'0'` 和 `'1'`

思路：把字符串全部置为同一个元素，然后从头往尾遍历，若是该节点为分界点，消耗多少，这个消耗可以由前一位的消耗得出，为前一位消耗+当前位改变的消耗。

代码

```
class Solution {
public:
    int minFlipsMonoIncr(string S) {
        int num=0;
        int min=1;
        for(int i=0;i<S.size();i++){
            if(S[i]=='0'){
                num++;
```

```
            }
        }
        min=num;
        for(int i=0;i<S.size();i++){
            if(S[i]=='1'){
                num++;
            }else{
                num--;
            }
            if(min>num){
                min=num;
            }
        }
        return min;
    }
};
```