AcWing-4114：垃圾桶

题目描述

难度：	简单
时/空限制：	1s / 64MB
总通过数：	862
总尝试数：	3247
来源：	Google Kickstart2021 Round F Problem A
算法标签	

一条很长的街道上有 N 个房子。

第一个房子在位置
1，第二个房子在位置
2，以此类推。

任意一对房子
i 和
j 之间的距离为
$|i-j|$。

一些房子的位置处有垃圾桶。

每个房子的主人都要倒垃圾。

如果自己房子前面有垃圾桶，则无需移动，直接倒垃圾即可。

如果自己房子前面没有垃圾桶，则前往距离自己最近的垃圾桶处倒垃圾，如果这样的垃圾桶不唯一，则任意前往一个即可。

请计算，所有房子的主人倒垃圾需要行走的总距离之和。

输入格式

第一行包含整数
T，表示共有
T 组测试数据。

每组数据第一行包含整数 N。

第二行包含一个长度为
N 的
01 字符串，第
i 个字符如果为
1，则表示第
i 个房屋门前有垃圾桶，如果为
0，则表示第
i 个房屋门前没有垃圾桶。

输出格式

每组数据输出一个结果，每个结果占一行。

结果表示为 `Case #x: y`，其中
x 为组别编号（从
1 开始），
y 为所有房子的主人倒垃圾需要行走的总距离之和。

数据范围

$1 \le T \le 100$,
$1 \le N \le 5 \times 10^5$,
字符串中至少包含一个 1。

输入样例：

```
2
3
111
6
100100
```

输出样例：

```
Case #1: 0
Case #2: 5
```

样例解释

对于 Case 1，每个房子门前都有垃圾桶，所以大家都不用移动。

对于 Case 2，第 1,4 个房子的门前有垃圾桶，这两家主人不用移动，第 2 个房子的主人去第 1 个房子处倒垃圾，第 3,5,6 个房子的主人去第 2 个房子处倒垃圾。

算法求解

最直接能想到的方法就是直接遍历，如果当前位置没有垃圾桶，就往两端查找。该算法最坏情况下时间复杂度为`O(n^2)`，导致超时。

改进策略为牺牲空间换取时间，用两个数组分别存储每一位左边和右边距离最近的垃圾桶所在处。

关于这两个数组的求解，都可以通过`dp`的方式在`O(n)`时间复杂度内求解：如果该地不是垃圾桶，那么“左边距离最近的垃圾桶”的位置一定和其左边一位“左边最近垃圾桶”的值相同，即：

```
for(int i=1;i<=N;i++){
      lefts[i]=(s[i]=='1'?i:lefts[i-1]);
}
```

右边同理，不过是从右往左遍历。

求出两个数组后再花`O(n)`的时间遍历整条街，取每个位置左右距离最小值即可。注意结果需要使用`long long int`存储。

Code：

```
//
//  main.cpp
//  4114-垃圾桶
//
//  Created by MacBook Pro on 2023/8/12.
//
```

```cpp
#include <iostream>
#include <math.h>
#include <climits>
using namespace std;

// O(n^2) 时间复杂度过高
//int main() {
//    int T;
//    scanf("%d",&T);
//    for(int cases=1;cases<=T;cases++){
//        int N;
//        scanf("%d",&N);
//        char s[500005];
//        scanf("%s",s);
//        long long ans = 0;
//        for(int i=0;i<N;i++){
//            if(s[i]=='0'){
//                //寻找垃圾桶
//                int offset = 1;
//                while(true){
//                    if((i+offset<N && s[i+offset]=='1')||(i-offset>=0 && s[i-
offset]=='1')) break;
//                    else offset++;
//                }
//                ans += offset;
//            }
//        }
//        printf("Case #%d: %lld\n",cases,ans);
//    }
//}

char s[500005];
int lefts[500005],rights[500005]; //分别存储左右最近

// O(n) 存储法
int main() {
    int T;
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        int N;
        scanf("%d",&N);
        scanf("%s",s+1);
        long long ans = 0;
        //dp思想
        lefts[0] = -1e6; rights[N + 1] = 1e6;
        for(int i=1;i<=N;i++){
            lefts[i]=(s[i]=='1'?i:lefts[i-1]);
        }
        for(int i=N;i>=1;i--){
```

```
            rights[i]=(s[i]=='1'?i:rights[i+1]);
        }
        //O(N)求结果
        for(int i=1;i<=N;i++){
            ans += min(i-lefts[i],rights[i]-i);
        }
        printf("Case #%d: %lld\n",cases,ans);
    }
}
```