AcWing-3748：递增子串

题目描述

难度：	简单
时/空限制：	2s / 64MB
总通过数：	868
总尝试数：	1476
来源：	Google Kickstart2021 Round B Problem A
算法标签	

你的朋友约翰刚刚度假归来，他迫不及待地想要跟你分享他了解到的关于字符串的一个新性质。

他了解到，如果一个长度为 L 的大写字母构成的字符串 C，满足对于每对索引 i,j（$1 \le i < j \le L$，索引编号 $1 \sim L$），位置 i 处的字符均小于位置 j 处的字符，则该字符串是严格递增的。

例如，字符串 `ABC` 和 `ADF` 是严格递增的，而字符串 `ACC` 和 `FDA` 则不是。

在教给你这个关于字符串的新性质后，他打算考一考你：

给定一个长度为 N 的字符串 S，请你计算对于每个位置 i（$1 \le i \le N$），以该位置结束的最长严格递增子串的长度是多少？

输入格式

第一行包含整数 T，表示共有 T 组测试数据。

每组数据占两行，第一行包含整数 N，第二行包含一个长度为 N 的由大写字母构成的字符串 S。

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: y1 y2 ... yn`，其中 x 为组别编号（从 1 开始），y_i 为以位置 i 结束的最长严格递增子串的长度。

数据范围

全部数据：
$1 \le T \le 100$。
测试点 1 （小数据测试点）：
$1 \le N \le 100$。
测试点 2 （大数据测试点）：
$1 \le N \le 2 \times 10^5$。

输入样例：

```
2
4
ABBC
6
ABACDA
```

输出样例：

```
Case #1: 1 2 1 2
Case #2: 1 2 1 2 3 1
```

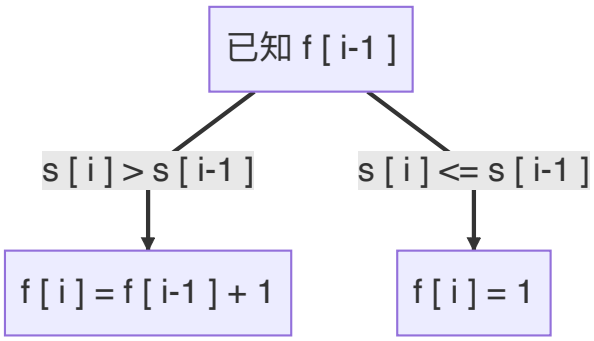
样例解释

对于测试数据 1，在位置 1、2、3 和 4 处结束的最长严格递增子串分别为 `A`、`AB`、`B`、`BC`。

对于测试数据 2，在位置 1~6 处结束的最长严格递增子串分别为 `A`、`AB`、`A`、`AC`、`ACD`、`A`。

算法求解

`dp`问题，假设字符串为`s`，使用`f[i]`来表示第`i`个字符末尾时，最长的子串长度。



以上子问题的关系构成动态规划，因此只需要遍历一遍即可，时间复杂度为`O(n)`

Code：

```
//
//  main.cpp
//  3748-递增子串
//
//  Created by MacBook Pro on 2023/8/10.
//

#include <iostream>
using namespace std;
```

```
int main() {
    int T;
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        int n;
        string s;
        cin>>n>>s;
        int bef = 0;
        printf("Case #%d:",cases);
        for(int i=0;i<n;i++){
            if(i>0 && s[i]>s[i-1]){
                printf(" %d",++bef);
            }
            else{
                printf(" %d",1);
                bef = 1;
            }

        }
        printf("\n");
    }
}
```