Pattern in Strings - Hackerearth

Saturday, February 29, 2020 11:06 AM

The Problem:

You are given a string which only consists of 'a' and 'b' in it.
You are allowed to remove any number characters from this string(possibly none).
You need to find maximum possible size of string of form a^ib^ja^k or b^ia^jb^k wherei≥0, j≥0, k≥0.
Basically (a*b*c*) or (b*a*b*)

Constraints : -
1≤T≤100
1≤n≤106

Input : -
First line contains integer T , no. of test cases.
First line of each test case contains integer n ,size of string
Second line of each test case contains contains string s of size n.

Output : -
For each test case output maximum possible size of string of given form in new line.

SAMPLE INPUT

```
3
18
abaababbbaaaabaaab
3
bab
9
bbbbababb
```

SAMPLE OUTPUT

```
14
3
8
```

Explanation:
In first test case, you can remove 2nd,5th,14th,18th characters to get maximum possible size of 14.
In second test case, you no need to remove anything, so maximum size is 3.
In third test case,you remove 6th character to get string of size 8.

The code:

```
/*
```

```
 * The Logic is simple. For aba or bab, I'm trying to find the maximum consecutive a's or b's in the center. There can be zero or more a's or b's on either sides.
 * Let's say there's a string: baaabaaab
 * Maximum Consecutive a's will be 'aaabaaa'. Here we gain 6 a's, but lose 1 b.
 * So let's say addedA = 6-1 = 5. As we're counting the addedA's and not total a's, we can simply add it to the 'b' count for bab pattern size.
 * Similarly for aba. Return whichever is maximum.
 * Note that when the addedA count becomes <= 0, we start the count afresh.
 * And there can be two patterns with the same addedA value, but we pick the one where we lose less b's.
 * For example: while checking for aaabaabbbaaaa, the substring 'aaabaa' will give addedA = 4. But lostB = 2.
 * But for the substring 'aaaa' addedA = 4 and lostB = 0. So we pick this.
 */


#include <stdio.h>

long int findMaxPattrnLen(char arr[], int length) {
   long int i, maxA, countA, addedA, lostA, minLostA;
   long int maxB, countB, addedB, lostB, minLostB;

   maxA = countA = addedA = lostA = 0; minLostA = 9999999;
   maxB = countB = addedB = lostB = 0; minLostB = 9999999;
   for(i=0; i<length; i++) {

      if(arr[i] == 'a'){
         countA++;
         addedA++;
         addedB--;

         lostA++;
      } else {
         countB++;
         addedA--;
         addedB++;

         lostB++;
      }

      if(addedA <= 0){
         addedA = 0;
         lostB = 0;
      } else if((addedA > maxA) || (addedA == maxA && lostB < minLostB)) {
         maxA = addedA;
         minLostB = lostB;
      }

      if(addedB <= 0){
         addedB = 0;
         lostA = 0;
      } else if((addedB > maxB) || (addedB == maxB && lostA < minLostA)) {
         maxB = addedB;
```

```
            minLostA = lostA;
        }
    }

    maxA += countB;
    maxB += countA;

    return maxA > maxB ? maxA : maxB;
}



int main() {
        int caseCount, len;
        scanf("%d", &caseCount);

        char pattern[1000000];
        long int result;

        while(caseCount > 0) {

            scanf("%d %s", &len, &pattern);

            result = findMaxPattrnLen(pattern, len);

            printf("%ld\n", result);

            caseCount--;
        }
}
```

The Stats:

Score
30.0

Time (sec)
1.6678

Memory (KiB)
1168

Language
C