Substring Diff

 **Editorial by kapilsingh93**

As the maximum size of the strings can be 1500 , this gives a little hint that a O(n^2) approach can be acceptable.

So we first keep the differences between the two strings in an difference[1500][1500] array, where difference[i][j] represents the whether the ith element of string1 is different from jth element of string2.

As it's not necessary that i and j are same therefore the two resultant substrings may end up at two different points in two strings.

We maintain a front_pointer that is used to calculate the ending point of the two substrings of the two given strings. Also we keep on increasing the gap between the endings of the two substrings . So at any point we consider two set of substrings.

1. substring ending at front_pointer in string1 and substring ending at front_pointer+gap in string2.
2. substring ending at front_pointer+gap in string1 and substring ending at front_pointer in string2.

We maintain the total number of different characters encountered so far in front_sum. At any point if the total different characters exceeds k, we increment the back_pointer (representing the starting of the substring) until we get less than k different characters. Total different characters at any point can be calculated by front_sum-back_sum . back_sum represents total different characters till back_pointer . Needless to mention that we keep a check on the maximum length of the substring possible at each point .

Tested by srikanth

Problem Tester's code :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define MAX_SIZE 1500
int main(void){
    int num_cases;
    int k;
    char string1[MAX_SIZE+1],string2[MAX_SIZE+1];
    char diff_array[MAX_SIZE][MAX_SIZE];
    int length;
    int i;

    scanf("%d",&num_cases);
    while(num_cases--){
        scanf("%d %s %s",&k,string1,string2);
        length=strlen(string1);


        int j;
        for(i=0;i<length;i++){
            for(j=0;j<length;j++)
                diff_array[i][j]=(string1[i]!=string2[j]);
        }
        int front_pointer,back_ptr1,back_ptr2,front_sum1,front_sum2,curr_max=-1;
        int back_sum1,back_sum2;
```

Statistics

Difficulty: 0.620689655
Publish Date: Jun 13 201

This is a Practice Challe

```
        for(i=0;i<length;i++){
            front_sum1=front_sum2=back_sum1=back_sum2=0;
            back_ptr1=back_ptr2=-1;
            for(front_pointer=0;front_pointer+i<length;front_pointer++){
                front_sum1+=diff_array[front_pointer][i+front_pointer];
                front_sum2+=diff_array[i+front_pointer][front_pointer];
                while(front_sum1-back_sum1>k){
                    back_ptr1++;
                    back_sum1+=diff_array[back_ptr1][i+back_ptr1];
                }
                while(front_sum2-back_sum2>k){
                    back_ptr2++;
                    back_sum2+=diff_array[i+back_ptr2][back_ptr2];
                }

                if(front_pointer-back_ptr1>curr_max)
                    curr_max=front_pointer-back_ptr1;
                if(front_pointer-back_ptr2>curr_max)
                    curr_max=front_pointer-back_ptr2;
            }
        }
        printf("%d\n",curr_max);
    }
    return 0;
}
```