A string is called *boring* if all the characters of the string are **same**.

You are given a string *S* of length *N*, consisting of lowercase english alphabets. Find the length of the longest *boring* substring of *S* which occurs **more than once**.

Note that if there is no *boring* substring which occurs more than once in *S*, the answer will be 00.

A substring is obtained by deleting some (possibly zero) elements from the beginning of the string and some (possibly zero) elements from the end of the string.

Input Format

- The first line of input will contain a single integer *T*, denoting the number of test cases.
- Each test case consists of two lines of input.
  - The first line of each test case contains an integer *N*, denoting the length of string *S*.
  - The next contains string *S*.

Output Format

For each test case, output on a new line, the length of the longest *boring* substring of *S* which occurs **more than once**.

Sample Input

```
4

3

aaa

3

abc

5

bcaca

6

caabaa
```

Sample Output

```
2

0

1
```

2

C Solution

```
#include <stdio.h>
int main(void) {
        int t;
        scanf("%d",&t);
        int n;
        while(t--)
        {
         scanf("%d",&n);
         char s[n];

         int first[26]={0};
         int second[26]={0};

         scanf("\n");
         for(int i=0;i<n;i++)
         {
           scanf("%c",&s[i]);


         }
         int count=1;
         for(int i=0;i<n;i++)
         {


           if((s[i]==s[i+1])&&(i!=n-1))
           {
```

```
    count++;
  }
  else
  {
   if(first[s[i]-'a']<count)
   {
     first[s[i]-'a']=count;


   }
   else if(first[s[i]-'a']==count)
   {
     second[s[i]-'a']=count;
   }
   count=1;
  }
}
int max_ind=0;
int max=-1;
for(int i=0;i<26;i++)
{
  if(max<first[i])
  {
    max=first[i];
    max_ind=i;
  }
  else if(max==first[i])
  {
    if(second[i]==first[i])
    {
```

```
              max_ind=i;
            }
          }
        }
        if(first[max_ind]==second[max_ind])
        {
          printf("%d\n",first[max_ind]);
        }
        else
        {
          printf("%d\n",first[max_ind]-1);
        }
       }
       return 0;
}
```

C++ Solution

```
#include <bits/stdc++.h>
using namespace std;
#define st unordered_set

int main() {
	int tc;
	cin>>tc;
	while(tc--){
	  int n;
	  cin>>n;
	  string str;
	  cin>>str;
```

```
string res = "";
st<string>s;
res=str[0];
int l,mx,sum;
l=mx=0;
sum=1;
for(int i=1; i<n; i++){
    if(str[i-1]!=str[i]){
        if(mx<sum){
            mx = sum;
            mx--;
        }
        if(s.find(res)!=s.end())
        l = max(l, sum);
        s.insert(res);
        res = str[i];
        sum=1;
    }
    else{
        sum++;
        res+=str[i];
    }
}
if(s.find(res)!=s.end()){
    l = max(l, sum);
}
else{
    if(mx<sum){
      mx = sum;
```

```
                    mx--;
                   }
               }
               int ans = max(l, mx);
               cout<<ans<<endl;
            }
            return 0;
}
```

Java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class Main {
   public static void main(String[] args) throws IOException {
      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
      int n = Integer.parseInt(in.readLine());
      int m = 0, currLength = 0, longest = 0;
      char lastChar;
      char[] c;
      int[] charCounter;

      while (n >0) {
         m = Integer.parseInt(in.readLine());
         c = in.readLine().trim().toCharArray();
         charCounter = new int[30];
         longest = 0;
```

```
lastChar = c[0];
currLength = 1;

for (int i = 1; i < c.length; i++) {
    if (c[i] == lastChar) {
        currLength++;
    } else {
        if (currLength >= charCounter[lastChar - 'a']) {
            if (currLength > longest) {
                if (currLength > charCounter[lastChar - 'a'] + 1) {
                    longest = currLength - 1;
                } else {
                    longest = charCounter[lastChar - 'a'];
                }
            }
            charCounter[lastChar - 'a'] = currLength;
        }
        lastChar = c[i];
        currLength = 1;
    }

    if (i == (c.length - 1) && currLength > longest) {
        if (currLength == c.length) {
            longest = currLength - 1;
        } else if (currLength >= charCounter[lastChar - 'a']) {
            if (currLength > charCounter[lastChar - 'a'] + 1) {
                longest = currLength - 1;
            } else {
              longest = charCounter[lastChar - 'a'];
            }
          }
        }
      }
      System.out.println(longest);
      n--;
    }
  }
}
```

Python

```
t=int(input())
for _ in range(t):
 d={}
 n=int(input())
 s=input()
 mx=0
 i=0
 while(i<n):
  c=1
  ss=s[i]
  while(i<n-1 and s[i]==s[i+1]):
   c+=1
   i+=1
   ss+=s[i]
  mx=max(mx,c-1)
  d[ss]=d.get(ss,0)+1
  if(d[ss]==2):
```

```
  mx=max(mx,len(ss))
 i+=1;

print(mx)
```