Daily Test Happy Coding from necse SkillRack

Time Left: 00:00:00

Palindrome Queries

The program must accept a string **S** containing only lower case alphabets and **Q queries** as the input. Each query contains two integers **L** and **R** represent the starting position and ending position of a substring in S. For each query, the program must rearrange the alphabets in the specified substring to make a palindrome. If two or more palindromes are possible, then the program must consider the lexicographically smallest palindrome. The program must ignore the query if no palindrome is possible on rearranging the alphabets. After each query, the program must print the revised string as the output.

Boundary Condition(s):
1 <= L < R <= Length of S <= 100
1 <= Q <= 100

Input Format:
The first line contains S.
The second line contains Q.
The next Q lines, each contains L and R separated by a space.

Output Format:
The first Q lines, each contains the revised string.

Example Input/Output 1:
Input:
skillrack
1
3 5

Output:
sklilrack

Explanation:
Here the given string is **skillrack** and **Q = 1**.
Query 1: L=3, **R=5** and the substring is **ill**. After rearranging the alphabets in the substring to a palindrome, the string becomes sk**lil**rack.

Example Input/Output 2:
Input:
adccbcaacb
4
2 5
6 9
4 10
1 9

Output:
adccbcaacb
adccbaccab
adcabcccba
abccdccbaa

Max Execution Time Limit: 50 millisecs

Ambiance

Java (12.0)

```java
import java.util.*;
public class Hello {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String x = sc.nextLine();

        int n =sc.nextInt();

        while(n-->0){
            int start = sc.nextInt();
            int end = sc.nextInt();

            String alt = printPalindromicString(x,start,end);

            System.out.println(x.substring(0,start-1)+alt+x.substring((x
                .length()/2)+1,x.length()));
        }

    }
    public static String printPalindromicString(String s,int st,int end){

        String subs = s.substring(st-1,end);
        String res="";

        TreeMap<Character,Integer> ht = new TreeMap<>();

        for(char i:subs.toCharArray()){
            if(ht.containsKey(i)){
                ht.put(i,ht.get(i)+1);
            }else{
                ht.put(i,1);
            }
        }

        boolean impos=false;
        for(Map.Entry<Character,Integer> e: ht.entrySet()){
            int key = e.getKey();
            int val = e.getValue();

            if(val%2!=0){
                if(impos) return "-1";
                impos=true;
            }
        }

        char singular = '-';
        for(Map.Entry<Character,Integer> e:ht.entrySet()){
            if(e.getValue()%2!=0){
                singular=e.getKey();
            }else{
                for(int i=1;i<=(e.getValue()/2);i++){
                    res+=(""+e.getKey());
                }
            }
        }

        String rev =  new StringBuilder(res).reverse().toString();

        if(singular!='-'){
            return res+(""+singular)+rev;
        }

       // System.out.println(ht);

        return res;

    }
}
```

1912067@nec

Please wait while we run the program ....