💻 Resolução de Problemas – Strings

Exercício 1 - Different Consecutive Characters

Pseudo-código

```
Início
    Ler o número de casos de teste T
    Para cada caso de teste t de 0 até T-1
        Ler o valor N (tamanho da string binária)
        Ler a string S de tamanho N
        Inicializar o contador count como 0

        Para cada índice i de 1 até N-1
            Se o caractere S[i] for igual ao caractere S[i-1]
                Incrementar o contador count
        Imprimir o valor de count (número de ocorrências de caracteres consecutivos iguais na string)
    Fim
```

Solução em Java

```
import java.util.*;
public class DifferentConsecutiveCharacters {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        //Numero de casos de teste
        int T = sc.nextInt();
```

```
//Laço para processamento de cada caso de teste
        for (int t = 0; t < T; t++) {
            int N = sc.nextInt(); //Lê o tamanho da string binaria
            String S = sc.next(); //Lê a string de tamanho N

            int count = 0; //Inicializa o contador para o número de
ocorrências de caracteres consecutivos iguais

            //Percorre cada caractere da string e verifica caracteres iguais
consecutivos
            for (int i = 1; i < N; i++) { //Verifica a partir do segundo
caractere
                if (S.charAt(i) == S.charAt(i - 1)) {
                    count++; //Se o caractere atual for igual ao anterior,
incrementa o contador
                }
            }
            System.out.println(count); //Saída (número de ocorrências de
caracteres consecutivos iguais na string)
        }
        sc.close();
    }
}
```

Exercício 2 - Apaxiaaaaaaaaaaans!

Pseudo-código

```
Início
    Leia nome  // Recebe o nome do usuário
    Converta nome para minúsculas

    Se comprimento do nome < 1 ou comprimento do nome > 250 então
        Retorne  // O nome não é válido

    Crie uma variável nomeCompacto como uma lista ou string vazia
```

```
  Adicione o primeiro caractere de nome ao nomeCompacto


    Para i de 1 até comprimento de nome - 1 faça:
        Se nome[i] for diferente de nome[i-1] então
            Adicione nome[i] a nomeCompacto


    Escreva nomeCompacto  // Exibe o nome compactado


Fim
```

Solução em Java

```
import java.util.*;
public class Apaxians {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String nome = sc.next().toLowerCase(); //Converte o nome para
caracteres minúsculos


        //O nome deve ter no mínimo 1 caractere e no máximo 250
        if(nome.length() < 1 || nome.length() > 250){
            return;
        }
        // Usa StringBuilder para construir a versão compactada do nome
        StringBuilder nomeCompacto = new StringBuilder();


        //Adiciona o primeiro caractere ao nome compacto
        nomeCompacto.append(nome.charAt(0));


        //Percorre cada caractere do nome a partir da segunda posição
        for(int i = 1; i < nome.length(); i++){
            //Se o caractere atual for diferente do anterior, adiciona ao
nome compactado
            if(nome.charAt(i) != nome.charAt(i-1)){
                nomeCompacto.append(nome.charAt(i));
            }
        }
        System.out.println(nomeCompacto.toString());
        sc.close();
    }
}
```

Exercício 3 – Normal Problem

Pseudo-código

```
Início
    Função validaString(a)
        Para cada caractere c em a faça:
            Se c não for 'p', 'q' ou 'w' então
                Retorne Falso
        Retorne Verdadeiro

    Função retornaString(a)
        Se validaString(a) for Verdadeira então
            Criar novaString vazia
            Para cada caractere c em a faça:
                Se c for 'q' então
                    Adicione 'p' a novaString
                Se c for 'p' então
                    Adicione 'q' a novaString
                Se c for 'w' então
                    Adicione 'w' a novaString
            Inverter novaString
            Retorne novaString

    Leia t  // Número de casos de teste
    Se t for válido (1 <= t <= 100) então
        Para i de 1 até t faça:
            Leia a  // String de entrada
            b = retornaString(a)  // Processa a string
            Escreva b  // Exibe a string processada
Fim
```

Solução em Java

```
//Problema Normal Problem

import java.util.*;

public class NormalProblem {

    //FUNÇÕES

    public static boolean validaString(String a) {

        for (int i = 0; i < a.length(); i++) {

            char c = a.charAt(i);

            if (c != 'p' && c != 'q' && c != 'w') {

                return false; // Retorna falso se encontrar um caractere
inválido

            }

        }

        return true; // Retorna verdadeiro se todos os caracteres forem
válidos (p, q, w)

    }

    public static String retornaString(String a) {

        StringBuilder novaString = new StringBuilder(); //permite criar e
manipular dados de Strings

        //Utiliza a função para validar a string e realiza a troca dos
caracteres

        if(validaString(a)) {

            for(int i = 0; i < a.length(); i++) {

                char c = a.charAt(i);

                if(c == 'q') {

                    novaString.append('p');

                }else if(c == 'p') {

                    novaString.append('q');

                }else if(c == 'w'){

                    novaString.append(c); //Se for w continuara w

                }

            }

        }

        return novaString.reverse().toString(); //reverte e imprime a
string

    }
```

```
public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int t = scanner.nextInt();

        scanner.nextLine(); // Consome o caractere de quebra de linha após
o número


        //Valida se t é um número maior ou igual a 1 e menor ou igual a
100

        if(t >= 1 && t <= 100) {


            for(int i = 0; i < t; i++) {

                String a = scanner.nextLine();

                String b = retornaString(a);

                System.out.println(b); //imprime a string b que Ship vê
dentro da loja

            }

        }

        scanner.close();

    }

}
```

Exercício 4 – Aaah!

Pseudo-código

```
Início

    Função validaString(x)

        Contador countA <- 0

        Para cada caractere c em x faça:

            Se c for 'a' então

                countA += 1

            Se c não for 'a' e não for 'h' então

                Retorne Falso

        Fim Para

        Se x não terminar com 'h' ou countA não estiver entre 0 e 999 então

            Retorne Falso

        Retorne Verdadeiro
```

```
Função compara_aah(string1, string2)
        Contador count1 <- 0, count2 <- 0
        Para cada 'a' em string1 faça count1 += 1
        Para cada 'a' em string2 faça count2 += 1
        Se count1 for maior ou igual a count2 então
            Escreva "go"
        Senão
            Escreva "no"
    Leia aah_Jon
    Leia aah_Medico
    Se validaString(aah_Jon) e validaString(aah_Medico) então
        Chame compara_aah(aah_Jon, aah_Medico)
    Senão
        Escreva "Entrada inválida."
Fim
```

Solução em java

```
import java.util.*;
public class aah {
    //FUNÇÕES
    //Função que valida as restrições da string
    public static boolean validaString(String x) {
        // Conta o número de a's
        int countA = 0;
        for(int i = 0; i < x.length(); i++) {
            if (x.charAt(i) == 'a') {
                countA += 1;
            }else if (x.charAt(i) != 'h') { // Garantir que só existam
'a' e 'h'
                return false;
            }
        }
        // Verifica se a string termina com um 'h'
        if (!x.endsWith("h")) {
            return false;
        }
```

```
//Verifica se o número de a's está entre 0 e 999
        if (countA < 0 || countA > 999) {
            return false;
        }
        // Se passar por todas as restrições, a string é válida
        return true;
    }
    //Função que conta as strings e as compara para gerar o resultado
    public static void compara_aah(String string1, String string2) {
        int count1 = 0;
        int count2 = 0;
        //Conta o número de a's de Jon
        for (int i = 0; i < string1.length(); i++) {
            if (string1.charAt(i) == 'a') {
                count1 += 1;
            }
        }
        //Conta o número a's que o médico quer
        for (int i = 0; i < string2.length(); i++) {
            if (string2.charAt(i) == 'a') {
                count2 += 1;
            }
        }
        //Se o números de a's dito por Jon for maior ou igual ao número
de a's pedido pelo médico ele poderá ir para a consulta
        if(count1 >= count2) {
            System.out.println("go");
        }else {
            System.out.println("no");
        }
    }
```

```
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        String aah_Jon = scanner.nextLine();

        String aah_Medico = scanner.nextLine();


        if (validaString(aah_Jon) && validaString(aah_Medico)) { //Se
as strings forem válidas

            compara_aah(aah_Jon, aah_Medico); //compara ambas e imprime
o resultado

        } else {

            System.out.println("Entrada inválida."); //caso contrário,
as strings não são válidas

        }

    }

}
```

Exercício 5 – Anti-Palindrome

Pseudo-código

```
Início

    Função palindromo(texto)

        Defina left como 0

        Defina right como o último índice de texto

        Repita até left ser maior ou igual a right:

            Se texto[left] for diferente de texto[right] então

                Retorne Falso

            Incrementa left

            Decrementa right

        Fim do laço

        Retorne Verdadeiro se texto tiver pelo menos 2 caracteres, caso
contrário, Falso


    Função subStringPalindromo(texto)

        Defina n como o comprimento de texto

        i <- 0
```

```
        Enquanto i < n faça:
            j <- i + 2
            Enquanto j <= n faça:
                Obtenha substring de texto entre posições i e j
                Se substring for palíndromo então
                    Retorne Verdadeiro
                Incrementa j
            Fim do laço
            Incrementa i
        Fim Enquanto
        Retorne Falso


    Leia texto
    Remova caracteres não alfabéticos e transforme as letras em
minúsculas


    Se palindromo(texto) for Verdadeiro então
        Escreva "Palindrome"
        Finalize


    Se subStringPalindromo(texto) for Verdadeiro então
        Escreva "Palindrome"
    Senão
        Escreva "Anti-palindrome"
Fim
```

Solução em Java

```
import java.util.*;
public class AntiPalindrome {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String texto = sc.nextLine();
        sc.close();
        //Remoção de caracteres não alfabéticos e transformação de
letras maiusculas em minusculas
   String texto_limpo = texto.replaceAll("[^a-zA-Z]", "").toLowerCase();
```

```
//Verifica se é um palindromo
        if (palindromo(texto_limpo)) {
            System.out.println("Palindrome");
            return;
        }
        //Verifica se as subsequências (substrings) são palindromos
        if (subStringPalindromo(texto_limpo)) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Anti-palindrome");
        }
    }
    // Função para verificar se uma string é palíndromo
    public static boolean palindromo(String texto) {
        int left = 0; //Primeira posição
        int right = texto.length() - 1; //Última posição
        while (left < right) {
            if (texto.charAt(left) != texto.charAt(right)) { // Compara
os caracteres nas duas posições extremas (left e right)
                return false; // Retorna falso caso os caracteres sejam
diferentes
            }
            left++; //Incrementa a posição left caso os caracteres
comparados sejam iguais
            right--; //Decrementa a posição right caso os caractes
comparados sejam iguais
        }
        return texto.length() >= 2; // É necessário que o texto tenha no
mínimo 2 caracteres
    }
```

```
  //Função para verificar se uma subsequência é palindromo
    public static boolean subStringPalindromo(String texto) {
        int n = texto.length(); //Tamanho do texto

        //Verifica todas as substrings de tamanho >= 2
        for (int i = 0; i < n; i++) {
            for (int j = i + 2; j <= n; j++) {
                String substring = texto.substring(i, j); //Obtem a
substring que se inicia na posição i e finaliza na posição j
                if (palindromo(substring)) { //Caso a substring seja
palindromo, retorna true
                    return true;
                }
            }
        }
        return false;
    }
}
```

Exercício 6 – Alphabet

Pseudo-código

```
Início
    Leia a string
    Defina o alfabeto como "abcdefghijklmnopqrstuvwxyz"

    Defina m como o comprimento da string
    Defina n como o comprimento do alfabeto

    Crie uma matriz de tamanho (m+1) por (n+1), inicializando todos os
valores como 0

    Para i de m-1 até 0 faça:
        Para j de n-1 até 0 faça:
            Se a letra na posição i da string for igual à letra na
posição j do alfabeto então
                A matriz[i][j] recebe 1 + matriz[i+1][j+1]
```

```
            Senão
                A matriz[i][j] recebe o maior valor entre matriz[i+1][j]
e matriz[i][j+1]
    Defina result como o tamanho do alfabeto menos matriz[0][0]


    Escreva result
Fim
```

Solução em Python

```
# Ler entrada do usuario
string = input().strip()


# Definindo caracteres do alfabeto
alphabet = "abcdefghijklmnopqrstuvwxyz"


# Obter tamanho dos textos
m, n = len(string), len(alphabet)


# Cria uma matriz de tamanho (m+1) por (n+1), inicializando todos os
valores zerados usando [0] * (n + 1)
matrix = [[0] * (n + 1) for _ in range(m + 1)]


# Preenche a matriz percorrendo do fim para o inicio
for i in range(m - 1, -1, -1):
    for j in range(n - 1, -1, -1):
```

```
# Compara o caractere entre o alfabeto e o texto lido
        if string[i] == alphabet[j]:
            # Se forem iguais, adiciona 1 ao resultado anterior da
matriz
            matrix[i][j] = 1 + matrix[i + 1][j + 1]
        else:
            # Se nao forem iguais, replica o maior resultado anterior da
matriz
            matrix[i][j] = max(matrix[i + 1][j], matrix[i][j + 1])


# Subtrair o tamanho do alfabeto e as letras encontradas para obter
quantas letras estão faltando para completar
result = len(alphabet) - matrix[0][0]
print(result)
```

Exercício 7 – ABC String

Pseudo-código

```
// Ler a entrada do terminal
string = ler_entrada().remover_espacos()
// Variáveis para rastrear as subsequências
estado_0 = 0                // Subsequências prontas para novo trio
estado_1 = dicionário('A':0, 'B':0, 'C':0)  // Subsequências com 1
caractere
estado_2 = dicionário('AB':0, 'AC':0, 'BC':0) // Subsequências com 2
caracteres (pares ordenados)
// Mapeamento de caracteres para pares possíveis
mapeamento_pares = dicionário(
    'A': 'BC',
    'B': 'AC',
    'C': 'AB'
)
// Processar cada caractere da string
para cada caractere c em string faça:
    // Tentar completar um trio usando estado_2
    chave = mapeamento_pares[c]
    se estado_2[chave] > 0 então:
```

```
outros_caracteres = ['B', 'C']
    senão se c == 'B' então:
        outros_caracteres = ['A', 'C']
    senão:
        outros_caracteres = ['A', 'B']
    fim se

    par_encontrado = falso
    para cada outro em outros_caracteres faça:
        se estado_1[outro] > 0 então:
            estado_1[outro] = estado_1[outro] - 1
            novo_par = ordenar_caracteres(outro + c) // Ordena
alfabeticamente
            estado_2[novo_par] = estado_2[novo_par] + 1
            par_encontrado = verdadeiro
            interrompa // Sai do loop de outros caracteres
        fim se
    fim para
    se par_encontrado então:
        continua // Próximo caractere
    fim se
    // Usar ou criar nova subsequência
    se estado_0 > 0 então:
        estado_0 = estado_0 - 1
        estado_1[c] = estado_1[c] + 1
    senão:
        estado_1[c] = estado_1[c] + 1
    fim se
fim para
// Calcular total de subsequências
total_subsequencias = estado_0 + soma_valores(estado_1) +
soma_valores(estado_2)
// Exibir resultado
escrever(total_subsequencias)
Funções Auxiliares:
ordenar_caracteres(s): retorna string com caracteres ordenados
soma_valores(dicionario): retorna soma dos valores do dicionário
```

Solução em Python

```
# Ler a entrada do terminal

string = input().strip()

# Variaveis para rastrear a existencia de subsequencias

state_0 = 0  # para rastrear uma nova subsequencia

state_1 = {'A': 0, 'B': 0, 'C': 0}  # para rastrear uma subsequencia a
partir do primeiro caractere

state_2 = {'AB': 0, 'AC': 0, 'BC': 0}  # para rastrear uma subsequencia
a partir do segundo caractere, notar que as chaves estao em ordem
alfabetica


# mapear cada caractere (A, B ou C) ao par possivel no state_2

pair_map = {'A': 'BC', 'B': 'AC', 'C': 'AB'}


# Iterar entre as letras do texto de entrada

for c in string:

    # Checando se o caractere atual pode completar um trio no state_2

    key = pair_map[c]

    if state_2[key] > 0:

        # Caso sim, completar o trio e rastrear a subsequencia ao
state_0

        state_2[key] -= 1

        state_0 += 1

        continue  # Pula para o proximo caractere


    # Caso nao, vai checar se o caractere pode formar um par valido com
uma subsequencia existente do state_1


    # Escolher outros caracteres possiveis que podem formar um par com
o caractere atual

    if c == 'A':

        others = ['B', 'C']

    elif c == 'B':

        others = ['A', 'C']

    else:

        others = ['A', 'B']

  found = False
```

```
# Iterando nos outros caracteres possiveis
    for other in others:
        if state_1[other] > 0:
            # Encontrou um par valido, mover a contagem da subsequencia
do state_1 para o state_2
            state_1[other] -= 1
            new_pair = ''.join(sorted(other + c))  # Ordenar
alfabeticamente para ser possivel encontrar no state_2
            state_2[new_pair] += 1
            found = True
            break  # Pula para o proximo 'outro' caractere em others

    if found:
        continue  # Se encontrou, pula para o proximo caractere do
texto de entrada

    # Se nao encontrou nenhum par valido, vai usar ou criar uma nova
subsequencia no state_0 ou no state_1
    if state_0 > 0:
        # Usa uma subsequencia existente no state_0 movendo ela para o
state_1
        state_0 -= 1
        state_1[c] += 1
    else:
        # Cria uma nova subsequencia diretamente no state_1
        state_1[c] += 1

# Calcular o numero total de subsequencias
#  state_0: Subsequencias prontar para um novo trio
#  state_1: Subsequencias com 1 caractere
#  state_2: Subsequencias com 2 caracteres
total = state_0 + sum(state_1.values()) + sum(state_2.values())

# Imprimir o resultado
print(total)
```

Exercício 8 – Singularity Cup P2 - Reverse Substring Partitioning

Pseudo-código

```
Início
    // Ler entrada
    n ← ler_inteiro()
    s ← ler_texto().remover_espacos()

    total ← 0

    // Verificar se o primeiro e último caractere são iguais
    Se s[0] = s[-1] então
        total ← total + 1

        // Remover todas ocorrências do caractere das pontas
        caractere_base ← s[0]
        s ← s.remover_prefixo_e_sufixo(caractere_base)

        prev ← ''

        // Contar transições entre caracteres diferentes
        Para cada char em s faça:
            Se char ≠ prev então
                prev ← char
                total ← total + 1
            Fim Se
        Fim Para

        Escrever total
    Senão
        // Caso não seja possível nenhum merge
        Escrever n
    Fim Se
Fim
```

Solução em Python

```
# Ler os dados de entrada
n = int(input())
s = input()

# Variavel para armazernar o tamanho
total = 0

# Checa se o inicio e o fim da string possuem os mesmos caracteres
if s[0] == s[-1]:
    # Se forem iguais, ira percorrer e comparar caractere por caractere
    total += 1
    s = s.strip(s[0]) #remove todos os caracteres iguais a s[0]

    prev = ''
    for char in s:
        if char != prev: # Se forem diferentes, incrementa o tamanho, pois caracteres diferetes nao pode ser combinado
            prev = char
            total += 1

    print(total)

else:
    # Se forem diferentes, nao sera possivel realizar o merge das substrings, portanto retorna o tamanho completo
    print(n)
```