Módulo Pilhas e filas

Curso Estruturas de Dados e Algoritmos Expert

Prof. Dr. Nelio Alves
https://devsuperior.com.br

Lista de exercícios

Soluções:
https://github.com/devsuperior/curso-eda/tree/main/pilhas-filas

Problema "is-balanced"
Fazer uma função para verificar se os parênteses em uma string estão corretamente balanceados, ou seja, cada parêntese aberto "(" tem um correspondente fechado ")" e vice-versa.

Exemplo 1:

Entrada	**Saída**
`(())()`	`true`

Exemplo 2:

Entrada	**Saída**
`())(`	`false`

```
Assinaturas:

Javascript:
function isBalanced(text)

Java:
public static boolean isBalanced(String text)

C#:
public static bool IsBalanced(string text)

Python:
def is_balanced(text)
```

Problema "valid-parentheses" *(ref: Leetcode valid-parentheses)*
Empresas: Amazon, Facebook, Bloomberg, Google, Microsoft, Apple, etc.

Dada uma string contendo somente caracteres '(', ')', '{', '}', '[' e ']', determine se a string é válida. A string é válida se:

- Os símbolos abertos devem ser fechados pelo mesmo tipo de símbolos.
- Os símbolos abertos devem ser fechados na ordem correta.

Exemplo 1:

Entrada	**Saída**
`()([]{})`	`true`

Exemplo 2:

Entrada	**Saída**
`(){]{}`	`false`

Exemplo 3:

Entrada	**Saída**
`(({[()]}{{}}{[]})[])`	`true`

```
Assinaturas:

Javascript:
function validParentheses(text)

Java:
public static boolean validParentheses(String text)

C#:
public static bool validParentheses(string text)

Python:
def validParentheses(text)
```

Problema "remove-duplicates" *(ref: Leetcode remove-all-adjacent-duplicates-in-string)*
Empresas: Facebook, Google, Amazon, Grammarly, Microsoft, etc.

Crie uma função que remova repetidamente os dois primeiros caracteres adjacentes iguais em uma string de letras, até que não haja mais dois caracteres adjacentes repetidos.

Exemplo 1:

Entrada	**Saída**
`abbaca`	`ca`

Exemplo 2:

Entrada	**Saída**
`azxxzy`	`ay`

```
Assinaturas:

Javascript:
function removeDuplicates(text)

Java:
public static String removeDuplicates(String text)

C#:
public static string RemoveDuplicates(string text)

Python:
def remove_duplicates(text)
```

Problema "tickets" *(ref: Leetcode time-needed-to-buy-tickets)*
Empresas: Amazon, X (Twitter)

Há uma fila de N pessoas que querem comprar tickets de um evento.

A quantidade de tickets que cada pessoa deseja comprar deve ser representada por um array "tickets". Por exemplo, se há 3 pessoas A, B e C na fila, e o array "tickets" for [2, 4, 3], significa que a pessoa A quer comprar 2 tickets, a pessoa B quer comprar 4 tickets, e a pessoa C quer comprar 3 tickets.

Cada pessoa leva exatamente 1 segundo para comprar um ticket, e pode comprar apenas um ticket por vez. Se, após comprar um ticket, a pessoa ainda quiser comprar mais tickets, ela deve ir para o final da fila (despreze o tempo de movimentação na fila). Quando a pessoa compra o último ticket que deseja, ela sai da fila.

Faça uma função para receber o array "tickets" e um índice "k". A função deve retornar quantos segundos a pessoa da posição "k" demora para comprar todos seus tickets.

Exemplo 1:

Entrada	Saída

{
  "tickets": [1, 1],
  "k": 1
}
	2

Exemplo 2:

Entrada	Saída

{
  "tickets": [2, 3, 2],
  "k": 2
}
	6

Exemplo 3:

Entrada	Saída

{
  "tickets": [5, 1, 1, 1],
  "k": 0
}
	8

```
Assinaturas:

Javascript:
function timeRequiredToBuy(tickets, k)

Java:
public static int timeRequiredToBuy(int[] tickets, int k)

C#:
public static int timeRequiredToBuy(int[] tickets, int k)

Python:
def time_required_to_buy(tickets, k)
```

Problema "sandwich" *(ref: Leetcode number-of-students-unable-to-eat-lunch)*
Empresas: Amazon, Google, Uber, Microsoft

Uma fila de N alunos aguarda para comer N sanduíches disponíveis.

Cada sanduíche é representado por 0 ou 1, indicando se o sanduíche é redondo ou quadrado respectivamente. Cada aluno também é representado por 0 ou 1, indicando se esse aluno quer comer um sanduíche redondo ou quadrado respectivamente.

A dinâmica da fila de alunos funciona assim: o primeiro aluno da fila observa o primeiro sanduíche disponível. Se o sanduíche for do tipo que o aluno quer comer, o aluno pega o sanduíche e sai da fila, caso contrário o aluno vai para o final da fila.

O processo se repete até quando todos sanduíches forem pegos, ou até quando a fila de alunos for toda percorrida e nenhum aluno quis pegar o primeiro sanduíche disponível. Faça uma função que receba dois vetores representando a sequência de alunos e a sequência de sanduíches disponíveis. A função deve retornar quantos alunos sobraram na fila sem pegar seu sanduíche.

Exemplo 1:

Entrada	**Saída**

{
"students": [1, 1, 0, 0],
"sandwiches": [0, 1, 0, 1]
}
	0

Exemplo 2:

Entrada	**Saída**

{
"students": [1, 1, 1, 0, 0, 1],
"sandwiches": [1, 0, 0, 0, 1, 1]
}
	3

Assinaturas:

Javascript:
```
function countStudents(students, sandwiches)
```

Java:
```
public static int countStudents(int[] students, int[] sandwiches)
```

C#:
```
public static int CountStudents(int[] students, int[] sandwiches)
```

Python:
```
def count_students(students, sandwiches)
```