

# Listas Lineares

## Estrutura de Dados

**Professor:** Henrique Viana Oliveira, [henriq.viana@uece.br](mailto:henriq.viana@uece.br)

### Pilhas

#### Exercício: 1.

**(Parênteses Válidos)** Dada uma string  $s$  contendo apenas os caracteres  $(, ), \{, \}, [ e ]$ , determine se a string de entrada é válida. Uma string de entrada é válida se:

- Os colchetes de abertura devem ser fechados pelo mesmo tipo de colchete.
- Os colchetes de abertura devem ser fechados na ordem correta.
- Cada colchete de fechamento tem um colchete de abertura correspondente do mesmo tipo.

**Exemplo 1. Entrada:**  $s = ()$ . **Saída:** verdadeiro.

**Exemplo 2. Entrada:**  $s = ()[]$ . **Saída:** verdadeiro.

**Exemplo 3. Entrada:**  $s = ([$ . **Saída:** falso.

**Exemplo 4. Entrada:**  $s = ([)]$ . **Saída:** verdadeiro.

**Exemplo 5. Entrada:**  $s = ([)]$ . **Saída:** falso.

#### Exercício: 2.

**(Jogo de Beisebol)** Você está registrando os placares de um jogo de beisebol com regras estranhas. No início do jogo, você começa com um registro vazio.

Você recebe uma lista de strings  $ops$ , onde  $ops[i]$  é a  $i$ -ésima operação que você deve aplicar ao registro e é uma das seguintes:

- Um inteiro  $x$ . Registre uma nova pontuação de  $x$ .
- '+' . Registre uma nova pontuação que seja a soma das duas pontuações anteriores.
- 'D' . Registre uma nova pontuação que seja o dobro da pontuação anterior.
- 'C' . Invalide a pontuação anterior, removendo-a do registro.

Retorne a soma de todas as pontuações no registro após aplicar todas as operações.

**Exemplo 1. Entrada:**  $ops = [5, 2, C, D, +]$ . **Saída:** 30. **Explicação:**

5 - Adicione 5 ao registro, o registro agora é [5].

2 - Adicione 2 ao registro, o registro agora é [5, 2].

C - Invalida e remove a pontuação anterior, o registro agora é [5].

D - Adicione  $2 * 5 = 10$  ao registro, o registro agora é [5, 10].

+ - Adicione  $5 + 10 = 15$  ao registro, o registro agora é  $[5, 10, 15]$ .

A soma total é  $5 + 10 + 15 = 30$ .

**Exemplo 2. Entrada:**  $ops = [5, -2, 4, C, D, 9, +, +]$ . **Saída:** 27. **Explicação:**

5 - Adicione 5 ao registro, o registro agora é  $[5]$ .

-2 - Adicione  $-2$  ao registro, o registro agora é  $[5, -2]$ .

4 - Adicione 4 ao registro, o registro agora é  $[5, -2, 4]$ .

C - Invalida e remove a pontuação anterior, o registro agora é  $[5, -2]$ .

D - Adicione  $2 * -2 = -4$  ao registro, o registro agora é  $[5, -2, -4]$ .

9 - Adicione 9 ao registro, o registro agora é  $[5, -2, -4, 9]$ .

+ - Adicione  $-4 + 9 = 5$  ao registro, o registro agora é  $[5, -2, -4, 9, 5]$ .

+ - Some  $9 + 5 = 14$  ao registro, que agora é  $[5, -2, -4, 9, 5, 14]$ .

A soma total é  $5 + -2 + -4 + 9 + 5 + 14 = 27$ .

**Exemplo 3. Entrada:**  $ops = [1, C]$ . **Saída:** 0. **Explicação:**

1 - Adiciona 1 ao registro, o registro agora é  $[1]$ .

C - Invalida e remove a pontuação anterior, o registro agora é  $[]$ .

Como o registro está vazio, a soma total é 0.

### Exercício: 3.

**(Preços Finais com Desconto Especial em Loja)** Você recebe um vetor de inteiros  $precos$ , onde  $precos[i]$  é o preço do  $i$ -ésimo item em uma loja. Há um desconto especial para itens na loja. Se você comprar o  $i$ -ésimo item, receberá um desconto equivalente a  $precos[j]$ , onde  $j$  é o índice mínimo tal que  $j > i$  e  $precos[j] \leq precos[i]$ . Caso contrário, você não receberá nenhum desconto. Retorne um vetor de inteiros resposta, onde  $resposta[i]$  é o preço final que você pagará pelo  $i$ -ésimo item da loja, considerando o desconto especial.

**Exemplo 1. Entrada:**  $precos = [8, 4, 6, 2, 3]$ . **Saída:**  $[4, 2, 4, 2, 3]$ . **Explicação:**

Para o item 1 com  $preco[1] = 8$ , você receberá um desconto equivalente a  $precos[2] = 4$ ; portanto, o preço final que você pagará será  $8 - 4 = 4$ .

Para o item 2 com  $preco[2] = 4$ , você receberá um desconto equivalente a  $precos[4] = 2$ ; portanto, o preço final que você pagará será  $4 - 2 = 2$ .

Para o item 3 com  $preco[3] = 6$ , você receberá um desconto equivalente a  $precos[4] = 2$ ; portanto, o preço final que você pagará será  $6 - 2 = 4$ .

Para os itens 3 e 4, você não receberá nenhum desconto.

**Exemplo 2. Entrada:**  $precos = [1, 2, 3, 4, 5]$ . **Saída:**  $[1, 2, 3, 4, 5]$ .

**Exemplo 3. Entrada:**  $precos = [10, 1, 1, 6]$ . **Saída:**  $[9, 0, 1, 6]$ .

#### Exercício: 4.

**(Pasta de log do rastreador)** O sistema de arquivos *Leetcode* mantém um log sempre que um usuário realiza uma operação de alteração de pasta. As operações são descritas abaixo:

`../`: Move para a pasta pai da pasta atual. (Se você já estiver na pasta principal, permaneça na mesma pasta).

`./`: Permanece na mesma pasta.

`x/`: Move para a pasta filha chamada *x* (É garantido que esta pasta sempre exista).

Você recebe uma lista de strings *logs*, onde *logs[i]* é a operação realizada pelo usuário na etapa *i*. O sistema de arquivos inicia na pasta principal e, em seguida, as operações em *logs* são realizadas. Retorne o número mínimo de operações necessárias para retornar à pasta principal após as operações de alteração de pasta.

**Exemplo 1. Entrada:** *logs* = [*d1/*, *d2/*, *../*, *d21/*, *../*]. **Saída:** 2. **Explicação:** Use esta operação de alteração de pasta `../` 2 vezes e retorne à pasta principal.

**Exemplo 2. Entrada:** *logs* = [*d1/*, *d2/*, *../*, *d3/*, *../*, *d31/*]. **Saída:** 3.

**Exemplo 3. Entrada:** *logs* = [*d1/*, *../*, *../*, *../*]. **Saída:** 0.

#### Exercício: 5.

**(Prefixo Reverso da Palavra)** Dada uma string *word* e um caractere *ch*, inverta o segmento de *word* que começa no índice 1 e termina no índice da primeira ocorrência de *ch* (inclusive). Se o caractere *ch* não existir em *word*, não faça nada.

Por exemplo, se *word* = *abcdefd* e *ch* = *d*, você deve inverter o segmento que começa em 1 e termina em 4 (inclusive). A string resultante será *dcbaefd*.

Retorne a string resultante.

**Exemplo 1. Entrada:** *word* = *abcdefd*, *ch* = *d*. **Saída:** *dcbaefd*. **Explicação:** A primeira ocorrência de *d* está no índice 4. Inverta a parte da palavra de 1 a 4 (inclusive), a string resultante é *dcbaefd*.

**Exemplo 2. Entrada:** *word* = *xyzzxe*, *ch* = *z*. **Saída:** *zyxxe*.

**Exemplo 3. Entrada:** *word* = *abcd*, *ch* = *z*. **Saída:** *abcd*.

### Filas

#### Exercício: 6.

**(Primeiro Caractere Único em uma String)** Dada uma string *s*, encontre o primeiro caractere não repetido nela e retorne seu índice. Se não existir, retorne -1.

**Exemplo 1. Entrada:** *s* = *leetcode*. **Saída:** 1. **Explicação:** O caractere *l* no índice 1 é o primeiro caractere que não ocorre em nenhum outro índice.

**Exemplo 2. Entrada:**  $s = \text{loveleetcode}$ . **Saída:** 3.

**Exemplo 3. Entrada:**  $s = \text{aabb}$ . **Saída:** -1.

#### Exercício: 7.

**(Tempo Necessário para Comprar Ingressos)** Há  $n$  pessoas em uma fila para comprar ingressos, onde a primeira pessoa está na frente da fila e a  $n$ -ésima pessoa está no final da fila. Você recebe um vetor de inteiros *tickets* de comprimento  $n$ , onde o número de ingressos que a  $i$ -ésima pessoa gostaria de comprar é  $tickets[i]$ . Cada pessoa leva exatamente 1 segundo para comprar um ingresso. Uma pessoa só pode comprar 1 ingresso por vez e precisa voltar ao final da fila (o que acontece instantaneamente) para comprar mais ingressos. Se uma pessoa não tiver mais ingressos para comprar, ela sairá da fila. Retorne o tempo que a pessoa inicialmente na posição  $k$  levou para finalizar a compra dos ingressos.

**Exemplo 1. Entrada:**  $tickets = [2, 3, 2]$ ,  $k = 3$ . **Saída:** 6. **Explicação:**

A fila começa como  $[2, 3, \underline{2}]$ , onde a  $k$ -ésima pessoa está sublinhada.

Após a pessoa da frente comprar um ingresso, a fila se torna  $[3, 2, 1]$  em 1 segundo.

Continuando esse processo, a fila se torna  $[\underline{2}, 1, 2]$  em 2 segundos.

Continuando esse processo, a fila se torna  $[1, 2, \underline{1}]$  em 3 segundos.

Continuando esse processo, a fila se torna  $[2, \underline{1}]$  em 4 segundos. Observação: a pessoa da frente saiu da fila.

Continuando esse processo, a fila se torna  $[\underline{1}, 1]$  em 5 segundos.

Continuando esse processo, a fila se torna  $[1]$  em 6 segundos. A  $k$ -ésima pessoa comprou todos os seus ingressos, então retorne 6.

**Exemplo 2. Entrada:**  $tickets = [5, 1, 1, 1]$ ,  $k = 1$ . **Saída:** 8.

#### Exercício: 8.

**(Número de Alunos que não Conseguem Almoçar)** O refeitório da escola oferece sanduíches circulares e quadrados no intervalo do almoço, designados pelos números 0 e 1, respectivamente. Todos os alunos formam uma fila. Cada aluno prefere sanduíches quadrados ou circulares. O número de sanduíches no refeitório é igual ao número de alunos. Os sanduíches são colocados em uma pilha. Em cada etapa:

- Se o aluno da frente da fila preferir o sanduíche do topo da pilha, ele o pegará e sairá da fila.
- Caso contrário, ele o deixará e irá para o final da fila.

Isso continua até que nenhum dos alunos da fila queira pegar o sanduíche do topo e, portanto, não consiga comer. Você recebe dois vetores de inteiros, *alunos* e *sandwiches*, onde  $sandwiches[i]$  é o tipo do  $i$ -ésimo sanduíche na pilha ( $i = 0$  é o topo da pilha) e  $alunos[j]$  é a preferência do  $j$ -ésimo aluno na fila inicial ( $j = 0$  é o início da fila). Retorne o número de alunos que não conseguem comer.

**Exemplo 1. Entrada:**  $alunos = [1, 1, 0, 0]$ ,  $sandwiches = [0, 1, 0, 1]$ . **Saída:** 0.  
**Explicação:**

O aluno da frente sai do sanduíche de cima e retorna ao final da fila, resultando em  $alunos = [1, 0, 0, 1]$ .

O aluno da frente sai do sanduíche de cima e retorna ao final da fila, resultando em  $alunos = [0, 0, 1, 1]$ .

O aluno da frente pega o sanduíche de cima e sai da fila, resultando em  $alunos = [0, 1, 1]$  e  $sandwiches = [1, 0, 1]$ .

O aluno da frente sai do sanduíche de cima e retorna ao final da fila, resultando em  $alunos = [1, 1, 0]$ .

O aluno da frente pega o sanduíche de cima e sai da fila, resultando em  $alunos = [1, 0]$  e  $sandwiches = [0, 1]$ .

O aluno da frente sai do sanduíche de cima e retorna ao final da fila, resultando em  $alunos = [0, 1]$ .

O aluno da frente pega o sanduíche de cima e sai da fila, resultando em  $alunos = [1]$  e  $sandwiches = [1]$ .

O aluno da frente pega o sanduíche de cima e sai da fila, formando  $alunos = []$  e  $sandwiches = []$ .

Portanto, todos os alunos conseguem comer.

**Exemplo 2. Entrada:**  $alunos = [1, 1, 1, 0, 0, 1]$ ,  $sandwiches = [1, 0, 0, 0, 1, 1]$ . **Saída:** 3.

### Exercício: 9.

**(Revelar Cartas em Ordem Crescente)** Você recebe um baralho de matrizes de inteiros. Há um baralho de cartas em que cada carta possui um número inteiro único. O número inteiro na  $i$ -ésima carta é  $baralho[i]$ . Você pode ordenar o baralho na ordem que desejar. Inicialmente, todas as cartas começam viradas para baixo (não reveladas) em um baralho. Você executará os seguintes passos repetidamente até que todas as cartas sejam reveladas:

- Pegue a carta do topo do baralho, revele-a e retire-a do baralho.
- Se ainda houver cartas no baralho, coloque a próxima carta do topo do baralho no fundo do baralho.
- Se ainda houver cartas não reveladas, volte para a etapa 1. Caso contrário, pare.

Retorne uma ordenação do baralho que revelaria as cartas em ordem crescente.

**Exemplo 1. Entrada:**  $baralho = [17, 13, 11, 2, 3, 5, 7]$ . **Saída:**  $[2, 13, 3, 11, 5, 17, 7]$ .  
**Explicação:**

Pegamos o baralho na ordem  $[17, 13, 11, 2, 3, 5, 7]$  (essa ordem não importa) e o reordenamos.

Após a reordenação, o baralho começa como  $[2,13,3,11,5,17,7]$ , onde 2 é o topo do baralho.

Revelamos 2 e movemos 13 para o fundo. O baralho agora é  $[3,11,5,17,7,13]$ .

Revelamos 3 e movemos 11 para o fundo. O baralho agora é  $[5,17,7,13,11]$ .

Revelamos 5 e movemos 17 para o fundo. O baralho agora é  $[7,13,11,17]$ .

Revelamos 7 e movemos 13 para o fundo. O baralho agora é  $[11,17,13]$ .

Revelamos 11 e movemos 17 para o fundo. O baralho agora é  $[13,17]$ .

Revelamos 13 e movemos 17 para o fundo. O baralho agora é  $[17]$ .

Revelamos 17.

Como todas as cartas reveladas estão em ordem crescente, a resposta está correta.

### Exercício: 10.

**(Encontre o Vencedor do Jogo Circular)** Há  $n$  amigos jogando. Os amigos estão sentados em círculo e são numerados de 1 a  $n$  no sentido horário. Mais formalmente, movendo-se no sentido horário a partir do  $i$ -ésimo amigo, você chega ao  $(i + 1)$ -ésimo amigo para  $1 \leq i < n$ , e movendo-se no sentido horário a partir do  $n$ -ésimo amigo, você chega ao primeiro amigo. As regras do jogo são as seguintes:

1. Comece no primeiro amigo.
2. Conte os próximos  $k$  amigos no sentido horário, incluindo o amigo com o qual você começou. A contagem se repete ao longo do círculo e pode contar alguns amigos mais de uma vez.
3. O último amigo que você contou sai do círculo e perde o jogo.
4. Se ainda houver mais de um amigo no círculo, volte para a etapa 2, começando pelo amigo imediatamente no sentido horário do amigo que acabou de perder e repita.
5. Caso contrário, o último amigo no círculo vence o jogo.

Dado o número de amigos,  $n$ , e um inteiro  $k$ , retorne o vencedor do jogo.

**Exemplo 1. Entrada:**  $n = 5, k = 2$ . **Saída:** 3. **Explicação:** Aqui estão os passos do jogo:

1. Comece com o amigo 1.
2. Conte 2 amigos no sentido horário, que são os amigos 1 e 2.
3. O amigo 2 sai do círculo. O próximo a começar é o amigo 3.
4. Conte 2 amigos no sentido horário, que são os amigos 3 e 4.
5. O amigo 4 sai do círculo. O próximo a começar é o amigo 5.
6. Conte 2 amigos no sentido horário, que são os amigos 5 e 1.
7. O amigo 1 sai do círculo. O próximo a começar é o amigo 3.
8. Conte 2 amigos no sentido horário, que são os amigos 3 e 5.

9. O amigo 5 sai do círculo. Restou apenas o amigo 3, então ele é o vencedor.

**Exemplo 2. Entrada:**  $n = 6, k = 5$ . **Saída:** 1. **Explicação:** Os amigos saem nesta ordem: 5, 4, 6, 2, 3. O vencedor é o amigo 1.