Questão 1

Completo

Não avaliada

This is the way

Após a queda do Império (O Retorno de Jedi), a galáxia passou a viver momentos de paz. Infelizmente, o lado Sombrio da Força não foi completamente destruído e remanescentes do império passaram a se organizar. Anos depois, uma nova ameaça surgiu, no que ficou conhecido como a Primeira Ordem (O Despertar da Força). Nossa história se passa no período entre a queda do Império e a ascensão da Primeira Ordem.

Um pouco antes da queda do Império, aconteceu a batalha pelo planeta Mandaloren, que ficou conhecida como **O Grande Expurgo**. Os Mandalorianos foram massacrados por tropas do Império e os poucos sobreviventes passaram a se esconder e a viver nas sombras. Sem acesso ao seu poderoso metal Beskar, os Mandalorianos passaram a trabalhar como caçadores de recompensa. Sempre em busca deste precioso metal para recompor suas armaduras e armas.

Em *O Mandaloriano*, conhecemos o personagem Mando: um lendário guerreiro que foi resgatado quando criança por um mandaloriano, que também o acompanhou em seu crescimento. Imerso na cultura Mandaloriana ortodoxa, ele conquistou o respeito dos seus pares e montou sua armadura com o precioso Beskar. Ao longo de suas aventuras, conheceu o jovem Grogu (baby Yoda), resgatando-o do Império e comprando uma briga intergaláctica no processo.

Grogu demonstra ter o dom da força! Mando jurou protegê-lo e entregá-lo para um mestre Jedi. Enquanto procura pelos Jedi remanescentes, Mando e Grogu fogem do Império, ou do que restou dele. Com muito cuidado, Mando precisa realizar uma *side quest* de forma a conseguir o Beskar e reconstruir sua armadura.

Neste trabalho, iremos utilizar nossos conhecimentos em Estruturas de Dados para ajudar Mando a selecionar as melhores missões disponíveis! Nosso objetivo é auxiliá-lo a maximizar seu ganho para que ele possa reconstruir sua armadura o quanto antes!

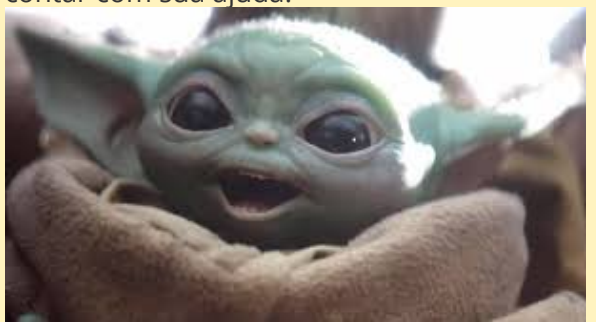
Podemos contar com a sua ajuda, jovem Padawan?

Escolha uma opção:

- a. Sim! This is the way!
- b. Sim! This **IS** the way!

Grogu ficou muito feliz em saber que pode

contar com sua ajuda!



Sua resposta está correta.

As respostas corretas são: Sim! This is the way!, Sim! This **IS** the way!

Questão 2

Completo

Não avaliada

Contextualização

Mando não possui muito tempo disponível. Além de fugir das tropas do Império Galáctico, fazer o bem e procurar por informações do paradeiro de algum cavaleiro Jedi, ele precisa cuidar de Grogu! Por isso, é urgente maximizar o uso de seu tempo disponível, escolhendo as missões com o maior retorno de Beskar possível.

Sua missão será ajudar Mando em duas tarefas específicas. A primeira é decodificar a lista de missões disponíveis. A segunda é selecionar o subconjunto de missões que maximizem a aquisição de Beskar (maximizem seu retorno financeiro), dada a quantidade de tempo que Mando tem disponível.

No seu primeiro desafio, você irá decodificar o conteúdo da lista de missões enviadas pela Guilda. Por se tratar de um conteúdo sensível, Greef Karga utiliza um tipo de criptografia para tentar embaralhar o conteúdo da lista antes de transmiti-la.

Fique tranquilo, pois Kuiil se adiantou e implementou toda a parte de recepção da transmissão interplanetária. Ele também implementou uma estrutura do tipo Deque para que você, obrigatoriamente, a utilize para implementar o algoritmo de deciframento.

Você até protestou dizendo que não era necessário o uso de Deque para resolver esta parte do problema. Contudo, Kuiil irredutível disse:

Depois disso, não tem jeito. Se nem Mando foi capaz de fazê-lo mudar de opinião no episódio *The Reckoning*, não será você que conseguirá. Ou seja, é obrigatório o uso da estrutura tipo Deque e o sistema de correção automático verificará se seu código está utilizando a estrutura fornecida por Kuiil. Esse código encontra-se descrito abaixo e já fará parte do sistema de correção. Sendo assim, não é necessário preocupar-se com a declaração desta classe. Ou seja, basta simplesmente utilizá-la.

```
class Deque:
    def __init__(self):
        self.__items = []

    def is_empty(self):
        return self.__items == []

    def add_front(self, item):
        self.__items.append(item)

    def add_rear(self, item):
        self.__items.insert(0, item)

    def remove_rear(self):
        return self.__items.pop(0)

    def remove_front(self):
        return self.__items.pop()

    def size(self):
        return len(self.__items)

    def __str__(self):
        sdeque = ''
        for i in self.__items:
            sdeque += i
        return sdeque
```

Agora que apresentamos uma contextualização geral do que será feito, apresentaremos alguns conceitos que serão utilizados no trabalho. Em seguida, serão fornecidas a formalização do problema, a definição do que precisa ser feito, bem como dos parâmetros de entrada e saída desta atividade.

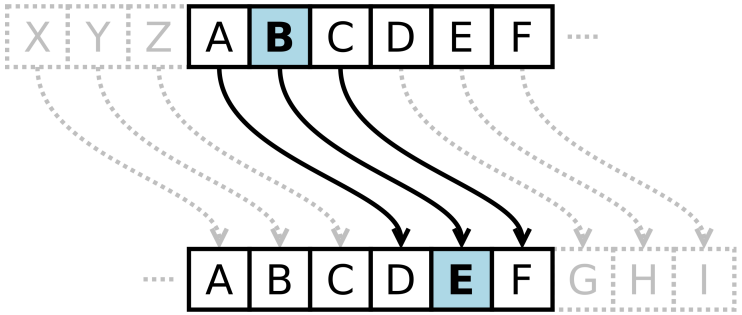
Algoritmo de Deciframento

O algoritmo utilizado pela Guilda é bem simples. Consiste em realizar um deslocamento de X posições fixas em cima de um alfabeto pré-definido. Note que tanto o alfabeto quanto a quantidade X de posições de deslocamento (chave) são parâmetros da solução a ser implementada.

A transformação pode ser representada alinhando dois alfabetos; o alfabeto cifrado é o alfabeto normal rotacionado à direita por um número de posições (chave). Por exemplo, abaixo está uma cifra usando uma rotação à direita de três posições (o parâmetro de troca, três neste caso, é usado como chave).

Normal: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ ' (texto sem aspas. Contém um espaço no final)

Cifrado: DEFGHIJKLMNOPQRSTUVWXYZABC



No exemplo acima, a letra B será cifrada com o uso da letra E.

Para cifrar uma mensagem, deve-se observar cada letra da mensagem na linha "Normal" e escrever a letra correspondente na linha "Cifrado". Para decifrar, deve-se fazer o contrário.

Normal: THIS IS THE WAY
Cifrado: WKLVCLVCWKHCZDA

O exemplo acima pode ser obtido com o uso do dicionário `'ABCDEFGHIJKLMNOPQRSTUVWXYZ '` (sem aspas) e utilizando chave (deslocamento) de valor 3 a direita. O valor original é obtido a partir do mesmo dicionário e utilizando a mesma chave, contudo com deslocamento a esquerda. Note que o caractere "espaço" faz parte deste alfabeto!

A criptografia também pode ser representada usando aritmética modular, primeiro transformando as letras em números, de acordo com o esquema: $A = 0$, $B = 1$, ..., $Z = 25$. A ciframento de uma letra x por uma troca fixa $chave$ pode ser descrita matematicamente como:

$$\mathbb{E}_n(x) = (x + chave)\ mod\ 26$$

O deciframento é feita de forma similar:

$$\mathbb{E}_n(x) = (x - chave)\ mod\ 26$$

Onde 26 é o tamanho do alfabeto utilizado.

O Problema de Maximizar os Ganhos de Mando

Uma vez de posse da lista de missões disponíveis para serem feitas, Mando precisa selecionar um subconjunto de missões, cuja soma de tempo necessário para cumprí-las não ultrapasse a quantidade de tempo disponível que ele tem e cuja recompensa seja a maior possível. Colocando de uma maneira mais formal, temos:

Dado um conjunto $I = \{1, 2, 3, \ldots, N\}$ de missões, onde cada missão $i \in I$ possui uma duração de tempo W_i horas e um valor de recompensa de V_i gramas de Beskar associados. Dado uma Janela de tempo disponível de W horas, selecione um subconjunto $S \in I$ de missões, tal que $\sum_{i \in S} W_i \leq W$ e $\sum_{i \in S} V_i$ é máximo.

Sendo um bom amigo, Kuiil identificou que o problema descrito encaixa-se no problema Intergalático conhecido como: O problema da Mochila Inteira ou Binária. Ele também verificou a existência de diversas formas de resolver este problema. Porém, tome cuidado. Nem toda solução retorna o valor máximo! Essas conhecidas soluções possuem complexidade computacional diferentes. Sendo assim, você é convidado a pesquisar mais sobre o assunto, de forma a permiti-lo resolver este problema. Recomendamos os vídeos link1 e link2 de uma Jedi sobre o assunto.

Tudo certo, Jovem Padawan? Podemos prosseguir?

- a. Sim, This is the way!
- b. Sim, I have Spoken!

Sua resposta está correta.

As respostas corretas são: Sim, This is the way!,
Sim, I have Spoken!

Questão 3

Parcialmente correto

Atingiu 9,17 de 10,00

A partir de agora focaremos na implementação do problema. Recapitulando o que precisa ser feito:

Você irá ajudar Mando em duas tarefas específicas. A primeira é decodificar a lista de missões disponíveis. A segunda é selecionar o subconjunto de missões que maximizem o retorno em Beskar (estamos falando de maximizar o retorno financeiro), dada a quantidade de tempo que Mando tem disponível.

Ambas tarefas serão implementadas e avaliadas nesta questão. Para isso, preste atenção no que está sendo solicitado abaixo.

Primeira Tarefa - Decodificação da Lista de Missões

Para ser possível decodificar a lista de missões, as seguintes assinaturas de funções precisam ser implementadas exatamente seguindo a descrição apresentada abaixo.

- **def adicionar_alfabeto(deque, alfabeto) -** esta função adiciona ao objeto **deque**, do tipo Deque, cada caractere contido no parâmetro **alfabeto** (tipo string). A sequencia de caracteres adicionadas em **deque** deve seguir exatamente a mesma sequência contida no parâmetro **alfabeto**. Utilize os métodos já implementados na classe Deque para manipular esta estrutura de dados.
- def decifrar(deque, texto_cifrado, chave) - esta função recebe como parâmetro o **texto_cifrado** e utilizando o parâmetro **chave** (deslocamento) aplica o algoritmo de decifração apresentado. Neste ponto, o parâmetro **deque** é a estrutura de dados do tipo Deque que contém o alfabeto utilizado para cifragem do **texto_cifrado**. Você deve utilizar os métodos estudados na estrutura do

tipo Deque, de forma a ser possível navegar pelo alfabeto armazenado. Note que no processo de remover sistematicamente o elemento da cabeça (front) e adicioná-lo na calda (rear), você estará movendo os elementos do alfabeto armazenados na estrutura para a direita. O processo de remover sistematicamente o elemento da calda (rear) e adicioná-lo na cabeça (front), você estará movendo os elementos do alfabeto para a esquerda. Todo o elemento removido do objeto **deque** precisa ser novamente inserido. Ao final, o objeto **deque**, recebido como parâmetro da função, precisa continuar a ter todos os elementos originais do dicionário previamente armazenados.

Entrada

A função adicionar_alfabeto(**deque**, **alfabeto**) tem como parâmetro o objeto **deque** do tipo Deque. Inicialmente, o objeto **deque** não possui valor armazenado. O parâmetro **alfabeto** é uma string, sendo $1 \leq len(\textbf{alfabeto}) \leq 149$, contendo o alfabeto que será utilizado pela função decifrar(). É assegurado que não há caracteres duplicados no alfabeto informado.

A função decifrar(deque, texto_cifrado, chave) tem como parâmetro o objeto **deque** do tipo Deque, sendo $1 \leq len(\textbf{alfabeto}) \leq 149$, inicializado com alfabeto a ser utilizado no processo de decriptação. O parâmetro **texto_cifrado** representa o texto a ser decifrado pela função decifrar(). Sendo $0 < len(\textbf{texto\_cifrado})$. O parâmetro **chave** representa a quantidade de posições de deslocamento a ser aplicado no algoritmo de decifração. Onde, $0 \leq$ **chave**.

Saída

A função adicionar_alfabeto() não retorna nenhum valor. Somente o conteúdo do objeto **deque** é alterado conforme descrição apresentada.

A função decifrar() retorna uma string contendo o conteúdo de **texto_cifrado**, agora decifrado.

Segunda Tarefa - Selecionar Subconjunto de Missões

Uma vez implementadas as funções da primeira tarefa, você está pronto para ajudar Mando a definir o subconjunto de Missões que maximizem o seu ganho.

Seja o problema definido por:

> Dado um conjunto $I = \{1, 2, 3, \ldots, N\}$ de missões, onde cada missão $i \in I$ possui uma duração de tempo W_i horas e um valor de recompensa de V_i gramas de Beskar associados. Dado uma Janela de tempo disponível de W horas, selecione um subconjunto $S \in I$ de missões, tal que $\sum_{i \in S} W_i \leq W$ e $\sum_{i \in S} V_i$ é máximo.

Entrada

Para esta tarefa, a implementação da descrição da entrada e da solução **DEVERÃO** ser implementadas no escopo da função selecionar_subconjunto_missoes(). É possível a implementação de sub-funções desde que elas, OBRIGATORIAMENTE, sejam invocadas dentro da função selecionar_subconjunto_missoes(). Códigos implementados fora do escopo desta função irão gerar erros nos casos de testes.

A entrada é composta por várias linhas. A primeira linha possui um inteiro W ($0 \leq W \leq 1000$) que indica a quantidade de horas que Mando tem disponível para participar das missões. A próxima linha possui um inteiro M, que pode assumir somente os valores 0 ou 1. O valor 1 indica que as missões pertencentes ao conjunto solução S devem ser apresentadas, conforme descrito na seção **Saída**. O valor 0 indica que elas não devem ser apresentadas. A próxima linha possui um inteiro O ($0 \leq O \leq 3$), que representa o índice da coluna em que deve ser iniciada a sequencia para a aplicação do algoritmo de ordenação crescente (ordenação crescente seguindo critério de múltiplas colunas), sendo elas:

- O = 0 -- Iniciar a ordenação da resposta pela coluna que representa o **nome** da missão, seguido por **duração**, **valor** e **nível de dificuldade** (0, 1, 2, 3);
- O = 1 -- Iniciar a ordenação da resposta pela coluna que representa a **duração** da missão, seguido por **nome**, **valor** e **nível de dificuldade** (1, 0, 2, 3);
- O = 2 -- Iniciar a ordenação da resposta pela coluna que representa a o **valor** da missão, seguido por **nome**, **duração** e **nível de dificuldade** (2, 0, 1, 3);
- O = 3 -- Iniciar a ordenação da resposta pela coluna que representa o **nível de dificuldade** da missão, seguido por **nome**, **duração** e **valor** (3, 0, 1, 2);

A próxima linha possui um valor string A ($0 < len(\text{A}) \leq 149$), que representa o alfabeto utilizado pela função cifrar e deve ser utilizado na função de decifrar(). É assegurado que não há caracteres duplicados no alfabeto informado. A próxima linha possui um valor inteiro C ($0 \leq$ C) que indica a quantidade de posições de deslocamento a ser aplicado no algoritmo de decriptação (chave). A próxima linha possui um valor inteiro N ($0 \leq N \leq 50$) que indica o número de missões cifradas que serão informadas. A seguir, as próximas N linhas possuem, cada uma, uma string i que representa uma missão cifrada. O conteúdo da string i encontra-se delimitado entre colchetes "[**texto_cifrado**]". É importante notar que os colchetes são apenas delimitadores e não fazem parte do conteúdo a ser decifrado.

Exemplo:

```
print(texto_cifrado)
"[d53vQurQ3z99r5Lu:8rtr5L'r258L4z'v2Quzwzt:2uruv]"
```

O **texto_cifrado** a ser decifrado (sem áspas):

```
print(decifrar(d, "d53vQurQ3z99r5Lu:8rtr5L'r258L4z'v2Quzwzt:2uruv", chave))
Nome_da_missao,duracao,valor,nivel_dificuldade
```

Após a decodificação do **texto_cifrado**, cada missão i é formada pelos seguintes parâmetros separados por vírgula.

- **Nome_da_missão** - string de texto, com $0 < len(nome\_da\_missao)$

- **duração** - valor inteiro, com $0 < duracao \leq 30$, representa W_i.
- **valor** - valor inteiro, com $0 < valor \leq 331$, representa V_i.
- **nivel_dificuldade** - string que pode assumir um dos valores: ***hard***, ***medium*** e ***easy***.

Exemplo:

```
print(decifrar(d, "Johw0lyg8Cg olgThukhsvyphuB87B979Bohyk", chave))
Chapter 1: The Mandalorian,10,202,hard
```

Saída

A saída é formada por multiplas linhas. Deve ser apresentado o conjunto solução S, tal que $S \in I$, $\sum_{i \in S} W_i \leq W$ e $\sum_{i \in S} V_i$ é máximo. As missões pertencentes ao conjunto solução S devem ser apresentadas uma por linha e estarem ordenadas segundo o parâmetro de entrada O, caso o parâmetro de entrada de M for $M == 1$. Em seguida, deve ser apresentado o tempo disponível para Mando após descontado os tempos W_i referentes as missões que fazem parte do conjunto solução S. Por fim, deve ser apresentado o valor em Beskar que será obtido por Mando ao realizar as missões selecionadas no conjunto solução S.

É importante destacar que o parâmetro de entrada M diz respeito somente a apresentação das missões pertecentes ao conjunto solução S. Os demais valores de saída devem ser apresentados independete do valor de M.

O formato em que essas informações devem ser informadas estão definidos nos casos de teste exemplo abaixo.

For example:

Test	Input	Result

`d = Deque()`
`adicionar_alfabeto(d,`
`'ABCDEFGHIJKLMNOPQRSTUVWXYZ')`
`print(f'Alfabeto: {d}')`
`print(f'Tamanho: {len(str(d))}')`
		`Alfabeto:`
		`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
		`Tamanho: 26`

`d = Deque()`
`adicionar_alfabeto(d, ' ')`
`print(f'Alfabeto: {d}')`
`print(f'Tamanho: {len(str(d))}')`
		`Alfabeto:`
		`Tamanho: 1`

Test	Input	Result
d = Deque() adicionar_alfabeto(d, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ') texto_cifrado = 'DBTB' print(f'texto_plano: {decifrar(d, texto_cifrado, 1)}') print(f'{str(d)}') print(f'{len(str(d))}')		texto_plano: CASA ABCDEFGHIJKLMNOPQRSTUVWXYZ 26
d = Deque() adicionar_alfabeto(d, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ') texto_cifrado = 'ECUC' print(f'texto_plano: {decifrar(d, texto_cifrado, 2)}') print(f'{str(d)}') print(f'{len(str(d))}')		texto_plano: CASA ABCDEFGHIJKLMNOPQRSTUVWXYZ 26
selecionar_subconjunto_missoes()	1000 1 0 ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789,;:'()_ 19 2 [l0xsg4wsZ7w;st6ws:0xsfx)MECMDFDM0t,w] [l0xsWt6vxs7ysW,tz76;MDDMFDFM5xw1'5]	The Dance of Dragons, 11, 313, medium The Old Gods and the New, 20, 131, hard Tempo restante: 969 Valor: 444
selecionar_subconjunto_missoes()	1000 0 0 ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789,;:'()_ 19 2 [V0t8:x,sDEOsl0xsk1xzxMEFMFJM5xw1'5] [e7v316zu1,wMFMEGDM5xw1'5]	Tempo restante: 974 Valor: 278

Answer: (penalty regime: 0, 0, 10, 20, ... %)

Reset answer

```
1 class Deque:
```