**Programa IT Academy – Processo Seletivo – Edição #18**

**Nome Completo: Eduardo Dalmás Faé**

**E-mail: eduardofaeef@gmail.com**

# **Etapa 1 – Questões de lógica**

Esta seleção possui 15 questões de lógica de caráter eliminatório. As questões são apresentadas no formulário de Exercício Técnico e devem ser respondidas no próprio formulário online, que deverá ser acessado através do link a seguir: https://forms.gle/yZtVcv1b5fCgScLBA

# **Etapa 2**

**RESUMO DA SOLUÇÃO**

A minha solução foi feita inteiramente em JS, e fez uso da biblioteca papaparse para fazer o parse do CSV e criar uma matriz de distâncias.

Essa matriz de distância tem sua primeira linha preenchida pelos nomes das cidades, dessa forma, para acessá-la, basta buscar na primeira linha os índices das cidades, e fazer uso desses índices para encontrar na matriz qual a distância entre as cidades. Outrossim, vale ressaltar que, por a matriz ter sua primeira linha preenchida pelo nome das cidades, ao acessar a linha da matriz deve-se somar 1 no índice procurado, para compensar a linha gasta.

Após o fim do parse, é chamada a função consultas que gere todas as consultas, ela imprime na tela um menu para o usuário e espera um input para decidir que função deve chamar.

Existem 3 funções principais:

- parseTrechos: implementa a funcionalidade 1

- parseTransportes: implementa a funcionalidade 2

- historico: implementa a funcionalidade 3 (apenas a parte de impressão)

Além disso existem funções internas:

- menu: imprime o menu no console.

- printCities: imprime a lista de cidades.

- getCities: recebe do usuário e valida a lista de cidades.

- getDistances: calcula um array com a distância entre cada trecho de uma viagem.

- getItens: recebe uma lista de itens e checa se o padrão da entrada está correto.

- getItensStrign: recebe a lista de itens e concatena a string de saída.

- getNumTrucks: calcula uma matriz com a quantidade de cada tipo de caminhão para cada trecho de uma dada viagem.

- getNumByTruckType: recebe um peso e calcula a melhor distribuição de caminhões para aquele peso.

- getNewWeight: calcula o novo peso para o novo trecho de uma viagem.

- getTruckString: recebe a lista de caminhões usados e concatena a string de saída.

- getValue: calcula o valor da corrida dado os caminhões utilizados e as distâncias percorridas.

- getMeanValue: calcula o valor unitário médio de uma viagem.

Estruturas criadas:

- Truck: Gerencia um tipo de caminhão - armazena tipo, custo, peso e quantidade.

- Item: Gerencia um item - armazena nome, peso e quantidade.

- TruckList: Criada unicamente para facilitar entendimento em parseTrechos, o uso das funções include e getCost.

Observações finais:

Por me faltar tempo e para facilitar a implementação, não levei em consideração como cada item poderia ser subdividido, pois, para calcular o uso de caminhões utilizei unicamente o peso total de todos os itens. Isso é, em minha implementação todos os itens podem ser subdivididos. Fiz isso pois estava relativamente sem tempo e, ao analisar o problema da subdivisão ótima, percebi que ele possuía semelhanças com o problema da mochila binária. E, sabendo que o problema da mochila binária é um problema NP-Completo, preferi considerar que todos os itens podem ser fracionados, para aproximar o problema ao da mochila fracionária, que é polinomial. Assim facilitando meu trabalho, afinal estava com pouco tempo.

Espero que entendam, e desde já agradeço a oportunidade de ter participado do processo seletivo. Ele, com certeza, foi bem divertido e desafiador de se implementar!

**TESTES (aqui você deverá colar capturas de tela de todas as funcionalidades desenvolvidas e realizar comentários, use o espaço que julgar necessário)**

**FUNCIONALIDADE 1 – Consultar trechos x modalidade**

[cole aqui a(s) captura(s) de tela mostrando a funcionalidade em ação]

[você pode incluir uma explicação ou detalhamento em texto]

**FUNCIONALIDADE 2 – Cadastrar transporte**

[cole aqui a(s) captura(s) de tela mostrando a funcionalidade em ação]

[você pode incluir uma explicação ou detalhamento em texto]

**FUNCIONALIDADE 3 – Dados estatísticos**

[cole aqui a(s) captura(s) de tela mostrando a funcionalidade em ação]

[você pode incluir uma explicação ou detalhamento em texto]

**AUTOAVALIAÇÃO**

Você concluiu a implementação de 100% das funcionalidades solicitadas?

(X) Sim ( ) Não

Para as 3 principais funcionalidades solicitadas, como você avalia a sua solução?

Marque um ‘X’.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Inexistente/ Insuficiente | Pouco satisfeito(a) | Satisfeito(a) | Muito satisfeito(a) |
| Funcionalidade 1 |  |  |  | X |
| Funcionalidade 2 |  |  | X |  |
| Funcionalidade 3 |  |  |  | X |

**Principais dificuldades**

A principal dificuldade que tive foi, de longe, o tempo. Infelizmente a data do processo seletivo convergiu justamente na semana em que já possuía uma viagem programada. E, por isso, não pude mexer no programa por 4 dias, o que significa que infelizmente perdi mais da metade do tempo que poderia ter dedicado ao cumprimento da tarefa. Além disso, tive alguns problemas com a sintaxe do JS, pois já fazia um tempo que não trabalhava puramente com ele, já que atualmente ando aprendendo C++.

**Desempenho Geral**

Acredito ter conseguido desempenhar bem no geral, apesar dos contratempos com a viagem. Contudo gostaria de ter tido mais tempo para automatizar os testes e desenvolver uma interface melhor. Contudo, estou relativamente satisfeito quanto às funcionalidades, já que todas foram implementadas sem grandes problemas.

Obrigado por participar deste processo seletivo.  
Salve o documento em PDF com o seu nome completo.