**Projeto C204 – L2**

Equipe:

- Felipe Silveira

- Gabriel Almeida

- Natacha Calixto

Sumário

[Introdução 3](#_Toc101212253)

[Funcionamento do algoritmo 3](#_Toc101212254)

[Análise de complexidade assintótica 4](#_Toc101212255)

[Exemplos de entrada e saída 4](#_Toc101212256)

[Testes e resultados 7](#_Toc101212257)

[Conclusão 7](#_Toc101212258)

[Referências 7](#_Toc101212259)

# Introdução

O nosso projeto foi desenvolvido do zero usando algumas lógicas que fomos inventando ao longo do desenvolvimento. O projeto foi criado a fim de resolver problemas relacionados à logística de produtos em um supermercado, calculando dinamicamente a melhor rota de entrega e qual motoboy deve fazer a entrega do pacote de acordo com seu destino. Criamos a nossa própria lógica para a escolha dos itens da mochila, o Dijsktra para o cálculo do menor custo com o caminho que passe pelas cidades que precisam e o QuickSort para ordenar o vetor de motoboys por ordem do mais perto do supermercado, que tem uma maior prioridade.

# Funcionamento do algoritmo

Desenvolvemos uma lógica que usamos uma struct de motoboys e com a entrada da distância até o supermercado e a identificação de cada motoboy, a função quicksort irá ordenar dando prioridade de escolha para os itens para o entregador mais próximo. Assim, os itens escolhidos serão os itens que cabem na mochila à medida que o vetor de itens vai sendo passado. Com os itens escolhidos para todos os motoboys, a função Dijkstra irá ser chamada a fim de calcular o caminho com o menor custo que passe pelas casas onde serão entregues as compras.

# Análise de complexidade assintótica

Após análise, verificamos que a complexidade do código é O(n²), por conta dos vários for/while dentro de outro for/while.

# Exemplos de entrada e saída

O algoritmo foi testado com os seguintes exemplos de entrada e saída:

**Teste 1:**

**Entrada:**

6

5

1 2 1

1 3 5

1 4 7

1 5 4

1 6 9

2 3 14

2 4 6

2 5 8

2 6 11

3 4 8

3 5 10

3 6 10

4 5 12

4 6 4

6 5 8

-1 -1 -1

4

1 10

2 9

3 9

4 3

5

18

7 5 10 9 4

**Saída:**

Motoboy 4 pegou o(s) item(ns): 1 2 6

Menor caminho: 5 1

Menor caminho: 1 2

Menor caminho: 2 1 6

Ele gastou 15 min

Motoboy 3 pegou o(s) item(ns): 3

Menor caminho: 5 1 3

Ele gastou 9 min

Motoboy 2 pegou o(s) item(ns): 4

Menor caminho: 5 1 4

Ele gastou 11 min

Motoboy 1 nao pegou nenhum item

Nao realizou entregas

Nao teve tempo de entrega

**Teste 2:**

**Entrada:**

4

3

1 2 4

1 3 3

1 4 1

2 3 2

2 4 7

3 4 5

-1 -1 -1

2

1 2

2 3

3

10

5 9 3

**Saída:**

Motoboy 1 pegou o(s) item(ns): 1 4

Menor caminho: 3 1

Menor caminho: 1 4

Ele gastou 6 min

Motoboy 2 pegou o(s) item(ns): 2

Menor caminho: 3 2

Ele gastou 5 min

# Testes e resultados

De acordo com a ferramenta de análise gProof, o código demora apenas 0.01 segundos para rodar. Mesmo com poucas variáveis, esse é um tempo excelente para um código dessa complexidade. Segue em anexo abaixo os logs do gProof:



# Conclusão

Pode-se verificar com o algoritmo tem uma complexidade O(n), logo é eficiente e funcional para o projeto proposto. Por conta de usar algoritmos muito versáteis, é perfeitamente possível utilizar o código para propostas semelhantes.

# Referências

FavTutor.com: [Dijkstra's Algorithm in C++ | Shortest Path Algorithm | FavTutor](https://favtutor.com/blogs/dijkstras-algorithm-cpp)