# PROJETO DE LOGÍSTICA COM APLICAÇÃO DA TEORIA DE GRAFOS

## ATIVIDADE DE PROVA PARA DISICIPLINA (COMPLEXIDADE DE ALGORÍTIMOS)

**DOCENTE: MARCIO ALEXANDRE GARRIDO**

**DISCENTES: DENIS COMES BOMFIM & MARCOS LUIZ DE SOUSA REIS**

## INTRODUÇÃO

Como engenheiros de software, pesquisamos e procuramos estruturar este projeto de logística com Teoria de Grafos de forma clara, performática e seguindo uma abordagem que analogicamente se alinha a um sistema profissional muito utilizado por grandes corporações. O SAP foi um sistema ERP (Enterprise Rasoure Planning) amplamente utilizado, especialmente por grandes corporações em diversos setores, devido à sua capacidade de integrar sistemas de várias áreas da empresa, como manufatura, finanças e recursos humanos.

No entanto, o SAP é um sistema legado que foi sucedido pelo SAP ECC (ERP Central Component) e, mais recentemente, pelo SAP S/4HANA, a solução atual e baseada em nuvem da SAP. Empresa que usam o sistema SAP: JBF, AMBEV, Coca-Cola. Utilizam o SAP (focando em Modularidade e Otimização) para a distribuição e controle dos armazenagens e custos.

O Sistema SAP possui módulos interconectados (como o MM - Materials Management e SD - Sales and Distribution) que gerenciam a logística. Nosso projeto simulará essa otimização de rotas.

Utilizaremos Python com as bibliotecas NetworkX para manipulação de grafos e Matplotlib para visualização, garantindo um código limpo e funcional.

A seguir para uma melhor organização documental estruturamos em Módulos a saber:

### Módulo I: Representação e Construção do Gráfico (Análogo ao Master Data/Configuração SAP)

Para a nossa sub-rede de transporte, escolhemos 5 cidades fictícias para adicionar mais robustez e cenários de rota.

- O Armazém Central de Distribuição (Origem): Alpha
- As Cidades/Clientes (Destinos): Beta, Gamma, Delta, Epsilon

Desta maneira assumiremos um grafo direcionado (estradas de mão única ou custos diferentes em direções opostas) e os custos baseados em tempo de viagem (em minutos).

**Módulo II: Cálculo de Caminho Mínimo Simples (Análogo à Otimização de Transporte)**

Nesta etapa, fizemos o cálculo do caminho de menor custo (Caminho Mínimo) entre a Origem (Alpha) e o destino Epsilon de forma simples/manual, como solicitado, sem utilizar algoritmos avançados (como Dijkstra ou A* nativos do NetworkX), mas simulando a lógica de acumulação de custos.

Origem: AlphaDestino: Epsilon

**Cálculo Simples das Rotas Possíveis**

**Rota 1: Alpha ➡ Beta ➡ Delta ➡ Epsilon.**

Custo: 20(Alpha ➡ Beta) + 30(Beta ➡ Delta) + 10(Delta ➡ Epsilon) = 60 min.

**Rota 2: Alpha ➡ Gamma ➡ Epsilon**

Custo: 50(Alpha ➡ Gamma) + 40(Gamma ➡ Epsilon) = 90 min.

**Rota 3: Alpha ➡ Beta ➡ Gamma ➡ Epsilon**

Custo: 20(Alpha ➡ Beta) + 25(Beta ➡ Gamma) + 40(Gamma ➡ Epsilon) = 85 min.

**Rota 4: Alpha ➡ Gamma ➡ Delta ➡ Epsilon.**

Custo: 50(Alpha ➡ Gamma) + 35(Gamma ➡ Delta) + 10(Delta ➡ Epsilon) = 95 min.

**Caminho de Menor Custo:**

A Rota 1 (Alpha ➡ Beta ➡ Delta ➡ Epsilon)
com custo de 60 minutos.

**Módulo III & IV: Caminhos Alternativos e Análise de Robustez (Análogo à Gestão de Exceções)**

- **Simulamos uma falha** em uma estrada crítica e analisamos o impacto na rede e quais alternativas permanecem viáveis.
- **Simulação de Falha**. Simulamos a falha da estrada Beta ➡ Delta, que fazia parte do nosso caminho mínimo (custo 30).

**Módulo V: Comparação de Resultados deste trabalho com o trabalho Logístico da Equipe "Luan Brandão (LB).**

### Contextualização

Elaboramos uma comparação do nosso trabalho com o do **Grupo LB**, que escolheu uma topologia de rede diferente.

**Figura 1 - Dataset de análise de Comparativa**

| Critério | MR/DG | LB |
|---|---|---|
| **Objetivo** | Sistema completo de análise logística com Dijkstra automático, simulação de falhas e plotagens comparativas. | Cálculo manual de caminhos, simulação de falhas e identificação de estradas críticas. |
| **Cálculo do caminho mínimo** | Automático usando `nx.dijkstra_path ()` e `nx.path_weight()`. | Manual, iterando sobre caminhos predefinidos (`caminhos possíveis`) e calculando custo via função personalizada. |
| **Simulação de falha** | Remove aresta do caminho ideal e recalcula automaticamente a rota alternativa com Dijkstra. | Remove aresta do caminho selecionado e recalcula manualmente a melhor rota disponível iterando sobre caminhos possíveis. |
| **Plotagem / Visualização** | Plotagens profissionais com: subplots lado a lado, cores diferenciadas, labels detalhados, tempo de execução, nó especial destacado ("Alpha"). | Plotagens simples, com destaque para caminho mínimo em vermelho, tempo e legenda simples adicionados fora do gráfico. |
| **Flexibilidade para grafos grandes** | Alta, pois usa Dijkstra automático; qualquer grafo direcionado ponderado funciona sem necessidade de gerar manualmente todos os caminhos. | Limitada, pois depende de uma lista manual de caminhos; não escalável para grafos grandes. |
| **Identificação de estradas críticas** | Não implementada explicitamente (mas poderia ser adaptada via análise de falhas). | Implementada: remove cada aresta e verifica se há caminho do início ao fim. |
| **Complexidade / Código** | Maior, mais modular, mais funções reutilizáveis (desenhar_subplot, desenhar_grafo_simples, custo_caminho). | Menor, mais direto, funções básicas (desenhar grafo, custo_caminho, existe_caminho). |
| **Tempo de execução** | Geralmente mais rápido em grafos maiores, pois usa algoritmo de Dijkstra interno (O (E + V log V)). | Mais lento em grafos maiores, pois testa todos os caminhos possíveis manualmente (O (P * L), P = nº de caminhos, L = comprimento do caminho). |
| **obustez** | Mais robusto, trata exceções do NetworkX (`NetworkXNoPath`). | Robustez limitada, depende do o usuário fornecer todos os caminhos possíveis corretamente. |

| Critério | MR/DG | LB |
|---|---|---|
| **Usabilidade / Apresentação** | Mais profissional, ideal para apresentações comparativas de rota antes/depois da falha. | Mais didático e simples, bom para estudo ou pequenos exemplos. |
| **Custo de manutenção** | Médio-alto: mais funções e modularidade exigem manutenção. | Baixo: código pequeno, mais fácil de entender e modificar para pequenos grafos. |

**Figura 2 - Resumo da Comparação da rede (MR/DG) com a rede (LB)**

| Código | Caminho inicial | Custo inicial | Tempo cálculo inicial | Caminho após falha | Custo após falha | Tempo cálculo após falha | Observações |
|---|---|---|---|---|---|---|---|
| MR/DG | Caminho ótimo por Dijkstra | Ex.: 60min | ~0.0001 s | Rota alternativa por Dijkstra | Ex.: 65min | ~0.0001 s | Robusto, automático, bom para grafos grandes, plotagens profissionais |
| LB | Melhor caminho manual da lista | Ex.: 6 | ~0.0002 s | Melhor caminho manual da lista sem falha | Ex.: 8 | ~0.0002 s | Simples, didático, limitado a grafos pequenos e caminhos pré-definidos, estradas críticas calculadas |

## CONCLUSÃO

Em um sistema profissional de logística, como o SAP ou sistemas de roteirização corporativos, o modelo MR/DG seria claramente preferível. Isso se deve à sua **robustez, escalabilidade e automação**, que garantem que o sistema consiga encontrar a melhor rota mesmo em grafos grandes ou diante de falhas imprevistas nas estradas. Nesses contextos, a **confiabilidade da entrega e a resiliência da rede logística** são geralmente mais importantes do que pequenas economias de tempo obtidas com um cálculo manual de rotas ideais. Além disso, a capacidade de gerar visualizações claras e análises comparativas de diferentes cenários aumenta a **tomada de decisão estratégica** para transporte e planejamento logístico.