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 R/3 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 R/3 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 R/3 (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 R/3)

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 (Análogo ao Benchmarking/Auditoria)

Contextualização

Uma comparação do nosso trabalho com um **Grupo Fictício B**, que escolheu uma topologia de rede diferente.

1. Topologia da Rede

Característica	Nosso Grupo (A)	Grupo Fictício (B)
Nº de Cidades	5	4
Nº de Estradas	8	5
Conectividade	Alta (Muitas alternativas)	Baixa (Topologia mais linear)
Origem/Destino	Alpha ➡ Epsilon	Hub ➡ Spoke

2. Rotas Mais Curtas e Custos

Cenário	Nosso Grupo (A)	Grupo Fictício (B)
Caminho Mínimo	60 min (Alpha ➡ Beta ➡ Delta ➡ Epsilon)	55 min (Hub ➡ City2 ➡ Spoke)

Cenário	Nosso Grupo (A)	Grupo Fictício (B)
Falha Crítica	Aumento para 85 min (+41.6%)	**Rota Inexistente** (Rede Desconectada)

Análise de Comparação

- **Variação das Rotas Mais Curtas:**
  - ✓ O Grupo B obteve um custo mínimo ligeiramente menor (55 min) devido à sua topologia mais linear (menos opções de desvio, rotas mais diretas).
  - ✓ O Grupo A tem um custo um pouco maior (60 min) inicial, mas com a vantagem de maior conectividade.
- **Impacto dos Custos de Transporte:**
  - ✓ Embora o Grupo B tenha custos nominais (por estrada) baixos, sua dependência de poucas rotas eleva o **Risco Total**. O custo total do transporte em A é mais robusto contra falhas.
- **Rotas Alternativas e Vantagem Competitiva (Robustez):**
  - Grupo A (Vantagem): A alta conectividade (8 arestas para 5 cidades) permite rotas alternativas em caso de falha. A entrega ainda ocorre, mesmo que com atraso (85 min).

    Prioriza a Confiabilidade.

  - Grupo B (Desvantagem): A baixa conectividade (5 arestas para 4 cidades) torna a rede extremamente frágil. A falha de uma única aresta crítica desconectaria a rede, impedindo a entrega.

    Prioriza a Rapidez em Cenário Ideal.

CONCLUSÃO

Em um sistema profissional de logística (como o SAP), o nosso modelo seria preferível, pois a **robustez e a garantia de entrega** (mesmo com atraso) são geralmente mais valiosas do que a economia marginal de tempo em uma rota ideal.

APENDICÊ A

Entregáveis: Códigos Python Completos (Para Google Colab)

Os códigos para a execução no Google Colab ou em qualquer ambiente Python com networkx e matplotlib instalados utilizados para fins de visualização gráficas das rotas de logística estão disponíveis. Para acesso utilize o arquivo anexo e ou link do google colab:

https://colab.research.google.com/drive/1eq8afpjgmTYMryTREAzfYp6oV3Tbj8fL?usp=sharing .

Disponível também no GirhHub

Manoreis/Projeto_Logistica_TeoriadeGrafos: Repositório de Atividades na Disciplina Complexidade de Algoritmos - Engenharia de Software

São eles:

Código Python:

1. **Módulo 1 -** Grafo direcionado

2. **Módulo 2 -** (Validação com Dijkstra): Para fins de código limpo e objetivo, incluímos o cálculo do algoritmo padrão de caminho mínimo (Dijkstra) do NetworkX para validar o resultado manual. (Em um ambiente profissional, essa seria a implementação padrão).

3. **Módulos 3 e 4 -** Código python de simulação de falhas.

4. **Módulos das tabelas**. Já inseridos no documento.