Relatório - ED02

Teoria dos Grafos

1. Introdução

Este trabalho implementa um sistema de navegação para a rede de estações espaciais descrita no estudo dirigido, utilizando conceitos de teoria dos grafos para resolver problemas de roteamento e análise de redes. O sistema foi desenvolvido em C e atende à missão de resgate da estação Centauri, encontrando rotas seguras e analisando a robustez da rede espacial.

2. Estruturas de Dados e Algoritmos

2.1 Estruturas de Dados Escolhidas

Grafo com Listas de Adjacência:

Justificativa da escolha

- **Eficiência em memória**: Ideal para grafos esparsos como a rede de estações
- **Flexibilidade**: Facilita adição e remoção dinâmica de rotas
- **Desempenho**: Acesso eficiente aos vizinhos de cada estação

2.2 Algoritmos Implementados

2.2.1 Algoritmo de Dijkstra

- **Propósito**: Encontrar o caminho mais curto considerando pesos das rotas
- **Complexidade**: $O(V^2)$
- **Adaptações**: Considera os pesos 1 (Large), 3 (Medium), 6 (Small)

2.2.2 Busca em Largura (BFS)

- **Propósito**: Verificar conectividade entre Terra e Centauri
- **Complexidade**: $O(V + E)$
- **Aplicação**: Garantia de existência de rotas alternativas

2.2.3 Análise de Redundância

- **Método**: Remoção sistemática de cada estação do caminho principal
- **Objetivo**: Identificar pontos críticos da rede
- **Implementação**: Teste de conectividade pós-remoção

3. Implementação

3.1 Carregamento e Inicialização

O sistema carrega automaticamente a rede a partir do arquivo CSV, processando cada linha para construir as listas de adjacência. A função `carregar_csv()` realiza o parsing do arquivo e chama `adicionar_aresta()` para cada conexão.

3.2 Gestão de Rotas Bloqueadas

As rotas bloqueadas especificadas no PDF são automaticamente removidas durante a inicialização:

3.3 Sistema de Pesos

O sistema considera a hierarquia de rotas:

- **Rotas Large**: peso 1 (alta capacidade)
- **Rotas Medium**: peso 3 (capacidade média)
- **Rotas Small**: peso 6 (baixa capacidade)

3.4 Medição de Desempenho

Implementação completa de medição de tempo usando `time.h`

4. Resultados e Análise

4.1 Caminho Mais Curto Encontrado

Rota Ótima:

`Terra → Hadrian → Oya → Castra → Nyx → Bremen → Vega → Nul → Centauri`

Características:

- **Custo total**: 10
- **Número de estações**: 8
- **Composição**: Combina rotas Large e Medium de forma eficiente

4.2 Análise de Conectividade

Resultado: **REDE CONECTADA**

A verificação com BFS confirmou que Terra e Centauri permanecem conectadas mesmo após a remoção das rotas bloqueadas, garantindo que a missão de resgate é viável.

4.3 Análise de Redundância

Resultado surpreendente: Todas as 7 estações intermediárias do caminho principal são **reduntantes.**

Interpretação: A rede espacial possui excelente tolerância a falhas, com múltiplos caminhos alternativos para cada segmento do percurso principal.

4.4 Desempenho Computacional

Tempos de Execução:

- **Carregamento do CSV**: 6.0000 ms
- **Algoritmo de Dijkstra**: 0.0000 ms
- **Verificação de Conectividade**: 0.0000 ms
- **Redundâncias**: 2.0000 ms
- **Tempo Total**: 9.0000 ms

Análise de Desempenho: Os algoritmos demonstraram alta eficiência, com processamento praticamente instantâneo para a maioria das operações.

5. Conclusões

5.1 Conclusões Técnicas

1. **Eficácia dos Algoritimos**: Tanto Dijkstra quanto BFS demonstraram ser adequados para os problemas de roteamento e conectividade em redes espaciais.

2. **Robustez da Rede**: A análise de redundância revelou uma rede bem dimensionada, com múltiplos caminhos alternativos entre Terra e Centauri.
3. **Desempenho**: A implementação com listas de adjacência mostrou-se eficiente tanto em uso de memória quanto em tempo de processamento.

5.2 Implicações para a Missão

1. **Viabilidade do Resgate**: A existência de caminhos seguros e redundantes torna a missão de resgate possível.
2. **Plano de Contingência**: A redundância identificada permite flexibilidade no planejamento de rotas alternativas em caso de falhas adicionais.
3. **Confiança Operacional**: A "forca"da rede da rede fornece margem de segurança adicional para operações críticas.

6. Instruções de Compilação e Execução

6.1 Requisitos

- Compilador C (GCC recomendado)
- Arquivo `arquivo.csv` na mesma pasta do executável

6.2 Compilação

```
gcc -o main main.c
```

6.3 Execução

```
./main
```