Trabalho Final — Análise Comparativa: Kruskal vs Prim em Grafos Espalhados e Densos

Objetivo Geral

Comparar os algoritmos de árvore geradora mínima (AGM) de Kruskal e Prim quanto ao desempenho e aplicabilidade em grafos com características estruturais diferentes (esparsos e densos), com base em testes práticos e fundamentação teórica.

1. **Parte Teórica (Pesquisa e Fundamentação)**

Cada grupo deverá realizar uma revisão teórica abordando:

- Conceito de Árvore Geradora Mínima (AGM) e propriedades fundamentais.
- Diferenças conceituais e operacionais entre os algoritmos de Kruskal e Prim.
- Estruturas de dados associadas (conjuntos disjuntos, min-heaps, listas de adjacência e matrizes).
- Complexidade computacional dos algoritmos
- Cenários ideais para uso de cada algoritmo (Kruskal para grafos esparsos; Prim mais vantajoso em grafos densos com estruturas apropriadas).
- Aplicações reais de AGMs: redes de computadores, redes de energia elétrica, redes de distribuição de água, redes de transporte etc.

2. **Parte Prática (Implementação e Testes)**

Requisitos:

- Implementar ambos os algoritmos (Kruskal e Prim) em linguagem C.
- Utilizar os mesmos conjuntos de grafos para comparar os dois algoritmos.
- Medir: tempo de execução, número de operações (se possível), uso de memória, e verificar a corretude da árvore geradora gerada.

Tipos de grafos sugeridos:

- Grafo Esparso Aleatório Exemplo: 1000 vértices e ~1500 arestas
  - Simula redes físicas com poucas conexões (ex: sensores, redes rurais)
  - Grafo Denso Aleatório Exemplo: 500 vértices e ~100.000 arestas
- Simula redes com alta conectividade (ex: redes de centros urbanos)
  - Grade 2D (malha) Vértices conectados como em uma matriz $m \times n$
  - Aplicável a redes elétricas ou de tubulações
- Grafo Real Pequeno (opcional)
  - Exemplo: mapa de uma rede de energia ou ruas de um bairro

3. Análises Esperadas:

- Comparar os resultados obtidos entre Kruskal e Prim nos diferentes cenários.
- Refletir sobre o custo-benefício de usar uma estrutura de dados mais sofisticada (heap, union-find).
- Apresentar tabelas e gráficos com tempo de execução e número de arestas analisadas.

Entrega

- Relatório completo com estrutura acadêmica padrão (introdução, teoria, metodologia, implementação, resultados, conclusão e referências).
- Código-fonte em C, comentado.
- Arquivos de entrada (grafos) utilizados nos testes.
- Os dados de desempenho devem estar organizados e bem apresentados.
-

Comentários gerais

- O trabalho é em grupo de cinco alunos;
- Trabalhos copiados (FONTE) terão nota zero.
- Trabalhos entregues em atraso serão aceitos, todavia a nota atribuída ao trabalho será diminuída em 10%;

Critérios de Avaliação:

1. Atendimento aos requisitos.
2. Qualidade do código (estruturação e comentário).

Data da entrega: 17/07/2025 (impreterivelmente)

Sugestões Adicionais

- Uso de bibliotecas como time.h e stdlib.h para medir desempenho.
- Pode-se gerar grafos com scripts em Python ou outra linguagem e apenas carregar os dados no programa C.
- Os alunos podem usar matrizes de adjacência ou listas, mas devem comentar os impactos da escolha no desempenho.