Trabalho Prático - 2026.1

Estudo Dirigido em Grafos — Seminário de Algoritmos

Estruturas de Dados — Bacharelado em Sistemas de Informação — Instituto Federal da Bahia

Entrega do material escrito: 30/08/2026

Apresentação Oral: 03/09/2026

Disciplina	Estruturas de Dados (ESP412)
Professora	Ana Carolina Sokolonski
Natureza	Estudo dirigido (pesquisa) + implementação + seminário
Linguagem	C (C99 ou superior) — **um único arquivo `.c`**
Modalidade	Em dupla (máximo de 2 estudantes — NÃO existe dupla de 3!) se quiser, pode ser individual
Valor	10,0 pontos
Entrega	Classroom
Apresentação	Seminário para a turma
Presença	A presença nos seminários dos colegas conta ponto

Do que se trata este trabalho

Diferentemente de um trabalho apenas de programação, aqui o **estudo** é o centro. Cada dupla receberá um **tema de grafos** e deverá *pesquisar*, *compreender a fundo* e *ensinar* esse tema, apoiando-se em uma pequena implementação própria. A nota valoriza fortemente a qualidade da investigação e do texto técnico, e não apenas o código.

Você entregará três coisas: (i) uma **monografia técnica curta**, (ii) um **programa em C — um único arquivo `.c`** — e (iii) os **slides** do seminário. Haverá ainda o **seminário** para a turma.

Postura de pesquisa

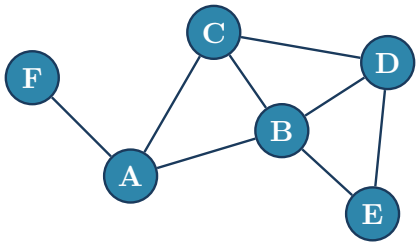
Pesquisar não é copiar a primeira página encontrada. Consulte **mais de uma fonte** (livro, artigo, notas de aula), compare as descrições, e explique o tema **com as suas próprias palavras**. Toda afirmação relevante (uma complexidade, um teorema, uma aplicação) deve vir acompanhada da fonte de onde você a obteve. Anote as referências desde o primeiro dia.

CONTEXTO

Por que estudar grafos a fundo

Grafos são uma das estruturas mais versáteis da computação: modelam redes de transporte, dependências entre tarefas, relações em redes sociais, circuitos, a Web, e muito mais. Sobre a mesma estrutura $G = (V, E)$ existe uma enorme família de problemas e algoritmos — busca, conectividade, caminhos mínimos, árvores geradoras, fluxo, emparelhamento, centralidade — cada um com sua teoria, sua análise de complexidade e suas estruturas de dados de apoio (filas, filas de prioridade, *union-find*, etc.).

Neste trabalho, cada dupla mergulha em **um** desses temas: estuda a teoria, entende *por que* o algoritmo funciona (e não só *como* programá-lo), analisa seu custo, implementa um pequeno demonstrador e o testa em exemplos. Ao final, apresenta o tema à turma. O conjunto de seminários constitui um panorama dos algoritmos clássicos de grafos, elaborado pela própria turma.



Uma mesma estrutura $G = (V, E)$ dá origem a dezenas de problemas distintos — este trabalho estuda um deles em profundidade.

OBJETIVOS

- desenvolver a capacidade de **pesquisar** e ler fontes técnicas;
- compreender a fundo um algoritmo em grafos: ideia, **corretude** e **complexidade**;
- relacionar o algoritmo às **estruturas de dados** que o sustentam;
- implementar e testar o algoritmo em **exemplos** pequenos;
- comunicar o conhecimento com clareza (texto técnico e seminário);
- defender individualmente as decisões e o entendimento do tema.

TEMAS

cada dupla recebe um; referência inicial entre colchetes

1. **Busca em largura e profundidade (BFS/DFS) e aplicações** — componentes conexas, detecção de ciclo, ordenação topológica. *[CLRS cap. 22]*
2. **Caminhos mínimos com pesos não negativos: Dijkstra** — e o papel da fila de prioridade (vetor vs. *heap* binário vs. Fibonacci). *[CLRS 24.3]*
3. **Caminhos mínimos com pesos negativos: Bellman–Ford** — e a detecção de ciclos negativos. *[CLRS 24.1]*

4. **Todos os pares de caminhos mínimos: Floyd–Warshall** — programação dinâmica sobre grafos. *[CLRS 25.2]*
5. **Árvore geradora mínima: Prim e Kruskal** — e a estrutura *union-find* (conjuntos disjuntos) em Kruskal. *[CLRS cap. 23; 21]*
6. **Componentes fortemente conexas: Tarjan ou Kosaraju.** *[CLRS 22.5]*
7. **Fluxo máximo e corte mínimo: Ford–Fulkerson / Edmonds–Karp** — e o teorema *max-flow min-cut*. *[CLRS cap. 26]*
8. **Emparelhamento em grafos bipartidos** — via fluxo ou Hopcroft–Karp; aplicações de alocação. *[CLRS 26.3]*
9. **Ordenação topológica e caminho crítico (CPM)** em DAGs — escalonamento de tarefas. *[CLRS 22.4; 24.2]*
10. **Centralidade e importância em redes: PageRank** ou centralidade de intermediação. *[Newman,* Networks*]*

A distribuição dos temas será feita em aula (sem repetição entre duplas). A referência entre colchetes é apenas o *ponto de partida* — espera-se que você encontre outras.

PARTE 1 — PESQUISA E MONOGRAFIA peso principal da nota

Produza uma **monografia técnica** de **4 a 6 páginas** (fora capa e referências) contendo:

- **Introdução e motivação:** o que o problema resolve e onde aparece na prática (com exemplos).
- **Definições:** os conceitos de grafos necessários ao tema, com notação clara.
- **O algoritmo:** descrição em pseudocódigo e explicação passo a passo em um pequeno exemplo feito à mão.
- **Corretude:** *por que* o algoritmo produz a resposta certa — a intuição ou o argumento (invariante, propriedade de subestrutura ótima, etc.), com as suas palavras.
- **Complexidade:** análise de tempo e de espaço, **justificada** (não apenas o resultado final $O(\cdot)$), destacando como a escolha da estrutura de dados afeta o custo.
- **Revisão bibliográfica:** pelo menos **três referências**, sendo ao menos uma *acadêmica* (livro ou artigo). Compare como duas fontes diferentes apresentam o algoritmo e comente as diferenças.
- **Referências** no final, em formato padronizado (ABNT ou similar).

PARTE 2 — IMPLEMENTAÇÃO (UM ÚNICO ARQUIVO `.c`)

Escreva um **único arquivo `.c`** que demonstre o algoritmo estudado. Requisitos:

- compilar com **um único comando** (ex.: `cc trabalho.c -o trabalho` ou `gcc trabalho.c -o trabalho`), usando apenas a biblioteca padrão;
- representar o grafo com uma estrutura adequada (lista de adjacência é a recomendada; matriz de adjacência é aceitável em grafos pequenos, desde que você justifique);
- executar o algoritmo do tema e exibir a **resposta** de forma clara (o caminho, a árvore, a ordem topológica, o valor do fluxo, etc.);
- incluir um **contador** pertinente ao tema (número de visitas, relaxamentos, iterações, aumentos de fluxo...) e imprimi-lo;
- demonstrar em **pelo menos dois exemplos pequenos** definidos por você — os grafos podem ser lidos de um arquivo texto simples ou ficar embutidos no próprio código; mostre a entrada e a saída;
- código **organizado e comentado**, com um cabeçalho no início do arquivo contendo o nome da dupla, o tema e as instruções de compilação e uso.

Não é necessário `Makefile` nem múltiplos arquivos: um só `.c` bem organizado basta. Se usar alocação dinâmica, libere a memória; se preferir vetores de tamanho fixo, documente os limites adotados.

Fontes sugeridas (ponto de partida)

- Cormen, Leiserson, Rivest, Stein, *Introduction to Algorithms* (CLRS);
- Sedgewick & Wayne, *Algorithms*;
- Ziviani, *Projeto de Algoritmos*;
- Procure também outras fontes por conta própria e cite todas na monografia.

ENTREGA

Um único `.zip` (nomeado com os nomes da dupla) contendo:

- `monografia.pdf` (4–6 páginas + referências);
- `trabalho.c` (o arquivo único, comentado);
- `slides.pdf` (os slides do seminário);

SEMINÁRIO

O seminário (10–15 min) apresenta o tema à turma: a ideia do algoritmo, um exemplo executado ao vivo (feito pela dupla) e a análise de complexidade.

A dupla deve explicar bem o tema e o código que fez. Qualquer integrante poderá ser solicitado a: executar o algoritmo à mão em uma instância dada; explicar a corretude ou a complexidade; apontar no código onde cada etapa ocorre; discutir uma fonte citada; ou fazer uma pequena modificação (trocar a estrutura usada, adaptar para um grafo não orientado, imprimir o caminho, etc.). A nota individual depende dessa demonstração de domínio.

O QUE ENTREGAR (RESUMO)

```
Trabalho/
├── monografia.pdf
├── trabalho.c
└── slides.pdf
```

CRITÉRIOS DE AVALIAÇÃO

Critério	Valor
Revisão bibliográfica e uso de referências	1,0 pt
Presença em TODAS as apresentações	1,0 pt
Monografia técnica (teoria, corretude, complexidade)	2,0 pts
Implementação (arquivo `.c`)	2,0 pts
Seminário (apresentação à turma)	4,0 pts
Total	**10,0 pts**

Uso de ferramentas de IA

Ferramentas de IA podem apoiar a busca por fontes, a revisão do texto e a depuração do código. Contudo, a **compreensão** e a **autoria** devem ser da dupla. Todo conteúdo técnico deve ser sustentado por fontes verificáveis e explicado com as suas próprias palavras; o texto e a implementação devem ser seus. No seminário e na defesa, qualquer integrante poderá ser solicitado a explicar, deduzir ou modificar o que foi apresentado — e é isso que determina a nota individual.

Pesquise com curiosidade, leia mais de uma fonte, e busque entender o *porquê* antes do *como*. Ensinar um tema é a melhor forma de aprendê-lo — prepare-se para ensiná-lo à turma. Bom estudo e boa pesquisa!