



PLANO DE ENSINO

1. IDENTIFICAÇÃO

Componente Curricular:	Estruturas de Dados II						
Unidade Ofertante:	Faculdade de Computação						
Código:	FACOM39401	Período/Série:	4	Turma:	I		
Carga Horária:				Natureza:			
Teórica:	30	Prática:	30	Total:	60	Obrigatória()	Optativa()
Professor(A):	Maria Adriana Vidigal de Lima				Ano/Semestre:		
Observações:							

2. EMENTA

Árvores: Tipos, Árvore de Busca Binária (ABB), Balanceamento de ABB. Grafos. Hashing.

3. JUSTIFICATIVA

Estruturas de dados permitem armazenar, com propriedades diferentes para cada modelo, as informações lidas pelo programa. Uma estrutura de dados é definida por uma coleção de valores, as relações entre os dados e o conjunto de operações que podem ser aplicadas sobre cada dado. Conhecer essas estruturas e entender como funcionam é fundamental para que o aluno possa construir soluções computacionais eficientes.

4. OBJETIVO

Objetivo Geral:

Capacitar o aluno a desenvolver soluções computacionais eficientes para problemas que necessitam de algoritmos e/ou estruturas de dados avançadas.

Objetivos Específicos:

- Discutir o custo computacional de algoritmos.
- Utilizar adequadamente solução iterativa e recursiva.
- Utilizar árvores, grafos e tabelas hash.

5. PROGRAMA

1. Árvores

1.1. Tipos de árvores e representação.

1.2. Árvore binária.

1.2.1. Tipo Abstrato de Dados.

1.2.2. Árvore de sintaxe abstrata: travessia em-ordem, pré-ordem e pós-ordem.

1.2.3. Atributos: Nível, Altura, Comprimento de Caminho.

1.2.4. Algoritmos (cálculo da altura e comprimento de caminho) e operações básicas.

1.2.5. Implementação de TAD Árvore Binária em Linguagem Python.

1.3. Árvore Binária de Busca.

1.3.1. Árvore AVL.

1.3.2. Fator de Balanceamento.

1.3.3. Operações: Inserção/Remoção de nós e implementação em Linguagem Python.

1.3.4. Análise e custo dos algoritmos.

2. Grafo

2.1. Definição e aplicações

2.2. Grau de vértice, adjacência, incidência, grau de um vértice, caminho, ciclo, densidade, grafo conectado, grafo completo.

2.3. Influência da densidade do grafo em algoritmos.

2.4. Tipo Abstrato de Dados.

2.4.1. Operações básicas: criação/destruição de um grafo, inserção/remoção de vértices e arestas, apresentação de vértices e arestas, apresentação do número de vértices, cópia de um grafo.

2.5. Representações.

2.5.1. Matriz de Adjacência.

2.5.2. Lista de Adjacência.

2.5.3. Comparação entre Matriz X Lista: Espaço e Tempo.

2.5.4. Implementação de TAD Grafo em Python.

2.6. Caminhamentos em grafos: largura e profundidade.

2.7. Menor caminho entre dois vértices.

3. Hash

3.1. O que é Hash.

3.2. Funções de Hashing.

3.2.1. Chaves: número real, número inteiro, string.

3.3. Tratamento de Colisão.

3.3.1. Lista de colisões.

3.3.2. Sondagem Linear.

3.3.3. Duplo Hash

6. METODOLOGIA

Aulas expositivas, trabalhos dirigidos, trabalhos práticos. O conteúdo programático da disciplina será disponibilizado por meio da plataforma Moodle (www.moodle.ufu.br), sendo necessária a auto-inscrição do aluno por meio de chave de inscrição. O link de acesso à disciplina e a chave de inscrição serão fornecidos na primeira semana de aula pelo professor. São disponibilizadas 2 horas semanais de atendimento para esclarecimento de dúvidas, revisão de conteúdo e resolução de exercícios.

Tabela 1 - CRONOGRAMA DAS ATIVIDADES

Semana	Datas	Atividades Previstas	Carga-Horária (hora/aula)
1	08/01 e 10/01	Apresentação da disciplina, e das atividades a serem realizadas. Introdução às árvores. Prática: Revisão em Python e ordenação.	4h/a

2	15/01 e 17/01	Tipos de árvores e representação. Árvore binária. Prática: Revisão em TAD	4h/a
3	22/01 e 24/01	Árvore de sintaxe abstrata: travessia em-ordem, pré-ordem e pós-ordem. Prática: TAD Árvore Binária - Estrutura e operação de criação da árvore.	4h/a
4	29/01 e 31/01	Atributos: Nível, Altura, Comprimento de Caminho. Algoritmos (cálculo da altura e comprimento de caminho) e operações básicas. Prática: TAD Árvore Binária - operações básicas.	4h/a
5	05/02 e 07/02	Árvore Binária de Busca. Prática: TAD Árvore Binária de Busca - inserção, busca e remoção.	4h/a
6	19/02 e 21/02	Árvore AVL e Balanceamento. Prática: Árvore AVL. Fator de Balanceamento. Implementação em Python.	4h/a
7	26/02 e 28/02	Prova 1 - 35 pontos. Prática: Árvore AVL. Operações de Inserção e Busca. Implementação em Python.	4h/a
8	04/03 e TDE	Árvore AVL e operação de remoção. Análise e custo dos algoritmos. TDE: Leitura sobre Estruturas de dados e Aplicações.	4h/a
9	06/03 e TDE	Prática: Introdução aos Grafos. TAD Grafo. TDE: Leitura sobre Grafos e aplicações.	4h/a
10	11/03 e 13/03	Grafo: Grau de um vértice, adjacência, incidência. Conceito de caminho, ciclo, densidade, grafo conectado, grafo completo. Influência da densidade do grafo em algoritmos. VEM PRA UFU - 13/03.	4h/a
11	18/03 e 20/03	Grafo: criação/destruição de um grafo, inserção/remoção de vértices e arestas, apresentação de vértices e arestas, apresentação do número de vértices, cópia de um grafo. Prática: TAD Grafo: operações de manipulação.	4h/a
12	25/03 e 27/03	Grafo: Comparação entre Matriz X Lista: Espaço e Tempo. Caminhamentos em grafos: largura e profundidade. Prática: TAD Grafo - caminhamento.	4h/a
13	TDE	Leitura sobre grafo e aplicações de caminho de custo mínimo.	4h/a
14	01/04 e 03/04	Grafo: Menor caminho entre dois vértices. Prática: Introdução à hash - espalhamento. Conceitos e Propriedades. Funções Hash.	4h/a
15	08/04 e 10/04	Hash: Colisão com encadeamento separado. Prática: Colisão com endereçamento aberto. Duplo hash.	4h/a
16	15/04 e TDE	Prova 2 - 35 pontos. TDE: Leitura sobre aplicações com Tabela Hash.	4h/a
17	17/04 e TDE	Vista de Prova. Entrega e apresentação de trabalhos. TDE: Prática de Recuperação e Lista de Exercícios.	4h/a
18	22/04 e 24/04	Prova de Recuperação (70 pontos). Vista da Recuperação.	4h/a
		Total de horas-aula em TDE: 12h/a Total de horas-aula presenciais: 60h/a	72h/a

7. AVALIAÇÃO

As atividades avaliativas individuais serão realizadas na forma de questões de múltipla escolha e abertas, totalizando 70 pontos:

1ª Prova individual - valor: 35 pontos - Data: **26/02/2024**

2ª Prova individual - valor: 35 pontos - Data: **15/04/2024**

As atividades avaliativas em grupo (trabalhos) totalizam 30 pontos. Serão realizados três trabalhos de 10 pontos, um para cada módulo apresentado: Árvore, Grafo e Hash. Os grupos poderão ter até 2 componentes. A recuperação de aprendizagem será feita por meio de atendimento em horário extra-classe através de exercícios e material complementar.

Será oferecida uma prova de recuperação, no valor de 70 pontos, para substituir a nota das avaliações de alunos que não obtiveram média de 60 pontos no total do semestre, e que não estiverem reprovados por frequência.

Prova de recuperação - valor: 70 pontos - Data: **22/04/2024**

8. BIBLIOGRAFIA

Básica

1. CORMEN, T. H. Algoritmos: Teoria e Prática. Rio de Janeiro: Campus, 2002.
2. GOODRICH, M.T.; TAMASSIA, R.; GOLDWASSER, M.H. Data structures and algorithms in Python. John Wiley & Sons Ltd, 2013.
3. SEDGEWICK, R. Algorithms. 4th ed. Upper Saddle River: Addison-Wesley, c2011.

Complementar

1. BOAVENTURA NETTO, P. O. Grafos: teoria, modelos, algoritmos. 4. ed. rev. ampl São Paulo: E. Blucher, 2006.
2. DROZDEK, A. Estruturas de dados e algoritmos em C++. São Paulo: Cengage Learning, 2002.
3. SEDGEWICK, R. Algorithms in C. 3rd. ed. Boston: Addison-Wesley, 2002.
4. TENENBAUM, A.M.; LANGSAM, Y.; AUGENSTEIN, M.J. Estrutura de Dados Usando C. São Paulo: Makron Books, 1995.
5. ZIVIANI, N. Projeto de Algoritmos. 2.ed. São Paulo: Pioneira , 2004.

9. APROVAÇÃO

Aprovado em reunião do Colegiado realizada em: ___/___/___

Coordenação do Curso de Graduação: _____



Documento assinado eletronicamente por **Maria Adriana Vidigal de Lima, Professor(a) do Magistério Superior**, em 17/01/2024, às 09:20, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5106447** e o código CRC **E5662CA3**.