Instituto de Computação/Unicamp - Estruturas de Dados - MC202EF

Prof. Lehilton Lelis Chaves Pedrosa

Programa de Ensino

Objetivos

Ao final do curso, o aluno deverá ser capaz de:

- desenvolver e testar programas usando estruturas de dados conhecidas: listas, árvores, grafos e generalizações;
- escolher as estruturas de dados mais adequadas e eficientes para problemas.

Atividades

Durante o semestre serão realizadas diversas atividades. As sublinhadas serão usadas com avaliação; as outras também são essenciais para o aprendizado. Em cada uma das atividades, o aluno deverá **participar** ativamente.

Exercício pré-aula: em unidades específicas, serão sugeridas leituras ou pesquisas, que deverão ser realizadas pelo aluno **antes da aula.**

Aula expositiva: durante as aulas, um ou mais problemas serão apresentados aos alunos e um ou mais alunos escolhidos pelo professor deverão sugerir soluções **oralmente** (ou indicar as dificuldades).

Exercícios de fixação: após as aulas, serão propostos exercícios de fixação para serem feitos em sala e/ou em casa; os alunos deverão levantar as principais dúvidas no **início** das aulas seguintes.

Teste individual (avaliação): serão realizados 5 testes **individuais e sem consulta** com duração de 30 a 40min, no horário das aulas, em datas a serem divulgadas na página da disciplina. A nota de cada teste será de 0 a 10.

Trabalho em dupla (avaliação): serão realizados 5 trabalhos a serem feitos em dupla na sala e/ou em casa com prazo definido. *Os integrantes da dupla não podem ser repetidos.* A nota do trabalho será de 0 a 10. A dinâmica será:

1- o trabalho é entregue até o prazo, **valendo 10 pontos**: deverá ser entregue em PDF pelo sistema *run.codes* por *exatamente* um aluno da dupla e poderá ser um texto digitado ou um manuscrito digitalizado; para ser aceito, o documento deve estar legível, sem rasuras, devidamente identificado e organizado;
2- monitor entrega correção e *feedback* do trabalho;
3- a dupla que desejar, poderá ressubmeter o trabalho, desde que realizadas as correções sugeridas no *feedback*; a nota poderá aumentar, **valendo até 7 pontos**.

Exercícios de laboratório *(avaliação)*: serão realizados 10 trabalhos de programação com prazos compatíveis com o nível de dificuldade. A nota será a soma de duas partes:

* 7 pontos proporcionais aos acertos de casos de teste;
* 3 pontos de critérios específicos relativos ao algoritmo e à qualidade de código.

A dinâmica e correção será:

1- aluno entrega trabalho até prazo, **valendo 10 pontos**, pelo sistema *run.codes;*
2- monitor entrega correção e *feedback* do trabalho;
3- o aluno que desejar, poderá ressubmeter o trabalho, desde que realizadas as correções sugeridas no *feedback*; a nota poderá aumentar, **valendo até 7 pontos**.

OBS: Testem os trabalhos localmente antes de submeter no sistema.

Avaliação
Média: A nota do semestre será a média aritmética das notas das 20 atividades (M). Será aprovado com nota A := M o aluno que satisfizer todos os critérios abaixo**:**

1. 75% de frequência;
2. média M ≥ 5;
3. nota maior ou igual a 6 em **3 testes individuais**;
4. nota maior ou igual a 6 em **4 trabalhos em dupla**;
5. nota maior ou igual a 6 em **8 exercícios de laboratório**.

Do contrário, a nota de aproveitamento do semestre será A := *mínimo {4, M}*.

Exame: Poderá realizar exame, com nota E, entre 0 e 10, no dia 20/12/16, o aluno com 75% de frequência e nota A ≥ 2,5. Se o aluno fizer o exame, a nota final será *mínimo {5, (A+E)/2},* senão a nota final será *A*.

Fraude: Em caso de fraude, os envolvidos serão reprovados com nota 0.

Submissão
Para simplificar a correção, os exercícios de laboratório e trabalhos em dupla serão submetidos pelo sistema https://run.codes/. O aluno criará uma conta no sistema e deverá matricular-se com o código da disciplina divulgado. Embora o aluno poderá ressubmeter tarefas depois do prazo, se ele acumular muitas tarefas não feitas ou submeter diversas atividades em um tempo muito curto, a ressubmissão pode ser ignorada. Dúvidas, procurar monitores das turmas EF no atendimento: Mateus, Anderson, Márcio, Victor.

Gabaritos
Não serão disponibilizadas quaisquer listas de gabaritos (para exercícios, trabalhos, testes, etc.). Os alunos são fortemente encorajados a tentar resolver os exercícios e procurar frequentemente o atendimento dos monitores, que poderão verificar soluções, tirar dúvidas, sugerir ideias para soluções ou mostrar exemplos de código. Além disso, os principais exercícios e problemas serão resolvidos em sala ou na aula de laboratório.

Atendimento
Local: Laboratório 304, IC 3

Monitores: *(tabela preliminar)*	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
13h às 14h	Mateus	Márcio	Victor	Daniel
18h às 19h	Anderson	Lise	Natália	Erik

Os alunos poderão ir ao atendimento em qualquer dia/horário de preferência.

Bibliografia
O professor não seguirá nenhum livro específico. Os livros listados a seguir cobrem o conteúdo.

- A. V. Aho, J. E. Hopcroft, J. Ullmann. Data Structures and Algorithms. Addison-Wesley, 1983.
- W. Celes, R. Cerqueira, J. L. Rangel. Introdução a Estruturas de Dados. Campus, 2004.
- T. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Algoritmos - Teoria e Prática. Campus, 2002.
- M. J. Folk e B. Zoellick. File Structures. Addison-Wesley, 1992.
- F. Lorenzi, P. N. de Mattos, T. P. de Carvalho. Estruturas de Dados. Thomson, 2007.
- R. Sedgewick, Algorithms in C. Addison-Wesley, 1990.J. L. Szwarcfiter e L. Markenzon. Estruturas de Dados e Seus Algoritmos. Editora LTC, 1994.
- D. E. Knuth, The Art of Computer Programming, Vol I: Fundamental Algorithms. Addison-Wesley, 1978.
- A. M. Tenembaum. Estruturas de Dados Usando C. Makron Books, 1995.
- N. Ziviani. Projeto de Algoritmos. Thomson, 2004.

Organização do curso
Os conteúdos programados estão divididos em ***21 unidades***. As unidades estão agrupadas em grupos relacionados.

I - Estruturas Básicas de representação de dados

1. Revisão e Tipo Abstrato de Dados
   conteúdos: alocação dinâmica de memória; uso de variáveis apontadoras e vetores; definição de tipo abstrato de dados
   objetivos: alocar dados dinamicamente; abstrair estruturas complexas separando dados, operações e implementação
2. Estruturas Ligadas, Noções de Eficiência
   conteúdos: nó; notação assintótica
   objetivos: criar listas dinamicamente; calcular a eficiência de algoritmos simples
3. Operações em listas e variações
   conteúdos: listas circulares, duplamente encadeadas, operações
   objetivos: implementar operações e variantes de listas
4. Pilhas, Filas
   conteúdos: conjunto dinâmico; políticas de remoção de conjuntos; aplicação de pilhas
   objetivos: identificar situações reais que podem ser modeladas como pilhas e filas

II - Técnicas de programação recursivas

5. Recursão
   conteúdos: recursão simples; eficiência de algoritmos recursivos; pilha de execução
   objetivos: simular recursão utilizando pilha de execução
6. Divisão e Conquista
   conteúdos: algoritmos de ordenação recursivos: *merge-sort* e *quick-sort*
   objetivos: resolver subproblemas recursivamente usando divisão e conquista
7. Retrocesso
   conteúdos: problema das n-damas; solução parcial de problema; solução de problemas por retrocesso (*backtracking*)
   objetivos: identificar subproblemas; resolver subproblemas recursivamente usando retrocesso

III - Conjuntos dinâmicos em árvores

8. Árvores
   conteúdos: árvore binária
   objetivos: representar estruturas e conjuntos hierarquicamente
9. Árvores Binárias de Busca
   conteúdos: busca, inserção e remoção em árvores binárias de busca
   objetivos: implementar conjuntos dinâmicos em árvore binária
10. Árvores Balanceadas
    conteúdos: árvore AVL
    objetivos: implementar conjuntos dinâmicos em árvore binária eficientemente
11. Árvores de Afunilamento
    conteúdos: árvore de afunilamento (*splay tree*); princípio da localidade e referência
    objetivos: identificar situações em que o princípio de localidade e referência pode ser utilizado para melhorar a eficiência de um programa
12. Árvore-B
    conteúdos: árvore-B; conceitos de latência e página de disco
    objetivos: implementar conjunto dinâmico em meio de armazenamento

permanente

13. Filas de prioridade e *heap-sort*
    conteúdos: *max-heap*; método de ordenação *heap-sort*
    objetivos: implementar operação de máximo eficientemente
14. Árvore de Prefixo e Codificação de Huffman
    conteúdos: dicionário; codificação; algoritmo de Huffman
    objetivos: aplicar fila de prioridade; obter codificação de Huffman
15. Avaliando estrutura de dados para conjuntos dinâmicos **(unidade síntese)**
    conteúdos: visão geral de árvores
    objetivos: avaliar o melhor algoritmo e estrutura de dados para problemas com conjunto dinâmico de dados

IV - Estruturas de dados avançadas

16. Hashing
    conteúdos: função de hashing, tabela de espalhamento
    objetivos: implementar conjunto dinâmico em tempo quase constante
17. Grafos
    conteúdos: definição de grafos; terminologia de grafos
    objetivos: identificar situações que podem ser modeladas com grafos
18. Representação de grafos
    conteúdos: representação de grafos em memória: lista de adjacência e matriz de adjacência; percursos de grafos em largura e profundidade
    objetivos: implementar operações básicas com grafos
19. Algoritmos em grafo
    conteúdos: algoritmo de caminho mínimo (Dijkstra); ordenação topológica
    objetivos: aplicar algoritmos clássicos de grafos em problemas reais
20. Listas generalizadas e árvore geral
    conteúdos: listas generalizadas: átomo, lista; árvore geral
    objetivos: interpretar estruturas de dados como listas generalizadas
21. Gerenciamento de memória
    conteúdos: lista livre; sistema de pares; coleta de lixo; contagem de referências
    objetivos: entender implementações de alocação dinâmica e sua interferência com estrutura de dados