

Programa

Este é o plano de desenvolvimento da disciplina e um guia de estudos. Leia-o com atenção e consulte este documento durante todo o semestre. Também, sempre acompanhe os avisos na página da disciplina: <https://www.ic.unicamp.br/~lehilton/mo417a/>.

Adequação das atividades

Considerando a situação excepcional, devido à pandemia do novo coronavírus, e as instruções da reitoria da Unicamp, as atividades de ensino-aprendizagem devem continuar, adequando as atividades que originalmente foram planejadas para serem realizadas nas salas de aula e de laboratório. Tenham em mente que isso poderá implicar prejuízo no aprendizado, então é importante que tod@s estudantes se mantenham interessad@s e dedicad@s à disciplina. Serão feitas as seguintes adequações:

- as aulas expositivas serão substituídas pelo estudo dirigido e realização de exercícios de fixação; deverão ser lidos os capítulos de CLRS correspondentes aos slides disponibilizados no site, nas datas correspondentes e, posteriormente, realizados os exercícios de fixação;
- não haverá testes individuais e a avaliação será feita integralmente pelos exercícios entregues conforme definido abaixo;
- havendo demanda, o professor dará atendimento via videoconferência (usando-se a ferramenta <https://meet.google.com/>) no horário reservado para aula; para isso, os alunos devem encaminhar e-mail para o professor com suas dúvidas com pelo menos três dias de antecedência para que o professor possa se preparar.

Objetivos

Ao final do curso, @ estudante deverá ser capaz de:

- modelar problemas e subproblemas comuns: ordenação e problemas em grafos;
- projetar e desenvolver algoritmos utilizando técnicas clássicas: divisão e conquista, algoritmos gulosos e programação dinâmica;
- analisar a correção e a complexidade de algoritmos de maneira formal;
- descrever a teoria de NP-completude e suas implicações.

Pré-requisitos

Ao início do curso, @ estudante deve ser capaz de:

- descrever e desenvolver algoritmos com comandos de repetição e recursão;
- descrever e aplicar estruturas de dados: listas, filas, pilhas, árvores binárias, grafos;
- aplicar conceitos matemáticos: conjuntos, relações, funções, somatórios, lógica (proposicional) e demonstrações.

Atividades

Aulas: Antes das aulas, @s estudantes devem ler os capítulos ou seções do livro-texto correspondentes. Os assuntos e datas preliminares serão divulgados na página da disciplina com antecedência. Durante as aulas, @s estudantes devem participar levantando dúvidas, sugestões, ou possivelmente resolvendo os problemas solicitados. Após as aulas, serão propostos diversos exercícios de fixação (retirados do livro-texto e de outras fontes) para serem feitos em casa. O conteúdo das listas é considerado parte integrante do curso e @s estudantes devem resolvê-los para se prepararem para as avaliações. Recomenda-se tentar fazer as atividades individualmente e, só então, discutir em grupo.

Teste individual (avaliação): Serão realizados testes de unidade **individuais e sem consulta** com duração de 15 a 30 min (especificado em cada teste), no horário das aulas, em datas a serem divulgadas na página da disciplina. Alguns dos testes conterão questões selecionadas dentre os exercícios de fixação, com conteúdo igual ou com pequenas modificações. A nota de cada teste valerá de 0 a 10.

Exercícios (avaliação): Serão divulgadas listas de exercícios (retirados do livro-texto e de outras fontes), cada uma com prazo de entrega de uma semana, em datas a serem divulgadas na página da disciplina.

- Cada lista corresponderá a um documento em PDF no formato A4 contendo o enunciado das questões e o espaço reservado para a resposta de cada uma. Após resolvidas as questões, @estudante deverá imprimir o documento e escrever as respostas de forma **manuscrita** e somente nos campos reservados.
- Terminada a lista, ela deverá ser escaneada em formato PDF e submetida na página da disciplina <https://www.ic.unicamp.br/~lehilton/mo417a/submit/> usando a chave fornecida na primeira aula. **Recomendação**: utilize um escâner de mesa ou um aplicativo de celular com a função scan para corrigir a perspectiva criar um arquivo PDF. Trabalhos copiados, não identificados, desorganizados, incompletos, ilegíveis, ou rasurados terão a nota zerada.
- A nota de cada lista valerá de 0 a 10 e corresponderá à correção de **uma ou duas questões**, que serão sorteadas **somente depois** do prazo de entrega. O número de questões a serem corrigidas será especificado em cada lista.

Avaliação

Serão calculadas as médias aritméticas dos testes individuais (T) e dos exercícios (E). Para ser aprovado, @estudante deve satisfazer todos os critérios abaixo:

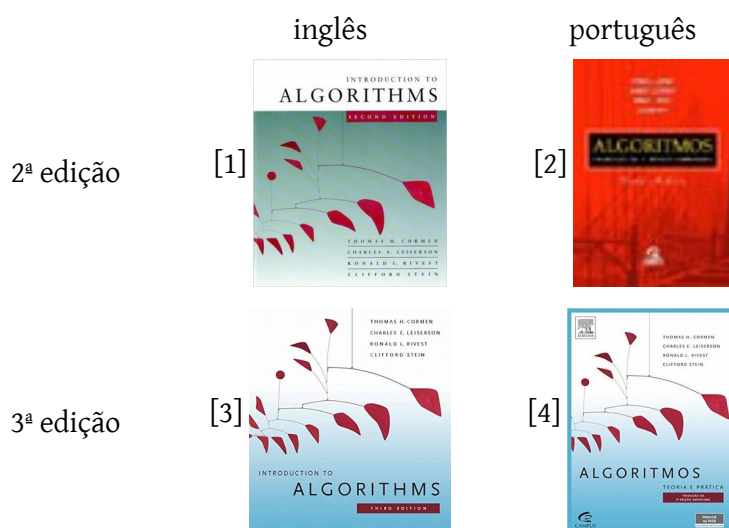
1. 75% de frequência;
2. $T \geq 5$; $E \geq 7$.

Para aprovad@s, a média será $M := (T+E)/2$, do contrário, $M := 4$. O conceito será:

A	se	$M \geq 8,5$;
B	se	$7 \leq M < 8,5$;
C	se	$6 \leq M < 7$;
D	se	$M < 6$.

Bibliografia

Será adotado como livro-texto: “Algoritmos – Teoria e Prática”, de T. Cormen, C. Leiserson, R. Rivest, C. Stein (CLRS). Poderá ser usada a segunda ou a terceira edição, tanto na versão em inglês quanto na versão traduzida. Por causa das diferentes numerações entre as edições, os exercícios solicitados do livro poderão ser transcritos.



[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms (2ª edição), 2001.

[2] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos – Teoria e Prática (2ª edição), 2002.

[3] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms (3ª edição), 2009.

[4] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos – Teoria e Prática (3ª edição), 2012.

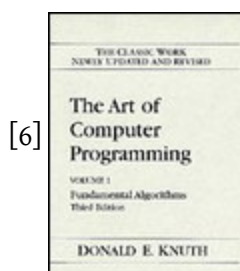
Além do livro de CLRS, em algumas unidades, também será utilizado o livro de Udi Manber,

particularmente nas unidades referentes a indução e projeto de algoritmos recursivos.

[5] U. Manber, Algorithms. A Creative Approach, 1989.

Além desses, poderão ser consultados outros livros, disponíveis na biblioteca, como o livro clássico de Donald Knuth, que é considerado o pai da análise de algoritmo.

[6] D. E. Knuth. The Art of Computer Programming, 1974.



Há alguns livros pouco mais recentes, como o livro do Jon Kleinberg e da Éva Tardos, que inclui, além dos tópicos estudados, alguns capítulos para tratamento de problemas NP-difíceis,

[7] J. Kleinberg, E. Tardos. Algorithm Design, 2005.

e o livro de Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani, que é um pouco mais informal e com um ritmo diferente do CLRS.

[8] S. Dasgupta, C. Papadimitriou, U. Vazirani. Algorithms, 2006.

Livros de autores brasileiros importantes também estão disponíveis.

[9] J. L. Szwarcfiter. Grafos e Algoritmos Computacionais, 1984.

[10] N. Ziviani. Projeto de Algoritmos (2ª edição), 2004.

A teoria de NP-completude e os modelos de computação serão tratados apenas superficialmente nesse curso e utilizando a abordagem de CLRS. Um tratamento mais aprofundado de Teoria da Computação e com a abordagem mais tradicional é assunto da disciplina Linguagens Formais e Autômatos, que tem entre referências clássicas o livro de Sipser.

[11] M. Sipser. Introduction to the Theory of Computation (3ª edição), 2012.

Já um aprofundamento em complexidade pode ser encontrado no livro de Arora e Barak.

[12] S. Arora and B. Barak. Computational Complexity: A Modern Approach, 2009.

Uma leitura mais gentil sobre modelos computacionais e reduções e recomendada para estudantes desse curso é o primeiro capítulo do livro dos professores Rezende e Stolfi.

[13] P. J. de Rezende, J. Stolfi. Fundamentos de geometria computacional, 1994. Disponível em: <https://www.ic.unicamp.br/~rezende/rez-sto-94-fgc.pdf>.

Não menos importante, na internet há diversos e excelentes recursos que podem servir para pesquisa. Em particular, veja o curso de análise de algoritmos do prof. Paulo Feofiloff.

[14] P. Feofiloff. Análise de Algoritmos. https://www.ime.usp.br/~pf/analise_de_algoritmos/.

Um bom conjunto de notas de aulas está livremente disponível.

[15] J. Erickson. Algorithms, Etc. <http://jeffe.cs.illinois.edu/teaching/algorithms/>.

E, lendo com cuidado, a Wikipédia é uma excelente fonte de consulta.

[16] Wikipédia. https://en.wikipedia.org/wiki/Analysis_of_algorithms.

Sugestão: Não tente ler várias referências de uma vez; na maior parte do conteúdo, atenha-se ao livro-texto ([1]-[4]) e consulte o livro [5] nos capítulos em que ele for usado. Consulte a bibliografia complementar sempre que solicitado, para se aprofundar, ou se tiver dificuldade com algum assunto do livro-texto e quiser uma apresentação diferente. Ler os exercícios de outras fontes (e tentar resolver os que achar mais interessantes) também é uma boa maneira de estudar. Não leia assuntos mais aprofundados (sobre Teoria da Computação e Complexidade Computacional) antes de entender o

conteúdo básico correspondente tratado por CRLS. Finalmente, antes de começar a ler e fazer exercícios, certifique-se de que entendeu o espírito, a forma e o porquê de uma demonstração matemática. Se não, então o link [O que é uma prova matemática?](#) da referência [14] é um bom lugar para começar a estudar.

Material didático

Os slides usados em aula serão disponibilizados. A maioria dos exemplos dos slides e exercícios adicionais foram ou criados e gentilmente cedidos pelo prof. [Cid Carvalho de Souza](#) e pela profa. [Cândida Nunes da Silva](#) (particularmente com modificações do prof. [Orlando Lee](#)); ou criados e gentilmente cedidos pelo prof. [Flávio Keidi Miyazawa](#). Eu recriei ou reestruturei o conteúdo das unidades, possivelmente introduzindo erros, que devem ser reportados a mim.

Horário e local

As aulas serão ministradas na sala 352, das 16 h às 18 h, terças e quintas-feiras.

Atendimento

Poderá ser combinado um horário para atendimento com o professor via e-mail por um@ ou mais estudantes, desde que solicitado com, pelo menos, três dias de antecedência.

Rotina de estudo

Importante: Note que não há provas final e de meio de semestre. Portanto é fundamental criar uma rotina de estudos contínua. Planeje-se e separe algumas horas e alguns dias por semana para ler o livro e resolver as listas de exercícios. @s estudantes também são encorajados a se reunir e estudar em grupo (sempre depois de tentar resolver os exercícios individualmente). Cada um@ tem sua própria maneira de estudar. Não obstante, algumas sugestões são úteis para o bom desenvolvimento da disciplina:

1. Leia o capítulo ou seção do livro correspondente ao conteúdo antes da aula correspondente (veja a ordem dos conteúdos abaixo); utilize a aula principalmente para tirar dúvidas e confirmar o seu entendimento.
2. Faça ou tente fazer os exercícios correspondentes a uma aula no próximo horário que tiver reservado para estudar a disciplina; anote as principais dificuldades e discuta com colegas e, persistindo, com o professor no início das próximas aulas.
3. Veja o resultado das avaliações! Elas servem para que você identifique os problemas (erro de lógica, incorreção, conceitos incorretos, dificuldade com formalismo, incompletude, erros de português e escrita matemática, etc.). Tente refazer as tarefas e, se não puder identificar o que está errado ou não conseguir corrigir o problema, anote a dúvida e leve-a ao professor (mas evite pedir aumento de nota, ou comparar notas de colegas).

Organização do curso

O curso é dividido em unidades. A ordem de unidades e de conteúdos abaixo serve para que @ estudante se planeje e estude para as aulas. Ela pode variar dependendo do andamento da disciplina e as datas serão divulgadas na página da disciplina.

1. **Introdução**
 - ▶ Introdução a análise e projeto de algoritmos
 - ▶ Análise de complexidade
 - ▶ Projeto de algoritmos
2. **Demonstrações e princípio da indução**
 - ▶ Técnicas e escrita de demonstração
 - ▶ Princípio da indução
3. **Crescimento de funções**
 - ▶ Notação assintótica
 - ▶ Recorrências
4. **Correção de algoritmos**
 - ▶ Invariante de laço
 - ▶ Algoritmos recursivos
5. **Projeto de algoritmos recursivos**
 - ▶ Recursão simples
 - ▶ Recursão composta
 - ▶ Divisão e conquista
6. **Fila de prioridade**
 - ▶ Selectionsort
 - ▶ Heapsort e fila de prioridade
 - ▶ Visão geral de algoritmos de ordenação
7. **Análise probabilística e algoritmos aleatorizados**
 - ▶ Revisão de conceitos de probabilidade
 - ▶ Análise probabilística
 - ▶ Quicksort determinístico
 - ▶ Quicksort aleatorizado
8. **Ordenação em tempo linear**
 - ▶ Cota inferior para ordenação

- ▶ Ordenação de tempo linear
- 9. **Estatísticas de ordem**
 - ▶ Problema seleção
- 10. **Programação Dinâmica**
 - ▶ Subproblemas
 - ▶ Algoritmos baseados em programação dinâmica
- 11. **Algoritmos Gulosos**
 - ▶ Escolha local
 - ▶ Algoritmos gulosos
- 12. **Algoritmos e conceitos fundamentais de grafos**
 - ▶ Conceitos de grafos
 - ▶ Fatos básicos de grafos
 - ▶ Representação de grafos
- 13. **Buscas em grafos**
 - ▶ Busca em largura
 - ▶ Busca em profundidade
 - ▶ Ordenação topológica
- ▶ Componentes fortemente conexas
- 14. **Árvore geradora mínima**
 - ▶ Arvore geradora mínima e algoritmo de Prim
 - ▶ Algoritmo de Kruskal e conjuntos disjuntos
 - ▶ Conjuntos disjuntos com florestas disjuntas
- 15. **Caminhos mínimos**
 - ▶ Caminhos mínimos com uma origem
 - ▶ Algoritmo de Dijkstra
 - ▶ Algoritmo de Bellman-Ford
 - ▶ Caminhos mínimos entre todos os pares de vértices
- 16. **Redução entre problemas**
 - ▶ Conceitos de redução entre problemas
 - ▶ Exemplos de reduções
- 17. **NP-completude**
 - ▶ Classes de problemas e problemas polinomiais
 - ▶ Problemas verificáveis em tempo polinomial
 - ▶ Demonstrações de NP-completude