Instituto de Computação/Unicamp – Complexidade de Algoritmos I – M0417A

Prof. Lehilton Lelis Chaves Pedrosa

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, acompanhe os avisos no Google Classroom.

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 correspondes. 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 primeiro ler e tentar fazer as atividades individualmente e, depois, discutir em grupo.

Tarefas (avaliação): Haverá diversas tarefas com prazo de entrega de pelo menos uma semana com divulgação e entrega via Google Classroom. Cada tarefa será ou uma lista de exercícios, ou uma atividade de programação.

- As tarefas devem ser realizadas individualmente ou em dupla. Se realizada em dupla, apenas um estudante deve entregar. É permitida a consulta apenas ao texto da bibliografia, i.e., não é permitida a ajuda de terceiros ou consulta a respostas prontas na internet, etc. Ao entregar uma tarefa, os estudantes declaram que realizaram as tarefas individualmente ou em dupla.
- As listas de exercícios poderão conter exercícios selecionados do livro-texto e de outras fontes. As respostas devem ser dissertativas, concisas e coesas (não serão aceitas respostas contendo apenas fórmulas soltas, sequências sem explicação, frases desconectadas, etc.).
- As respostas dos exercícios devem estar em folha A4, preferencialmente manuscritas, respeitando margens de pelo menos 3cm em todos os lados, organizadas e ordenadas pelo número da questão. Se manuscritas, a letra deve ser grade e legível, limpa e saltando amplo espaço entre linhas; se digitadas, deve-se usar fonte com tamanho pelo menos 12pt e com espaçamento duplo. O documento deve ser digitalizado e entregue em formato PDF.
- A nota de cada lista de exercícios 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.

Prova (avaliação): Haverá uma prova individual, cobrindo todo o conteúdo ministrado, em data a ser divulgada no Google Classroom com pelo menos uma semana de antecedência. A prova conterá questões extraídas das listas de exercícios. Poderão ser consultadas folhas de papel (pautado ou não), desde que **redigidas e manuscritas pelo próprio estudante**. Caso se levem folhas de consulta, todas elas devem estar devidamente grampeadas e devem ser entregues no final da prova. Não será permitida a consulta a qualquer outro material ou equipamento.

Avaliação

Será calculada as média aritmética das tarefas (T) e a nota da prova (P). Para ser aprovad@, @ estudante deverá satisfizer todos os critérios abaixo.

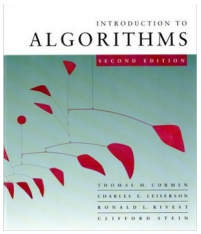
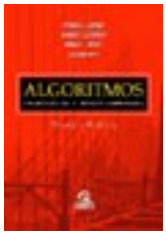
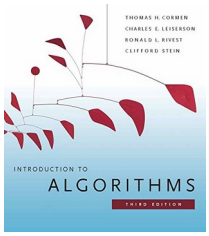
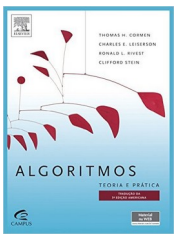
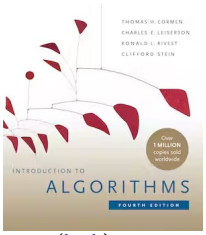
1. 75% de frequência;
2. T ≥ 7; P ≥ 4.

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

$$\begin{cases} \text{A} & \text{se} \quad M \geq 8{,}5\,; \\ \text{B} & \text{se} \quad 7 \leq M < 8{,}5\,; \\ \text{C} & \text{se} \quad 6 \leq M < 7\,; \\ \text{D} & \text{se} \quad M < 6\ . \end{cases}$$

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. Caso utilize a quarta edição, deve-se procurar os exercícios correspondentes. Por causa das diferentes numerações entre as edições, os exercícios solicitados do livro poderão ser transcritos. A biblioteca deve conter alguns exemplares do livro; reserve com antecedência.

	inglês	português
2ª edição	[1] 	[2] 
3ª edição	[3] 	[4] 
4ª edição	[5] 	

(link)

[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.
[5] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms (4ª edição), 2022.

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

[6] 

[7] 

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

Livros de autores brasileiros importantes também estão disponíveis.
[10] J. L. Szwarcfiter. Grafos e Algoritmos Computacionais, 1984.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] P. Feofiloff. Análise de Algoritmos. https://www.ime.usp.br/~pf/analise_de_algoritmos/.

Um bom conjunto de notas de aulas está livremente disponível.
[16] J. Erickson. Algorithms, Etc. http://jeffe.cs.illinois.edu/teaching/algorithms/.

E, lendo com cuidado, a Wikipédia é uma excelente fonte de consulta.
[17] 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]-[5]) e consulte o livro [6] 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 [15] 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 351, das 10 h às 12 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: A prova final consiste em resolver alguns exercícios selecionados das listas de fixação. Portanto é fundamental criar uma rotina de estudos contínua e tentar realizar o maior número de exercícios durante o semestre. Planeje-se e separe algumas horas e alguns dias por semana para ler o livro e resolver as listas. @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.

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 no Google Classroom.

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