Instituto de Computação — Universidade de Campinas

Estrutura de Dados

Segundo Semestre de 2020 – MC202AB

Prof. *Lehilton Lelis Chaves Pedrosa*

Monitores Leonardo, Elcio, Pedro (PED)

Monitores Guilherme, Ieremis, Natan (PAD)

Introdução

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.

Pré-requisitos

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

- descrever problemas computacionais, computabilidade e limitações;
- descrever organização básica de computadores e de sistemas de software;
- utilizar estruturas de dados elementares: variável, lista, matriz
- escrever algoritmos iterativos e recursivos;
- implementar e testar programas escritos em Python ou em C.

Além disso, é recomendável que @ estudante seja capaz de:

- interpretar e descrever problemas e situações textualmente;
- desenvolver e executar um cronograma de estudo contínuo;
- consultar, organizar e resumir fontes de pesquisa diversas (bibliotecas e internet).

Objetivos

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

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

Adequação das atividades

Considerando a situação excepcional, devido à pandemia do novo coronavírus, as atividades de ensino-aprendizagem continuarão online. Caso algum@ estudante não tenha acesso aos recursos necessários, ou não possa realizar as atividades programadas por questões pessoais, deve-se enviar um e-mail

diretamente para o professor explicando os motivos. Será proposto um plano alternativo adequado a cada caso.

Atividades

Aula expositiva

As aulas ocorrerão online via Google Meet. O professor fará a exposição do conteúdo das várias unidades e, depois disso, os @s estudantes devem participar levantando questões, sugestões, ou possivelmente resolvendo os problemas solicitados. A qualquer momento, estudantes podem tirar dúvidas com o professor.

Exercícios de fixação

Serão propostos diversos exercícios de fixação para serem realizado após as aulas. O conteúdo das listas é considerado parte integrante do curso. Há tanto questões teóricas quanto práticas. Implementar os programas solicitados nas questões serve como um exercício mais simples antes das tarefas de programação. Recomenda-se tentar fazer esses exercícios tanto individualmente quanto em grupo.

Vocês podem compartilhar e discutir e trechos de códigos nesses **exercícios de fixação**. Sempre que copiar ou discutir qualquer trecho de código de um exercício, faça um comentário indicando a fonte ou as pessoas com quem discutiu. Essa é uma oportunidade de compartilhar e trocar experiências com seus colegas. Como as questões são abertas, não há gabarito; as dúvidas devem ser anotadas e levadas a monitoria nos horários de atendimentos, ou ao professor.

Todos os exercícios de fixação **deverão ser colocados no repositório pessoal git** de cada estudante, de maneira organizada. Esses exercícios não serão corrigidos e não é obrigatório realizar cada lista completamente, mas ter realizado ou tentado realizar uma parte considerável dos exercícios é importante para a nota de participação.

Testes de aprendizagem

Haverá diversos testes corrigidos automaticamente via Google Classroom. Cada teste serve para que @ estudante verifique seu entendimento logo após cada unidade ser ministrada. Não é obrigatório acertar todas as questões de cada teste, mas a média dos testes será considerada na nota de participação.

Tarefas

Haverá 12 tarefas que servirão de avaliação da disciplina. O número previsto de tarefas pode mudar a depender do andamento da turma. Cada tarefa é um conjunto de um ou mais exercícios de programação ou trabalhos mais elaborados, que deverão ser implementados na linguagem de programação C. Todas as tarefas devem ser desenvolvidas em um repositório de controle de versões Git, utilizando a infraestrutura do IC https://gitlab.ic.unicamp.br/. Para criar uma conta, logue-se no sistema utilizando a senha fornecida

pelo IC. Cada estudante terá um repositório de nome `raXXXXXX`, em que `XXXXXX` corresponde ao número de RA.

A sequência das atividades será a seguinte:

1. O enunciado da tarefa é publicado após ou simultaneamente os conteúdos correspondentes serem ministrados.

2. Um conjunto de arquivos auxiliares da tarefa será inserido automaticamente no repositório pessoal em um diretório de nome `tarefaNN`, onde `NN` é o número da tarefa. Esses arquivos auxiliares podem ser acessados bastando executar o comando `git pull` na cópia de trabalho pessoal.

3. @ estudante realiza os vários exercícios da tarefa, alterando ou incluindo novos arquivos no repositório correspondente. À medida em que cada pequeno passo for realizado, deve-se fazer um `git commit` com uma mensagem adequada. É **obrigatório** que todo o desenvolvimento da tarefa seja registrado utilizando o controle de versões. Não serão aceitas tarefas que não tenham histórico de commits ou submetidas por outros meios.

4. A cada pequena parte implementada, deve-se testar o programa. Para isso, pode-se executar em um terminal `python3 testar.py`, o que executará um script de teste fornecido juntamente com os arquivos auxiliares. Além desses, pode ser necessário criar outros testes.

5. Terminada a hora de trabalho, @ estudante executa o comando `git push`. Isso enviará as alterações para o repositório remoto. Além disso, ativará um script de correção, que testará automaticamente a tarefa (utilizando os mesmos arquivos auxiliares e/ou outros testes). O resultado dessa correção ficará anotada na planilha de notas disponibilizada para cada aluno.

Algumas tarefas serão corrigidas apenas pelo sistema de correção automático. As demais tarefas serão corrigidas tanto automaticamente quanto por um monitor (**uma única vez**). O monitor fará a correção offline e escreverá um pequeno relatório contendo problemas, sugestões e outros comentários a respeito da tarefa. ***Não*** *será necessário apresentar nenhuma tarefa pessoalmente*. Embora não seja necessário apresentar as tarefas aos monitores, @s estudantes são encorajad@s a mostrar e conversar sobre suas dúvidas e seus códigos com os monitores, via chat e videoconferência, sempre que desejarem.

Prazos

A disciplina adotará um método de avaliação contínua e individualizado. Assim, as tarefas serão divulgadas continuamente, de acordo com os conteúdos ministrados, mas cada um@ poderá levar mais tempo ou menos tempo para executá-las, de acordo com seu aprendizado sobre o conteúdo. As regras são as seguintes:

- Cada tarefa terá uma nota de **0 a 10**, que será calculada a partir de testes automáticos fechados, além de outros critérios especificados na tarefa que serão verificados pelo monitor.
- Cada tarefa terá um prazo recomendado de **pelo menos uma semana**. Se uma tarefa for entregue depois do prazo recomendado, será **descontado 10%** da nota correspondente.
- Todas as tarefas são **obrigatórias** e devem ser entregues até **10/1/2021**.

- Uma tarefá só será corrigida após as tarefas anteriores tiverem sido realizadas corretamente (conceitos A, B ou C). O conceito D significa que a tarefa deve ser terminada ou corrigida antes que o monitor possa revisá-la.

É responsabilidade de cada um@ organizar e dividir o tempo para realizar cada atividade. Recomenda-se fortemente que se realizem todas as tarfas no prazo recomendado e que não se deixem tarefas pendentes acumuladas.

Avaliação

A nota da disciplina será a média ponderada de duas partes

1. A média aritmética *M* das notas de todas as tarefas, valendo de 0 a 10.
2. A nota de participação *P* do estudante atribuída pessoalmente pelo professor, valendo de 0 a 10.

Serão considerados aprovados os estudantes que tiverem obtido conceito pelo menos C em todas as tarefas. Nesse caso, a nota de aproveitamento será ***A=0,7M+0,3P***. Do contrário, a nota de aproveitamento do semestre será ***A=mínimo{4,M}***.

Todos estudantes terão frequência 100%. Para calcular a nota de participação serão levadas em conta tanto a participação nas aulas online quanto a realização das demais atividades. Isso será monitorado considerando a atividade no repositório git bem como a participação no chat de discussão e atendimento. Para garantir a nota máxima de participação, deve-se:

- Manter atividade regular no repositório git com realização de 50% ou mais das questões em cada uma das listas de exercícios de fixação.
- Realizar todos os testes no Google Classroom e obter média ponderada pelo menos 5.
- Participar das aulas e/ou dos canais de chat, compartilhando dúvidas, respondendo perguntas ou fazendo comentários sobre a disciplina.
- Ter apresentado pelo menos uma tarefa ao professor nos canais de atendimento (entre as tarefas de número 4 a 12).

Exame

Aquel@ estudante que não tiver conseguido completar todas as tarefas até o prazo poderá realizar exame. Como exame, o professor solicitará a realização de um subconjunto das tarefas com prazo de pelo menos uma semana e atribuirá nota *E*. Para estudantes que fizerem o exame, a nota final será ***mínimo{5, (A+E)/2}***.

Fraude

Em caso de fraude (plágio, atestado falso, assinar lista por colegas, abandonar aula após assinar, usar bibliotecas não permitidas, copiar quaisquer trechos da internet sem autorização expressa, mostrar ou distribuir laboratório de programação individual, cola independentemente de origem, consulta a material

proibido etc.), os envolvidos serão reprovados com nota 0 e será registrada a ocorrência no histórico escolar. Fique atento:

- Cada um@ é responsável por manter seguros os arquivos de seu repositório. Não compartilhe sua senha nem deixe cópias de arquivos em computadores compartilhados. **Todas** as tarefas são individuais.
- É permitido compartilhar e discutir sobre trechos de código apenas dos exercícios de fixação. Sempre que utilizar trechos de código de outras pessoas ou de outras fontes, deve-se indicar isso explicitamente.
- É permitido tirar dúvidas e discutir a respeito de tarefas de programação com colegas e monitores. Mas toda discussão deve ser feita em canais abertos no Discord e não é permitido compartilhar algoritmos (em português, código ou pseudocódigo) nem ideias prontas de solução etc.
- O material disponibilizado e os trabalhos apresentados nas avaliações são para uso exclusivo da disciplina. Não compartilhem seus trabalhos mesmo após o término da disciplina. Se desejar publicar algum trabalho que realizou, converse antes com o professor.
- Cláusula de arrependimento: se você cometer qualquer tipo de fraude, mas reconhecê-la **antes** de ser comunicad@, então o professor irá relaxar as ações acima por ações locais.

Conteúdo e bibliografia

O professor não seguirá nenhum livro específico. O conteúdo abordado é coberto por capítulos dos livros listados a seguir.

1. T. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Algoritmos - Teoria e Prática. Campus, 2002.
2. R. Sedgewick, Algorithms in C. Addison-Wesley, 1990.
3. D. E. Knuth, The Art of Computer Programming, Vol I. Addison-Wesley, 1978.
4. A. V. Aho, J. E. Hopcroft, J. Ullmann. Data Structures and Algorithms. Addison-Wesley, 1983.
5. W. Celes, R. Cerqueira, J. L. Rangel. Introdução a Estruturas de Dados. Campus, 2004.
6. M. J. Folk e B. Zoellick. File Structures. Addison-Wesley, 1992.
7. F. Lorenzi, P. N. de Mattos, T. P. de Carvalho. Estruturas de Dados. Thomson, 2007.
8. S. L. Pereira. Estruturas de Dados Fundamentais. Érica, 1996.
9. E. M. Reingold e W. J. Hanson, Data Structures. Little-Brown, 1983.
10. J. L. Szwarcfiter e L. Markenzon. Estruturas de Dados e Seus Algoritmos. Editora LTC, 1994.
11. N. Wirth, Algorithms + Data Structures = Programs. Prentice-Hall, 1976.
12. A. M. Tenembaum. Estruturas de Dados Usando C. Makron Books, 1995.
13. N. Ziviani. Projeto de Algoritmos. Thomson, 2004.

As principais referências são os livros [1] e [2] (particularmente para conteúdos de árvores rubro-negra, árvores B, etc). O livro [3] é um livro clássico em computação e aborda tanto estruturas elementares quanto estruturas mais avançadas (como generalizações de árvores e listas). Alguns exercícios propostos foram retirados ou inspirados pelas demais referências (em particular, [4], [10], [11] e [12]). Embora

estejam listadas as edições clássicas dos livros, as edições mais novas também podem ser consultadas sem prejuízo do conteúdo a ser visto; há alguns exemplares disponíveis na biblioteca. Além dos livros, pesquisar com cuidado na internet também é um bom jeito de aprender. Procure os verbetes correspondentes às estruturas vistas na Wikipédia. Você pode visualizar vários dos algoritmos estudados no site em http://visualgo.net/.

Links úteis

Links recomendados para a disciplina:

- Introdução à linguagem de programação C
- Programas recomendados para a disciplina
    - Editor de texto VSCode
    - Ferramenta gráfica para comparar arquivos Meld ou WinMerge
- Usando o controle de versões Git
    - Página oficial do Git
    - Um tutorial direto ao ponto

Alguns outros links interessantes:

- Ferramentas online
    - Visualização de vários dos algoritmos estudados
    - Simulador de pequenos trechos de código que permite visualizar a memória do programa
    - Um editor on-line multiplayer
- Textos de apoio
    - Uma série de belos artigos sobre estruturas de dados
    - Uma referência popular sobre C
    - Vários exemplos de programas
    - Alguns tutoriais para diferentes aplicações

Material didático

Será disponibilizado no Google Classroom material de apoio para estudo, que inclui um tutorial para programar em C, slides apresentados em sala de aula e exercícios para prática de programação. Parte do material dos slides foi construído e cedido pelo prof. Rafael. Os slides servem apenas para revisar a discussão em sala e não formam uma referência completa do conteúdo. Para o estudo, é recomendado que @ estudante, principalmente, pratique programação resolvendo e implementando os exercícios propostos e, além disso, busque e estude exemplos de código-fonte nos capítulos correspondentes dos livros-textos.

Observação: Nos trabalhos de programação, podem ser copiados trechos de códigos disponibilizados nos slides. Revise-os atentamente e observe que esses trechos têm função didática, então são incompletos e contêm erros. Cópia de trechos de código de qualquer outro material sem autorização expressa do professor será considerada fraude.

Atendimento

As aulas com o professor serão via Google Meet. O atendimento do professor e de monitores será feito via Discord. Cada estudante receberá um link por e-mail com o convite para se cadastrar no servidor. É recomendável acessar esse site utilizando um computador, mas também é possível baixar e instalar o aplicativo de celular disponível.

Todos podem escrever e ler as mensagens nessa sala e vocês são **fortemente encorajados** tanto a fazer perguntas quanto a responder dúvidas de colegas. O professor e os monitores acompanharão as conversas e tentarão responder sempre que possível. Pelo menos um monitor ficará disponível no chat nos horários de atendimento marcados na tabela. Esses horários podem sofre alterações de acordo com a demanda de estudantes e disponibilidade dos monitores.

(OS HORÁRIOS DE ATENDIMENTO DOS MONITORES SERÃO DEFINIDOS EM BREVE)

	segunda-feira	**terça-feira**	**quarta-feira**	**quinta-feira**	**sexta-feira**
10h - 11h		aula online		aula online	
13h - 15h	atendimento*				
17h - 19h	atendimento*				

O professor responderá dúvidas gerais durante as aulas após a parte inicial com a exposição do conteúdo. Para conversar pessoalmente e individualmente com o professor, você deve procurar atendimento nos horários marcados com asterisco que correspondem aos horários de laboratório reservados para a disciplina. Se houver várias pessoas procurando atendimento ao mesmo, deve-se esperar a vez ou voltar no próximo horário disponível, respeitando-se a fila.