Plano de desenvolvimento da disciplina

Instituto de Computação — Universidade de Campinas

Estrutura de Dados

Segundo Semestre de 2021 – MC202E

Prof. *Lehilton Lelis Chaves Pedrosa*

Monitores *Guilherme, Jonathas (PED)* e *Frederico, Matheus, Wallace (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.

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.

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:

- 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).

Adequação das atividades

Considerando a situação excepcional, devido à pandemia do novo coronavírus, as atividades de ensino-aprendizagem dar-se-ão online. Espera-se que cada estudante tenha disponível

- Um computador desktop ou laptop, com sistema operacional Unix-like (Linux/MacOS) ou Windows em que seja possível instalar Python 3, Git e toolchain GCC .
- Acesso a Internet com conexão estável nos horários de atendimento programados. Não é necessário ter conexão de alta velocidade.

Caso algum@ estudante não tenha acesso aos recursos necessários, ou não possa realizar as atividades programadas por quaisquer questões pessoais, deve-se enviar um e-mail diretamente para o professor explicando os motivos no início do semestre. 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.

Listas de exercícios para fixação

Serão propostos diversas listas de exercícios para fixação para serem realizados *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. Como as questões são abertas, não há gabarito. Os exercícios não serão corrigidos e não é obrigatório realizá-los.

Recomenda-se tentar fazer esses exercícios tanto individualmente quanto em grupo com no máximo **quatro** estudantes. Se desejarem resolver as atividades em grupo, então criem um repositório no Gitlab e adicionem o professor como convidado. Além disso, vocês também podem adicionar monitores como convidados, para que eles lhes auxiliem durante o atendimento.

Tarefas

Haverá diversas tarefas individuais que servirão de avaliação da disciplina. O número previsto de tarefas é 17 e 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, isso é, **não** escreva suas tarefas e copie de outros locais e faça commit mesmo dos programas não prontos, parcialmente escritos. 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.

As tarefas serão corrigidas pelo sistema de correção automática e por um monitor. Uma vez terminada a tarefa e após ela ter passado nos testes automáticos, deve-se **solicitar a correção ao monitor** clicando-se no botão correspondente da interface de notas (com exceção das tarefas que indique que serão corrigidas apenas automaticamente). Algumas tarefas determinadas (especificadas no enunciado), deverão ser *apresentadas pessoalmente* a um monitor por meio de videoconferência de maneira sucinta nos horários de atendimento disponíveis. Além disso, @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 nos canais de atendimento.

A cada tarefa será atribuído um conceito com os seguintes significados:

- **A**: a tarefa foi executada satisfatoriamente; deve-se prosseguir à proxima atividade;
- **B**: os objetivos mínimos foram alcançados, mas há questões pontuais que podem ser melhoradas; deve-se prosseguir à proxima atividade;
- **C**: nem todos os objetivos mínimos foram alcançados; a tarefa é considerada aprovada, mas recomenda-se corrigi-la e submeter novamente;
- **D**: a tarefa não foi realizada, ou os objetivos não foram satisfeitos; é obrigatório refazer a tarefa.

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 um prazo recomendado de pelo menos uma semana a partir da publicação. Se uma tarefa for entregue depois do prazo recomendado, será **descontado 15%** da nota correspondente.
- Todas as tarefas são **obrigatórias** e devem ser entregues até **5/12/2021**.
- Uma tarefá só será corrigida após as tarefas anteriores tiverem sido realizadas corretamente (tiverem recebido conceitos A, B ou C).

É responsabilidade de cada um@ organizar e dividir o tempo para realizar cada atividade. É proibido acumular tarefas deliberadamente: estudantes com tarefas atrasadas que não tiverem nenhuma atividade no repositório por duas semanas deverão apresentar justificativa ao professor. Caso um estudante não responda as mensagens de aviso do professor, será presumida a desistência da disciplina e a correção de suas tarefas será bloqueada. Recomenda-se fortemente que se realize a maior parte dos exercícios possível durante as aulas de laboratório e 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}***.

Nota de participação

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 regular no repositório git bem como a participação nas aulas, no chat de discussão e no atendimento. Para garantir a nota máxima de participação, deve-se:

- Manter atividade regular no repositório git, realizando inclusões e alterações de pelo menos **200** linhas de código no repositório em cada semana (até 7 pontos). Vocês podem incluir no repositório trechos de códigos das **listas de exercícios para fixação** realizados em grupo, mas é obrigatório fazer um comentário indicando as pessoas com quem colaborou.
- Participar continuamente das aulas e/ou dos canais de chat, compartilhando dúvidas, respondendo perguntas ou fazendo comentários pertinentes sobre a disciplina (até 3 pontos).
- Apresentar uma tarefa (a partir da tarefa 7) ao professor nos canais de atendimento aumentará a nota de participação em 3 pontos. Não será uma correção ou atribuição de nota da tarefa. O objetivo é dar uma

oportunidade para conversar e receber feedback diretamente do professor sobre alguma atividade mais elaborada realizada.

Exame

Aquel@ estudante com $A \geq 2,5$ 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*. O prazo para a entrega das tarefas do exame é 16/12. Para estudantes que fizerem o exame, a nota final será $\boldsymbol{mínimo\{5,(A+E)/2\}}$.

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 proibido utilizar ajuda de terceiros (amigos, professores particulares, etc.) para realizar as tarefas sem autorização expressa do professor da disciplina.
- É permitido compartilhar e discutir sobre trechos de código apenas dos exercícios para 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 para resolver a tarefa (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/.

Material didático

A página da disciplina conterá um 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. A página também inclui alguns tutoriais para instalar o ambiente, configurar um editor, utilizar git e depurar programas. Esses tutoriais foram gentilmente criados por monitores de diversos semestres.

Aulas e atendimento

As aulas expositivas com o professor serão via Google Meet às terças das 21 às 23h e às quintas das 19 às 21h. Será divulgado um link individual acessar a videoconferência de cada aula. Para acessar o link, deve-se usar a conta de e-mail institucional DAC (i.e., o e-mail da forma 123456@dac.unicamp.br).

O atendimento e comunicação entre estudantes, professor e monitores serão feitos via Discord. Cada estudante receberá um link por e-mail com o convite para se cadastrar no servidor. Você pode utilizar uma conta de Discord existente, mas deve alterar seu apelido no servidor da disciplina para seu nome completo e número de RA (e.g., `Fulano da Silva (123456)`). Se preferir, também pode criar uma conta nova com e-mail institucional.

É recomendável utilizar o Discord a partir do computador (utilizando o programa para desktop ou o site), mas também é possível baixar e instalar o aplicativo de celular disponível.

Todos podem escrever e ler as mensagens nas salas de atendimento 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. A tabela com os horários de antedimento estará disponível no Discord e pode sofrer alterações de acordo com a demanda de estudantes e disponibilidade dos monitores.

Observações importantes:

1. Observe os horários de atendimento com atenção e siga as regras descritas no servidor do Discord.

2. Para correção de tarefas que peçam apresentação, compareça a um atendimento com um monitor PED da turma E, de acordo com a tabela de horários abaixo.

3. 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, ou para apresentar uma tarefa ao 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 tempo, deve-se esperar a vez ou voltar no próximo horário disponível, respeitando a fila.

	segunda	terça	quarta	quinta	sexta	sábado
14h - 16h				Guilherme (PED)		
17h - 19h	Jonathas (PED)				Guilherme (PED)	
19h - 21h			Jonathas (PED) professor*	aula		
21h - 23h		aula				