MC202 — Estruturas de Dados — Turma AB
Professor: Rafael Crivellari Saliba Schouery
E-mail: rafael@ic.unicamp.br
Segundo Semestre de 2024

Página da Disciplina Na página `https://ic.unicamp.br/~rafael/mc202.html` é possível encontrar todos os slides da disciplina, bem como exercícios sugeridos e resolvidos, vídeos, materiais relacionados, tutoriais, links, e bibliografia.

Google Sala de Aula O Google Sala de Aula será utilizado, principalmente para a divulgação de testes (veja mais abaixo) e notas.

Atendimento O atendimento será prestado nas aulas presenciais de laboratório. Os monitores farão atendimentos também em horários adicionais a serem divulgados no começo do semestre. Além disso, será disponibilizado um servidor no Discord para dúvidas e discussões.

Ementa da Disciplina Estruturas básicas para representação de informações: listas, árvores, grafos e suas generalizações. Algoritmos para construção, consulta e manipulação de tais estruturas. Desenvolvimento, implementação e testes de programas usando tais estruturas em aplicações específicas.

Programa da Disciplina Estruturas ligadas: nó, apontador, variável apontadora, alocação dinâmica de memória • Listas ligadas simples: operações básicas • Comparação de listas ligadas com vetores • Algoritmos gerais para listas simples: enumeração, inversão, cópia, concatenação • Pilhas, filas, e aplicações • Intercalação (merge) de listas e mergesort; análise informal • Variações: listas circulares, duplamente ligadas, com cabeça. Lista livre • Algoritmos de ordenação • Árvores binárias: representação e percurso (recursivo) • Aplicação: árvores de busca (com inserção e remoção) • Árvores binárias de busca balanceadas • Fila de prioridade (heap) implementação com vetor e heapsort • Árvores B e generalizações • Introdução ao espalhamento (hashing): conceito, implementação com listas ligadas. • Grafos: conceito, representação por matrizes e listas ligadas • Percurso de grafos em largura e profundidade

Tarefas Haverá diversas tarefas a serem entregues durante o semestre, praticamente toda semana. Todas as tarefas terão um prazo total de 7 dias ou mais (para a primeira entrega — veja abaixo) e deverão ser feitas **individualmente, exceto quando explicitamente dito o contrário**.

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. A pessoa 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. Adicione mensagens de commit adequadas descrevendo as modificações realizadas. Tarefas que não tenham histórico de commits ou submetidas por outros meios serão desconsideradas.

4. A cada pequena parte implementada, deve-se testar o programa. Para isso, deve-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, a pessoa 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 estudante.

As tarefas serão corrigidas pelo sistema de correção automática e por uma monitora ou um monitor. Até o final do semestre, o professor irá verificar as correções e poderá revisar a nota das tarefas para cima ou para baixo. Uma vez terminada a tarefa e após ela ter passado nos testes automáticos, deve-se solicitar a correção clicando-se no botão correspondente da interface de notas (com exceção das tarefas que forem marcadas para correção automática apenas). As(os) estudantes são encorajados a mostrar e conversar sobre suas dúvidas e seus códigos com as(os) monitores.

A nota da tarefa será proporcional ao número de casos de teste resolvidos. Porém, a nota pode sofrer descontos de acordo com a qualidade do programa apresentado (podendo ser descontado até 2 pontos do total de 10). Caso o programa submetido não satisfaça os critérios estabelecidos no seu enunciado, ele terá a nota zerada.

Cada tarefa terá uma data para a entrega da solução. Isto é o que chamamos de **primeira chance**. Porém, na maioria das tarefas haverá uma **segunda chance**[1], em uma data determinada, para entregar a tarefa não entregue ou aumentar a nota da tarefa entregue. Porém, a nota obtida terá um desconto de 20% caso o estudante não tenha uma nota no mínimo 5 na primeira chance. Exemplificando:

[1]Note que para algumas tarefas, pode não haver a segunda chance, principalmente para aquelas com prazo de entrega próximo do final do semestre.

- Uma pessoa que tenha obtido nota 5 (ou mais) na primeira chance poderá entregar a tarefa novamente, podendo obter nota até 10.
- Uma pessoa que tenha obtido nota 4.9 (ou menos) na segunda chance poderá entregar a tarefa novamente, mas a nota final será no máximo 8.
- Uma pessoa que não entregou a tarefa na primeira chance, poderá entregar a tarefa, mas a nota final será no máximo 8.

Caso a pessoa falhe em aumentar a nota, ela continuará com a nota original, isto é, a nota não será diminuída da primeira chance para a segunda chance. Cada tarefa será corrigida no máximo uma vez na primeira chance e no máximo uma vez na segunda chance.

Formalmente, cada tarefa i terá uma nota ℓ_i^1 para a solução entregue na primeira chance (sendo zero, caso não for entregue) e uma nota ℓ_i^2 para a solução entregue na segunda chance (novamente sendo zero, caso não for entregue). A nota ℓ_i da tarefa i será, portanto,

$$\ell_i = \begin{cases} \max(\ell_i^1, \ell_i^2), & \text{se } \ell_i^1 \geq 5, \\ \max(\ell_i^1, 0{,}8 \cdot \ell_i^2) & \text{caso contrário.} \end{cases}$$

Cada tarefa i terá um peso P_i^ℓ dependendo da dificuldade. A média ML das tarefas será calculada como a média ponderada das tarefas, isto é,

$$ML = \frac{\sum_i P_i^\ell \ell_i}{\sum_i P_i^\ell}$$

As correções das tarefas serão entregues em até 15 dias após a data estipulada para a primeira ou segunda chance.

Testes Durante o semestre, vários testes (por volta de um por semana) serão propostos na página da disciplina no Google Sala de Aula. Tais testes terão um prazo máximo para serem cumpridos. A correção dos testes será feita automaticamente pelo Google Sala de Aula. Até a data de entrega, a pessoa poderá submeter a resposta quantas vezes quiser, sendo considerada a última submissão.

Cada teste i terá um peso P_i^t dependendo da dificuldade. A média MT dos testes será calculada como a média ponderada das notas t_i dos tarefas, isto é,

$$MT = \frac{\sum_i P_i^t t_i}{\sum_i P_i^t}$$

Note que não há segunda chance para os testes, seus prazos são finais.

Avaliação Pré-Exame

A média M, antes do exame, será calculada da seguinte forma:

$$M = \begin{cases} \min(ML, MT), & \text{se } ML < 5 \text{ ou } MT < 5 \quad (1) \\ \dfrac{9ML + MT}{10}, & \text{caso contrário.} \quad (2) \end{cases}$$

Exame

Caso a pessoa tenha média $2{,}5 \leq M < 5{,}0$ e pelo menos 75% de frequência, ela poderá, opcionalmente, fazer um exame final. O exame final consiste em entregar tarefas e testes propostos durante o semestre até **10/12/2024**. O cálculo da média é feito pela mesma fórmula acima (para M), após a atualização das notas das tarefas e testes de acordo com as regras abaixo.

Tarefas Cada tarefa terá uma nota ℓ_i^e atribuída para a entrega realizada durante o período de exame, sendo esta zero caso não seja entregue. A nota ℓ_i da tarefa i após o exame é atualizada pela seguinte fórmula:

$$\ell_i := \max(\ell_i, 0{,}5 \cdot \ell_i^e)$$

Portanto, o aluno pode entregar a tarefa novamente, mas a nota final será no máximo 5.

Testes Cada teste terá uma nota t_i^e atribuída para a entrega realizada durante o período de exame, sendo esta zero caso não seja entregue. A nota t_i do teste i após o exame é atualizada pela seguinte fórmula:

$$t_i := \max\{t_i^e, t_i\}.$$

Portanto, o aluno pode entregar os testes novamente, podendo obter a nota em sua totalidade.

Nota Final e Aprovação

Caso a pessoa não tenha realizado o exame, sua nota final F será a média M calculada antes do exame. Caso tenha realizado o exame, sua nota final F **será o mínimo entre** 5 **e a média** M atualizada, conforme descrito anteriormente.

A pessoa estará aprovada caso sua nota final F seja maior ou igual a 5,0 e tenha pelo menos 75% de presença e estará reprovada caso contrário.

Fraudes

Qualquer tentativa de fraude (plágio, atestado falso, assinar lista de presença por colegas, abandonar aula após a chamada ou assinar lista, usar bibliotecas não permitidas, copiar quaisquer trechos da internet sem autorização expressa, mostrar ou distribuir tarefa de programação individual, cola independentemente de origem, consulta a material proibido, tentar burlar testes automáticos, etc.), os envolvidos serão reprovados com nota 0 e será registrada a ocorrência no histórico escolar. Fique atento:

- Cada pessoal é responsável por manter seguros os arquivos de seu repositório. Não compartilhe sua senha nem deixe cópias de arquivos em computadores compartilhados.

- É proibido utilizar ajuda de terceiros (colegas, amigos, professores particulares, etc.) para realizar as tarefas sem autorização expressa do professor da disciplina.
- 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 submetidos 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.

Caso a pessoa realize uma fraude em uma tarefa ou teste e se arrependa, ela deve entrar em contato imediatamente com o professor explicando o que ocorreu e quem foram os envolvidos.

- Nesse caso, a penalidade será obter nota zero nas atividades envolvidas na fraude.
- Tal atitude só será válida se ocorrer antes do professor detectar e acusar a fraude.
- A pessoa não ficará imune a ser reprovada com nota final zero por outras fraudes existentes, apenas pela fraude declarada.
- Outras pessoas participantes da fraude que não se manifestarem serão enquadradas pela regra da nota final zero descrita anteriormente.
- Não serão aceitas declarações de arrependimento que não explicite todas as pessoas envolvidas na fraude.

A pessoa também pode se arrepender de fraudes que não envolvam tarefas ou testes, mas que tenham ocorrido durante a disciplina. O professor irá avaliar cada caso individualmente, levando em conta o arrependimento da pessoa na punição.

A pessoa pode, a qualquer momento, contatar o professor, inclusive de maneira anônima, para esclarecer se determinado comportamento é considerado fraude ou não.

Bibliografia O professor não seguirá um livro texto específico, entretanto, os livros abaixo cobrem o que será visto em aula. Em particular, as principais referências são os livros 1 e 2 da seguinte lista.

1. R. Sedgewick, Algorithms in C. Addison-Wesley ,1990.
2. T. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Algoritmos — Teoria e Prática. Campus, 2002.
3. A. V. Aho, J. E. Hopcroft, J. Ullmann. Data Structures and Algorithms. Addison-Wesley, 1983.
4. W. Celes, R. Cerqueira, J. L. Rangel. Introdução a Estruturas de Dados. Campus, 2004.

5. M. J. Folk e B. Zoellick. File Structures. Addison-Wesley, 1992.

6. F. Lorenzi, P. N. de Mattos, T. P. de Carvalho. Estruturas de Dados. Thomson, 2007.

7. S. L. Pereira. Estruturas de Dados Fundamentais. Érica, 1996.

8. E. M. Reingold e W. J. Hanson, Data Structures. Little-Brown (1983).

9. J. L. Szwarcfiter e L. Markenzon. Estruturas de Dados e Seus Algoritmos. Editora LTC (1994).

10. D. E. Knuth, The Art of Computer Programming, Vol I: Fundamental Algorithms. Addison-Wesley (1978).

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.