Apresentação da disciplina

Disciplina: Projeto e Análise de Algoritmos.
Professor: Críston Souza (criston@ufc.br).
Conteúdo: `moodle2.quixada.ufc.br`

Sobre a disciplina

Análise teórica e empírica de eficiência de algoritmos, e principais técnicas de projeto de algoritmos.

Sobre o professor

Sou professor da UFC-Quixadá desde 2010, lecionando principalmente FUP, PAA e Probabilidade e Estatística. Atualmente sou vice-coordenador do Mestrado em Computação. Fiz doutorado na PUC-Rio na área de Algoritmos e Otimização Combinatória, e portanto esta disciplina tem relação direta com minha área de pesquisa.

Chamada, cadastro Moodle e apresentação dos alunos

Para cada aluno na chamada:

1. **E-mail cadastrado no Moodle**.
2. **Condições de estudo remoto** (para aulas síncronas e atividades): disponibilidade de computador, Internet de qualidade suficiente, local apropriado (silêncio, sem distrações).
3. Quais **estruturas de dados** conhece bem (como funciona e complexidade das operações)? Ex.: lista encadeada, pilha, fila, heap, árvore binária de busca, tabela hash, union-find.
4. Participa, ou já participou, do treinamento para a **maratona de programação**?
5. Já desenvolveu ou deu manutenção em algum **software com problema de desempenho**? Qual foi o problema e qual foi a solução?

Sistemas acadêmicos

SIPPA: **controle de faltas, plano de ensino** (ementa, objetivo, cronograma de aulas e avaliações, bibliografia).

Moodle: **material** didático, **exercícios**, **notas durante** o semestre.

SI3: **avisos** (enviados para o email cadastrado lá), **notas e faltas ao final** do semestre.

Plano de ensino no SIPPA

Justificativa, ementa, objetivos, cronograma de aulas, metodologia, atividades, avaliações, bibliografia.

Como será o trabalho final?

A turma será dividida em equipes com cerca de 3 alunos. Cada equipe vai **escolher um problema e dois algoritmos distintos para este problema**, portanto os algoritmos devem receber as **mesmas entradas e produzir as mesmas saídas**. Caso não tenha algum problema em mente, consulte o catálogo de problemas e algoritmos disponível na parte 2 do livro do Skiena, que pode ser baixado no site da biblioteca (Pergamum).

Após definir o problema e os dois algoritmos, cada equipe vai realizar, para cada algoritmo, uma **análise teórica de complexidade** e uma **análise empírica de eficiência**. Com base nos resultados destas análises, a equipe vai **comparar os dois algoritmos**.

O trabalho será avaliado em **4 entregas**, todas com o mesmo peso:

1. Definição do problema e especificação dos algoritmos.
2. Análise teórica de complexidade de tempo de pior caso.
3. Análise empírica de eficiência.
4. Apresentação.

Modo remoto

Aulas serão **síncronas** no Google Meet no **horário regular** da disciplina. Farei **chamada** em cada aula síncrona. Caso tenha alguma **problema de conexão** durante a chamada, avise o quanto antes (na aula ou por email).

Aulas invertidas e organização semanal

Pela minha experiência com este disciplina, a aula expositiva tem pouco efeito no aprendizado, ou seja, o aprendizado realmente ocorre quando o aluno tenta fazer os exercícios. Por este motivo, adoto a metodologia de **aula invertida**, que consiste basicamente em **priorizar o esclarecimento de dúvidas durante as aulas**, ao invés de priorizar a exposição do conteúdo. Desta forma é necessária uma participação mais ativa do aluno, tendo que **dedicar tempo fora do horário de aula** para estudar o material e tentar fazer os exercícios, para que seja capaz de **trazer dúvidas para as aulas**.

As **aulas são organizadas em semanas**. Geralmente um novo conteúdo é discutido na última aula da semana, e um exercício sobre este conteúdo deve ser entregue até a primeira aula da semana seguinte, pois desta forma o aluno tem pelo menos uma semana para resolver as questões. A prioridade das aulas de discussão de conteúdo é tirar dúvidas sobre o novo conteúdo e sobre o enunciado das questões, e discutir a correção do exercício anterior. Nas aulas de exercício a prioridade é tirar dúvidas sobre a solução das questões, e dúvidas sobre como corrigir respostas dos colegas (laboratório de avaliação).

As **dúvidas devem ser específicas**, ou seja, caso não consiga fazer uma questão deve haver pelo menos o **esforço de elaborar boas perguntas** (cujas respostas permitam começar a resolver o problema). Perguntas muito abertas, ou afirmações de não sabe o que perguntar, serão interpretadas como baixa dedicação do aluno. Caso a dúvida seja de **conteúdo anterior a esta disciplina**, como matemática e estruturas de dados, posso orientar o aluno em um estudo paralelo deste conteúdo de base, mas uma dedicação muito maior será demandada para que consiga fazer as atividades.

O **horário limite para submissão das respostas** é 13:30 para turmas da manhã, e 19h para turmas da tarde. Os alunos sempre pedem para aumentar este prazo para meia noite, mas na prática o resultado disso não é bom, pois percebo que desta forma vários alunos deixam para começar a fazer o exercício de noite (no limite do prazo), e portanto não aproveitam o horário da aula para tirar dúvidas. Sabendo que terá pouco tempo depois da aula, o aluno precisa chegar na aula com algo já feito.

Caso o horário das aulas não seja suficiente para tirar todas as dúvidas, posso agendar **horário extra de atendimento**. Porém, caso sobre tempo livre durante as aulas, não faz sentido marcarmos horário extra. Prefiro atendimento por voz, então caso um aluno envie **dúvida por email, vou deixar para responder no horário da aula**, ou vou agendar um horário extra com o aluno.

O Moodle aceita **respostas em HTML**, e oferece um editor para que o aluno consiga formatar o texto sem precisar escrever tags HTML. O Moodle também permite escrever equações matemáticas utilizando LATEX, e embutir estas equações no texto HTML. Desta forma, **todas as equações devem ser escritas em LATEX**, para facilitar a leitura de quem vai corrigir as respostas. Para quem nunca usou LATEX, disponibilizei no Moodle um **tutorial LATEX** (feito pelo Overleaf) e a **Wiki** *"Teste de formatação das respostas"* para que vocês possam praticar e verificar o resultado da formatação. Os links para o tutorial LATEX estão disponíveis na Wiki.

Correção dos exercício

Os exercícios no Moodle serão **corrigidos pelo colegas, e conferidos por amostragem pelo professor**. As correções também **recebem nota** calculada automaticamente pelo Moodle: quanto mais parecida com a nota dada pelo colegas, maior é a nota de correção. Em cada exercício, **a resposta tem peso 8 e a correção tem peso 2**. Se o professor perceber **correções sistematicamente erradas** (por exemplo, sempre fornecer a nota máxima), o aluno ficará com **zero em todas as notas de correção**.

Na correção de cada item, o aluno deve escolher um dos níveis na escala abaixo:

Não respondeu: resposta vazia ou sem relação com a pergunta.

Insuficiente: tentou responder, mas nenhuma contribuição significativa para a resposta foi fornecida.

Parcialmente correto: alguns aspectos estão corretos, mas parte significativa da resposta está errada ou foi omitida.

Totalmente correto, porém confuso: apenas aspectos pouco significativos estão errados ou foram omitidos, mas a apresentação está confusa ou com vários erros de ortografia ou pontuação.

Totalmente correto e claro: nenhum problema significativo encontrado.

Sobre plágio

Os alunos **podem discutir as respostas** dos exercícios com o professor e com os colegas. **Podem também pesquisar** na Internet e em livros. Porém, o aluno **deve escrever a resposta com suas palavras**, ou seja, sem copiar conteúdo. Como cada pessoa tem um estilo próprio de escrita, é fácil perceber a presença de cópia. Caso a cópia fique evidente, o aluno ficará com **zero em todas as respostas e correções** (não apenas o exercício copiado). Quem forneceu a resposta também ficará com zero. Portanto, para evitar problemas, **nunca copie nem forneça suas respostas por escrito** para os colegas.

Dificuldades e sugestões

A principal dificuldade é o fato desta disciplina **utilizar conhecimentos de disciplinas anteriores**, que muitas vezes não foram suficientemente amadurecidos pelos alunos. As principais deficiências são em **programação, estruturas de dados e matemática discreta**. Reserve tempo adicional de estudo para revisar conteúdos anteriores se perceber que isto está dificultando o acompanhamento da disciplina.

Uma evidência do efeito da **boa base** é o fato dos alunos envolvidos no **treinamento da maratona** acharem a disciplina mais fácil. Infelizmente a maratona dá a **ilusão** de que o **conhecimento teórico** de prova de corretude e análise de complexidade **não é importante**. Por que isso ocorre? Na maratona existe um sistema que diz se seu código está correto, então você não precisa provar corretude. O sistema diz também que o tempo de execução não está bom, então você não precisa fazer análise de complexidade. No mundo real não existe um sistema que diz se o algoritmo está correto e que o tempo de execução é bom o bastante.

É muito importante **estudar o material antes da aula**, incluindo a **leitura dos enunciados**. O tempo que o professor vai dedicar a cada assunto depende das perguntas que serão feitas. Se estiver achando um assunto difícil, **elabore várias perguntas específicas**. Assim o professor poderá lhe ajudar nas atividades.

Dúvidas frequentes

Posso solicitar abono de faltas por problemas pessoais?
O aluno tem direito a faltar até 25% do curso (16 horas-aula = 8 aulas) por motivos pessoais, portanto **não é necessário** pedir abono de faltas. Caso tenha algum problema que provoque mais de 8 faltas, verique junto à secretaria se seu caso se encaixa em alguma situação de **regime especial**. Fique atento ao fato de que faço **chamada em todas as aulas**, incluindo provas e apresentação de trabalhos. Toda semana lanço as faltas no SIPPA. Tenha o **hábito de acompanhar suas faltas**.

Motivação para o estudo da disciplina

O que estudamos e por que este conteúdo é útil?

O **projeto de algoritmos** é geralmente uma **atividade criativa**, mas algumas **técnicas de projeto** funcionam em muitas situações e portanto podem servir de **ferramenta auxiliar** do projeto de algoritmos. Nesta disciplina praticamos várias destas técnicas de projeto, como divisão e conquista, programação dinâmica e algoritmos gulosos.

Uma vez projetado o algoritmo, precisamos **avaliar sua eficiência**. Na **análise teórica de eficiência** abstraímos característica particulares do computador e da linguagem de programação, e determinamos uma função de **complexidade** que relaciona o tamanho da entrada do algoritmo com seu **tempo de execução** ou **consumo de memória**. Além disso, costuma-se focar na eficiência do algoritmo para **entradas grandes**, o que nos permite simplificar a análise através da **notação assintótica**. Na disciplina veremos também técnicas para realizar **análise empírica de eficiência**, que consiste em tirar conclusões sobre a eficiência com base em medições do tempo de execução.

O projeto e a análise caminham juntos, pois muitas vezes precisamos **refinar o algoritmo para torná-lo cada vez mais eficiente**. Você projeta uma primeira versão e determina sua complexidade; em seguida percebe como baixar a complexidade e projeta novamente, assim sucessivamente. Quando o processamento é grande, a **complexidade do algoritmo tem impacto maior** no tempo de execução que a velocidade da máquina, a linguagem de programação e a experiência do programador. Para tentar tornar o algoritmo mais eficiente você pode aplicar as técnicas de projeto estudadas, ou buscar propriedades do problema que permitam isso.

Do ponto de vista teórico, um algoritmo é considerado **eficiente** quando sua complexidade de tempo é limitada por um polinômio (os chamados **algoritmo polinomiais**). Já na prática, um algoritmo é eficiente quando é **viável esperar a execução terminar** quando fornecemos entradas reais. Geralmente existe correspondência entre eficiência teórica e prática, embora tenhamos algumas exceções importantes.

Para muitos problemas não conhecemos algoritmo eficiente. Uma classe grandes de problemas, os **NP-completos**, enquadra-se nesta situação. Se você provar que um dado problema é NP-completo, então é **improvável (embora não impossível) que exista algoritmo eficiente** para ele. A vantagem de saber provar que um problema é NP-completo é **não perder tempo buscando um algoritmo eficiente** para o problema. Na disciplina veremos como provar que um problema é NP-completo.

Em quais situações este conhecimento pode ser aplicado?

Várias áreas de **pesquisa em computação propõem e avaliam novos algoritmos** (ex.: otimização, redes, IA, BD). É importante mostrar que estes novos algoritmos são computacionalmente **viáveis (eficientes)**. Por esta razão, programas de **pós-graduação** costumam exigir que os alunos façam PAA. **Empresas** que realizam pesquisa ou que resolvem problemas novos têm esta mesma necessidade.

Para **empresas** que precisam **processar grande volume de dados** ou realizar tarefas com **grande demanda computacional** (ex.: problemas combinatórios), algoritmos mais eficientes produzem grande **economia de hardware** (processamento e/ou memória). Esta deve ser a razão de grandes empresas, como Google, avaliarem o conhecimento de PAA nas **entrevistas de emprego**. Abaixo alguns exemplos deste tipo situação:

- BD de empresas com grande base de clientes (comércio eletrônico, telefonia, energia, água).
- BI (data mining, data warehouse): muitos registros operacionais.
- Internet (comércio eletrônico, recomendação, análise de redes sociais).
- Planejamento de produção/logístico: muitas configurações de planta de produção, controle de estoque, rotas de distribuição.