PUC Minas

Lista 8 – Técnicas de projeto de algoritmos

INFORMAÇÕES DOCENTE						
CURSO: ENGENHARIA DE SOFTWARE	**DISCIPLINA:** FUNDAMENTOS DE PROJETO E ANÁLISE DE ALGORITMOS	**TURNO**	**MANHÃ**	**TARDE**	**NOITE**	**PERÍODO/SALA:** 5º
					x	
PROFESSOR (A): João Paulo Carneiro Aramuni						

Lista 8

Força bruta e pesquisa exaustiva

1) Dado um problema de agendamento de reuniões entre n pessoas com diferentes disponibilidades, por que a aplicação direta de força bruta pode ser inviável em larga escala? Proponha uma alternativa e justifique a escolha.

2) Em que cenários o uso de força bruta pode ser preferível mesmo em projetos de engenharia de software profissionais?

3) Compare a viabilidade de força bruta na resolução de um quebra-cabeça como Sudoku com a abordagem de backtracking. Quais são os trade-offs envolvidos?

Redução e transformação

1) Um engenheiro de software está projetando um sistema de detecção de fraudes em transações financeiras. Ele propõe reduzir o problema a um grafo, onde cada transação é um nó, e arestas representam semelhanças suspeitas (mesmo IP, horário, valor etc.). O objetivo é identificar subconjuntos densos de transações potencialmente fraudulentas. Explique como essa transformação pode beneficiar a análise e quais técnicas de grafos podem ser aplicadas. Quais cuidados devem ser tomados ao realizar essa modelagem?

2) Em algoritmos de recomendação, problemas complexos de seleção de conteúdo podem ser reduzidos a problemas de maximização de cobertura ou de mochila. Explique os cuidados e decisões que um engenheiro deve tomar ao escolher a transformação adequada. Cite um possível risco associado.

3) Você foi encarregado de otimizar um processo de empacotamento de arquivos para backup. Ao analisar o problema, percebe que ele pode ser reduzido ao problema de bin packing. Como essa redução pode orientar suas decisões de implementação, e quais vantagens ela oferece em relação a tentar resolver o problema diretamente?

Divisão e conquista

1) Considere o problema de multiplicação de duas matrizes muito grandes.

Por que a abordagem de divisão e conquista (como Strassen) pode ser mais eficiente do que a multiplicação tradicional?

2) Um aluno propôs usar divisão e conquista para resolver a contagem de palavras em um texto massivo (100GB). Isso faz sentido? Justifique.

3) Ao implementar um algoritmo baseado em divisão e conquista, quais aspectos de desempenho devem ser analisados além da complexidade assintótica?

Decrementar para conquistar

1) Considere um algoritmo recursivo que resolve um problema de n elementos sempre chamando a si mesmo para n - 1. Em que condições essa estratégia pode ser mais cara que outras abordagens?

2) Compare a abordagem "decrementar para conquistar" com a técnica iterativa de cauda (*tail recursion*) em termos de eficiência e legibilidade de código.

3) Como um engenheiro de software pode avaliar quando não vale a pena aplicar o paradigma de "decrementar para conquistar"?

Retrocesso e poda

1) Em um sistema de recomendação de produtos com várias regras e preferências, por que um algoritmo de retrocesso com poda pode ser mais eficiente do que força bruta?

2) Quais estratégias você adotaria para implementar a poda de forma eficiente em um algoritmo de busca por solução de quebra-cabeças?

3) Retrocesso sempre encontra a solução ótima? Justifique sua resposta com base em algum cenário prático.

Algoritmos gulosos

1) Você está construindo um serviço de alocação de anúncios em tempo real, onde cada anúncio tem um valor e uma janela de tempo disponível. Um colega sugere um algoritmo guloso para sempre escolher o anúncio mais valioso que “cabe” na agenda. Quais critérios você usaria para avaliar se essa abordagem resultará em uma solução próxima da ideal? E se não for o caso, qual alternativa proporia?

2) Compare a eficácia de algoritmos gulosos em três contextos distintos:
(a) compactação de arquivos,
(b) roteamento de pacotes em redes,
(c) agendamento de aulas em uma universidade.
Quais são as limitações e vantagens em cada um?

3) Você desenvolve um sistema de logística para entregas com janelas de tempo. O cliente quer minimizar o tempo total de deslocamento. Por que um algoritmo guloso que sempre escolhe o destino mais próximo pode não gerar a melhor rota? Qual seria uma abordagem mais robusta?

Programação dinâmica

1) Explique por que a programação dinâmica é adequada para problemas com subproblemas sobrepostos. Dê um exemplo real em engenharia de software.

2) Um colega propõe resolver o problema de caminhos mínimos em grafo com programação dinâmica. Em que condições essa abordagem é vantajosa sobre Dijkstra?

3) Você precisa projetar um sistema de sugestão de texto com base em correções anteriores (autocorreção). Como a programação dinâmica pode ser usada nesse contexto?
