

Lista 7 – Técnicas de projeto de algoritmos

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

Lista 7 - Gabarito

Força bruta e pesquisa exaustiva

- Qual das alternativas melhor descreve a abordagem de força bruta?
 - Utiliza heurísticas para encontrar soluções aproximadas rapidamente.
 - Divide o problema em subproblemas menores e resolve cada um recursivamente.
 - Tenta todas as possibilidades até encontrar a solução correta.
 - Usa memoização para evitar cálculos repetidos.
 - Constrói soluções passo a passo, escolhendo a melhor opção local.

Resposta correta: C

Explicação: A abordagem de força bruta tenta todas as possibilidades possíveis para resolver um problema, sem utilizar heurísticas ou otimizações.

- Qual é a desvantagem mais comum de algoritmos de força bruta?
 - São muito difíceis de implementar.
 - Têm eficiência muito baixa em problemas com espaço de busca grande.
 - Produzem soluções erradas frequentemente.
 - Utilizam técnicas avançadas de otimização.
 - Exigem conhecimento avançado de matemática.

Resposta correta: B

Explicação: O principal problema da força bruta é a ineficiência, especialmente quando o espaço de solução é muito grande, levando a um tempo de execução elevado.

- Qual dos problemas a seguir pode ser resolvido de forma viável por força bruta em pequena escala?
 - Problema do Caixeiro Viajante com 1.000 cidades
 - Multiplicação de matrizes grandes
 - Busca em lista desordenada de 10 elementos
 - Ordenação de vetor com 1 milhão de elementos
 - Busca em árvore balanceada de 10.000 elementos

Resposta correta: C

Explicação: Em listas pequenas, a busca exaustiva é viável. Nos outros casos, a complexidade inviabiliza a força bruta.

4. Em que situação a força bruta pode ser preferível?

- A) Quando se busca uma solução aproximada.
- B) Quando o tempo de execução não é crítico e a implementação precisa ser simples.
- C) Quando o problema tem estrutura recursiva.
- D) Quando a solução depende de decisões gulosas.
- E) Quando se deseja evitar o uso de muita memória.

Resposta correta: B

Explicação: A força bruta é simples de implementar e pode ser usada quando eficiência não é prioridade.

5. O que caracteriza a pesquisa exaustiva em comparação com outras técnicas?

- A) Sempre encontra a solução ótima.
- B) Utiliza buscas em largura ou profundidade com heurísticas.
- C) Divide o problema em subproblemas disjuntos.
- D) Usa memoização para acelerar o processo.
- E) Produz soluções aproximadas de forma eficiente.

Resposta correta: A

Explicação: A pesquisa exaustiva analisa todas as possibilidades, garantindo que a melhor solução será encontrada, embora com alto custo computacional.

Redução e transformação

1. O que significa reduzir um problema A a um problema B?

- A) Resolver A diretamente usando força bruta.
- B) Substituir A por B para simplificar os dados.
- C) Transformar instâncias de A em instâncias de B de modo que soluções de B ajudem a resolver A.
- D) Dividir A em várias instâncias de B.
- E) Calcular o tempo de execução de A em função de B.

Resposta correta: C

Explicação: A redução de A para B consiste em transformar A de forma que resolver B ajude a resolver A, geralmente para mostrar equivalência de dificuldade.

2. Em Engenharia de Software, a técnica de redução entre problemas é usada para:

- A) Garantir que todas as funcionalidades do sistema sejam implementadas em tempo linear.
- B) Mapear formalmente um problema de verificação de propriedades de software para um problema conhecido, permitindo usar soluções estabelecidas.
- C) Automatizar a geração de código-fonte a partir de diagramas UML.
- D) Simplificar o processo de integração contínua ao dividir tarefas entre equipes.
- E) Otimizar o desempenho do sistema distribuído por meio de balanceamento de carga.

Resposta correta: B

Explicação: A redução é usada para formalizar problemas complexos (como verificação de propriedades, model checking, ou testes formais) transformando-os em problemas já estudados e resolvidos, permitindo aplicar técnicas e algoritmos consolidados para validar o software.

3. O que é uma transformação polinomial entre problemas?

- A) Um algoritmo que ordena instâncias de problemas.
- B) Um processo que reduz um problema a outro em tempo exponencial.
- C) Uma redução feita em tempo polinomial.
- D) Um método de resolução recursiva.
- E) Uma estratégia gulosa para problemas NP.

Resposta correta: C

Explicação: A transformação polinomial é uma redução feita com custo computacional limitado a tempo polinomial.

4. Qual é a importância da redução na prova de que um problema é NP-completo?

- A) Ela otimiza a solução do problema.
- B) Permite que o problema seja resolvido por busca binária.
- C) Mostra que o problema pode ser resolvido por algoritmos gulosos.
- D) Estabelece que o problema é tão difícil quanto outros problemas NP-completos.
- E) Permite utilizar algoritmos de ordenação eficientes.

Resposta correta: D

Explicação: Reduzindo um problema conhecido NP-completo para outro, provamos que este novo problema é ao menos tão difícil quanto o primeiro.

5. Em qual das situações é comum usar transformações entre problemas?

- A) Para implementar interfaces de usuário.
- B) Para converter dados de entrada em formatos de saída.
- C) Para analisar eficiência de algoritmos paralelos.
- D) Para demonstrar relações de complexidade entre problemas.
- E) Para calcular limites inferiores de algoritmos gulosos.

Resposta correta: D

Explicação: As transformações são usadas para mostrar relações de dificuldade computacional entre problemas, como nas provas de NP-completude.

Divisão e conquista

1. Qual é o princípio fundamental da técnica de divisão e conquista?

- A) Escolher a melhor opção local em cada passo.
- B) Explorar todas as possibilidades de forma exaustiva.
- C) Dividir o problema em subproblemas independentes, resolver cada um e combinar os resultados.
- D) Utilizar uma função de avaliação heurística.
- E) Resolver o problema iterativamente até a solução surgir.

Resposta correta: C

Explicação: A técnica divide o problema original em subproblemas menores, resolve cada um (geralmente de forma recursiva) e combina os resultados para obter a solução final.

2. Qual das alternativas é um exemplo clássico de algoritmo baseado em divisão e conquista?

- A) Dijkstra
- B) Busca binária
- C) Força bruta
- D) Algoritmo de Prim
- E) Guloso de Huffman

Resposta correta: B

Explicação: A busca binária divide repetidamente o intervalo de busca pela metade, aplicando divisão e conquista para reduzir o problema a instâncias menores.

3. Qual vantagem os algoritmos de divisão e conquista geralmente oferecem?

- A) Redução de memória ao custo de tempo exponencial.
- B) Melhor desempenho em problemas com soluções aproximadas.
- C) Paralelização facilitada e bom desempenho assintótico.
- D) Soluções mais simples que algoritmos gulosos.
- E) Garantia de encontrar a solução ótima em todos os casos.

Resposta correta: C

Explicação: Como os subproblemas são independentes, os algoritmos de divisão e conquista são naturalmente paralelizáveis e apresentam bom desempenho assintótico.

4. Em qual situação a técnica de divisão e conquista pode ser ineficiente?

- A) Quando os subproblemas são idênticos.
- B) Quando o custo de combinar os resultados é muito alto.
- C) Quando o problema não tem solução ótima.
- D) Quando não há memória suficiente.
- E) Quando se deseja usar programação dinâmica.

Resposta correta: B

Explicação: Se o custo de combinar os resultados das soluções parciais for alto, o ganho da divisão pode ser anulado, tornando a técnica ineficiente.

5. Qual das operações abaixo é típica da etapa de conquista em algoritmos de divisão e conquista?

- A) Escolher o menor elemento em uma lista.
- B) Aplicar força bruta nos dados.
- C) Combinar os resultados das chamadas recursivas.
- D) Dividir a entrada em partes iguais.
- E) Armazenar resultados em uma tabela para reuso.

Resposta correta: C

Explicação: A etapa de conquista consiste em unir os resultados obtidos dos subproblemas para formar a resposta final ao problema original.

Decrementar para conquistar

1. O que caracteriza a técnica de “Decrementar para Conquistar”?

- A) Dividir o problema em partes iguais e independentes.
- B) Reduzir o tamanho do problema em pequenas unidades até chegar a um caso base.
- C) Explorar todas as possibilidades de solução de forma exaustiva.
- D) Escolher a melhor decisão local em cada passo.
- E) Utilizar programação dinâmica para armazenar subproblemas.

Resposta correta: B

Explicação: Nessa técnica, o problema é reduzido gradualmente, geralmente em uma unidade por chamada recursiva, até que um caso trivial (base) possa ser resolvido diretamente.

2. Qual das opções é um exemplo clássico de algoritmo que usa a técnica de “Decrementar para Conquistar”?

- A) Merge Sort
- B) Quick Sort
- C) Busca Binária
- D) Cálculo fatorial recursivo
- E) Programação dinâmica de Fibonacci

Resposta correta: D

Explicação: O fatorial recursivo é um exemplo típico: $f(n) = n \times f(n-1)$, com o caso base $f(0) = 1$. A cada chamada, o problema é reduzido em uma unidade.

3. Qual é uma diferença entre as abordagens “Dividir e Conquistar” e “Decrementar para Conquistar”?

- A) A primeira usa estruturas iterativas; a segunda, recursão.
- B) “Decrementar para Conquistar” aumenta o tamanho do problema; “Dividir e Conquistar” reduz.
- C) A primeira cria múltiplos subproblemas; a segunda reduz o problema em uma unidade.
- D) Ambas sempre usam força bruta.
- E) Não há diferença significativa entre elas.

Resposta correta: C

Explicação: “Dividir e Conquistar” divide o problema em vários subproblemas; “Decrementar para Conquistar” reduz o problema em uma única unidade por passo.

4. Em que situação a técnica “Decrementar para Conquistar” é mais apropriada?

- A) Quando o problema não possui casos base.

- B) Quando o problema se reduz naturalmente por unidades menores até uma solução trivial.
- C) Quando o problema deve ser resolvido com alta paralelização.
- D) Quando há muitos subproblemas sobrepostos.
- E) Quando é possível evitar chamadas recursivas.

Resposta correta: B

Explicação: Essa técnica funciona bem quando o problema pode ser resolvido passo a passo até alcançar um caso trivial, como nos algoritmos de contagem, potência ou fatorial.

5. O algoritmo abaixo é um exemplo de qual técnica?

```
def soma_natural(n):  
    if n == 0:  
        return 0  
    return n + soma_natural(n - 1)
```

- A) Programação dinâmica
- B) Dividir para conquistar
- C) Decrementar para conquistar
- D) Algoritmo guloso
- E) Pesquisa exaustiva

Resposta correta: C

Explicação: A função `soma_natural` reduz o problema passo a passo ($n \rightarrow n-1$) até atingir o caso base ($n == 0$), caracterizando “Decrementar para Conquistar”.

Retrocesso e poda

1. Qual é o princípio central da técnica de retrocesso (backtracking)?

- A) Explorar todas as soluções paralelamente.
- B) Construir a solução passo a passo e retroceder ao detectar um caminho inválido.
- C) Utilizar estruturas de dados em árvore balanceada.
- D) Resolver subproblemas sobrepostos com memoização.
- E) Escolher sempre a melhor opção local.

Resposta correta: B

Explicação: O retrocesso constrói soluções incrementais e, ao identificar que uma escolha leva a um impasse, volta (retrocede) e tenta outra opção.

2. Em qual dos seguintes problemas o retrocesso é frequentemente utilizado?

- A) Ordenação de um vetor
- B) Busca binária
- C) Problema das N rainhas
- D) Multiplicação de matrizes
- E) Cálculo de média de números

Resposta correta: C

Explicação: O problema das N rainhas envolve escolhas sucessivas com restrições e é clássico para aplicação de backtracking, testando e voltando quando necessário.

3. O que significa "poda" no contexto de algoritmos de retrocesso?

- A) Interromper o algoritmo antes de completar a busca.
- B) Ignorar partes do espaço de busca que não levam à solução.
- C) Eliminar elementos repetidos da entrada.
- D) Usar busca exaustiva até o fim.
- E) Utilizar programação dinâmica para cortar chamadas recursivas.

Resposta correta: B

Explicação: A poda consiste em evitar a exploração de caminhos que, com base em critérios definidos, não levarão a uma solução válida — otimizando o processo.

4. Qual das seguintes opções melhora a eficiência de um algoritmo de backtracking?

- A) Substituir chamadas recursivas por laços
- B) Ignorar condições de parada
- C) Aplicar poda para reduzir o espaço de busca
- D) Utilizar buscas em largura sempre que possível
- E) Evitar o uso de estruturas auxiliares

Resposta correta: C

Explicação: A aplicação de poda ajuda a eliminar caminhos infrutíferos antecipadamente, tornando a busca mais eficiente.

5. Qual alternativa melhor representa a complexidade dos algoritmos baseados em retrocesso?

- A) Geralmente têm complexidade linear
- B) Têm sempre complexidade constante
- C) Dependem do uso de estruturas como filas de prioridade
- D) Podem ter complexidade exponencial, dependendo do espaço de busca
- E) São sempre mais eficientes que algoritmos gulosos

Resposta correta: D

Explicação: Como o backtracking pode explorar muitas combinações, sua complexidade geralmente é exponencial — embora a poda ajude a reduzir o número de possibilidades exploradas.

Algoritmos gulosos

1. Qual é a principal característica de um algoritmo guloso?

- A) Resolve o problema por força bruta.
- B) Armazena soluções de subproblemas para evitar recomputação.
- C) Toma a melhor decisão local esperando que leve à melhor solução global.
- D) Divide o problema em dois subproblemas recursivos.
- E) Tenta todas as soluções possíveis.

Resposta correta: C

Explicação: Algoritmos gulosos constroem a solução passo a passo, sempre escolhendo a opção localmente mais vantajosa, esperando que essa escolha leve a uma solução ótima global.

2. Em qual dos seguintes problemas um algoritmo guloso fornece uma solução ótima?

- A) Problema do Caixeiro Viajante
- B) Problema da Mochila 0-1
- C) Soma de subconjunto
- D) Codificação de Huffman
- E) Multiplicação de cadeias de matrizes

Resposta correta: D

Explicação: A codificação de Huffman é um exemplo clássico onde o algoritmo guloso sempre gera uma árvore ótima de codificação.

3. Qual condição é necessária para que um algoritmo guloso produza a solução ótima?

- A) A função objetivo deve ser linear.
- B) O problema precisa ter subproblemas independentes.
- C) O problema deve ter a propriedade de sobreposição de subproblemas.
- D) O problema deve possuir a propriedade de escolha gulosa e subestrutura ótima.
- E) O problema deve ser resolvido por tentativa e erro.

Resposta correta: D

Explicação: Para que a abordagem gulosa funcione, o problema precisa permitir que decisões locais levem a uma solução global ótima (escolha gulosa) e que partes da solução também sejam ótimas (subestrutura ótima).

4. Um algoritmo guloso pode falhar em qual tipo de problema?

- A) Problemas com subestrutura ótima
- B) Problemas com soluções únicas
- C) Problemas que exigem comparação entre várias decisões futuras
- D) Problemas com estruturas de dados simples
- E) Problemas que utilizam busca linear

Resposta correta: C

Explicação: Algoritmos gulosos não revisitam decisões anteriores. Em problemas que exigem comparação entre decisões futuras (como a mochila 0-1), podem não encontrar a solução ótima.

5. O que distingue algoritmos gulosos de programação dinâmica?

- A) Gulosos usam recursão; programação dinâmica não.
- B) Programação dinâmica tenta soluções locais; gulosos, globais.
- C) Gulosos não armazenam subsoluções; programação dinâmica sim.
- D) Gulosos testam todas as combinações; programação dinâmica, não.
- E) Não há diferença prática entre eles.

Resposta correta: C

Explicação: A principal diferença é que algoritmos gulosos não utilizam armazenamento de subsoluções (como tabelas), enquanto programação dinâmica depende disso para evitar recomputações.

Programação dinâmica

1. Qual é o principal objetivo da programação dinâmica?

- A) Explorar todas as possibilidades do problema.
- B) Dividir o problema em várias instâncias menores sem reutilizá-las.
- C) Resolver problemas com subproblemas independentes.
- D) Armazenar resultados de subproblemas para evitar recomputações.
- E) Escolher a melhor decisão local em cada passo.

Resposta correta: D

Explicação: A programação dinâmica visa evitar cálculos repetidos armazenando os resultados de subproblemas já resolvidos (memoização ou tabulação).

2. Quando a programação dinâmica é mais eficaz?

- A) Quando o problema pode ser resolvido com uma única decisão gulosa.
- B) Quando os subproblemas são independentes.
- C) Quando o problema tem subestrutura ótima e sobreposição de subproblemas.
- D) Quando o problema pode ser dividido em instâncias completamente novas.
- E) Quando não há casos base definidos.

Resposta correta: C

Explicação: A programação dinâmica é eficaz quando o problema pode ser resolvido a partir de soluções ótimas de subproblemas (subestrutura ótima) e esses subproblemas se repetem (sobreposição).

3. Qual das alternativas abaixo representa um problema clássico resolvido com programação dinâmica?

- A) Busca em profundidade
- B) Problema das N rainhas
- C) Ordenação por seleção
- D) Cálculo da sequência de Fibonacci com memoização
- E) Algoritmo de Kruskal

Resposta correta: D

Explicação: A sequência de Fibonacci com memoização evita cálculos redundantes armazenando resultados intermediários — aplicação direta de programação dinâmica.

4. Qual é a diferença entre memoização e tabulação em programação dinâmica?

- A) Memoização resolve o problema de forma iterativa, tabulação recursiva.
- B) Memoização usa tabelas de forma explícita, tabulação não.
- C) Memoização calcula os subproblemas sob demanda, tabulação os resolve em ordem.
- D) Memoização é usada apenas em algoritmos gulosos.

E) Tabulação não armazena subsoluções.

Resposta correta: C

Explicação: Memoização é recursiva e armazena resultados sob demanda; tabulação resolve iterativamente todos os subproblemas de menor para maior.

5. Qual dessas características não está presente em um problema que pode ser resolvido com programação dinâmica?

- A) Subestrutura ótima
- B) Subproblemas que se repetem
- C) Independência completa entre subproblemas
- D) Possibilidade de armazenamento de resultados intermediários
- E) Existência de casos base

Resposta correta: C

Explicação: Se os subproblemas forem completamente independentes, não há ganho em armazenar soluções, o que inviabiliza o uso de programação dinâmica.
