



Universidade do Sul de Santa Catarina – UNISUL
Curso de Ciência da Computação
ANÁLISE DE ALGORITMOS
Professor: Max
Data: 03/05/2017

e-mail: max.pereira@unisul.br

NOME:

Roberto Alves Bento

Questões:

AValiação II

1. (1,5) Um algoritmo **A** resolve um problema de tamanho n derivando dele cinco subproblemas de tamanho $n/2$, resolvendo-os recursivamente, e então combinando suas soluções em tempo linear. Qual a complexidade desse algoritmo e qual a abordagem de programação utilizada?

2. (2,0) Analise o seguinte "jogo": um número com n dígitos é escrito em um quadro, o jogador deve então apagar d dígitos do número que está no quadro. Os números que restarem equivalem a um prêmio em dinheiro. Exemplo: número escrito no quadro: **1 2 3 1 2 3 9**, dígitos a serem apagados: **3**. Valor em dinheiro **1 2 3 1 2 3 9** = 3.239,00! Outro exemplo: número escrito no quadro: **1 0 0 0 0 0 1 9**, dígitos a serem apagados: **4**. Valor em dinheiro: **1 0 0 0 0 0 1 9** = 1.019,00!

Escreva um algoritmo utilizando a abordagem "gulosa" para resolver o problema e determine sua complexidade (Big-O).

3. (1,5) O roteamento de pacotes, em redes de computadores, pode ser modelado como um problema do caminho mínimo, cujo objetivo é obter a rota de transmissão mais rápida. O critério de seleção do próximo nó, em alguns algoritmos, é a distância mais curta do nó atual até o próximo. Esse critério é típico de qual abordagem de programação? Justifique a sua resposta.

4. (1,5) Os Números de Fibonacci são uma sequência definida como recursiva pela fórmula:

$$\begin{aligned} F(n) &= 1, \text{ se } n=1 \\ F(n) &= 1, \text{ se } n=2 \\ F(n) &= F(n-1) + F(n-2), \text{ se } n > 2 \end{aligned}$$

Escreva um algoritmo utilizando a abordagem dinâmica para calcular o n -ésimo termo da sequência de Fibonacci e determine sua complexidade (Big-O).

5. (2,0) Temos o seguinte problema: executar a soma de n números $a_1 + a_2 + \dots + a_n$. Com base na estratégia dividir e conquistar, escreva um algoritmo para dividir o problema em duas instâncias do mesmo problema (subproblemas) e determine sua complexidade (Big-O).

Soma dos primeiros $\lfloor n/2 \rfloor$ números.
Soma dos $\lceil n/2 \rceil$ números restantes.

Esta maneira é mais eficiente do que uma adição tradicional (soma sequencial de todos os n números)?

6. (1,5) Analise as afirmativas abaixo.

I. A programação dinâmica é um método ascendente que aborda um dado problema subdividindo-o em problemas mínimos, soluciona esses subproblemas, guarda as soluções parciais, combina os subproblemas e subresultados para obter e resolver os problemas maiores, até recompor e resolver o problema original.

II. A divisão e conquista é um método recursivo e, por isso, descendente que decompõe sucessivamente um problema em subproblemas independentes triviais, resolvendo-os e combinando as soluções em uma solução para o problema original.

III. Algoritmos que aplicam a estratégia da programação dinâmica nem sempre asseguram soluções ótimas, mas, para muitos problemas, as soluções são ótimas. Os problemas ideais para essa estratégia não devem ter a propriedade de subestrutura ótima.

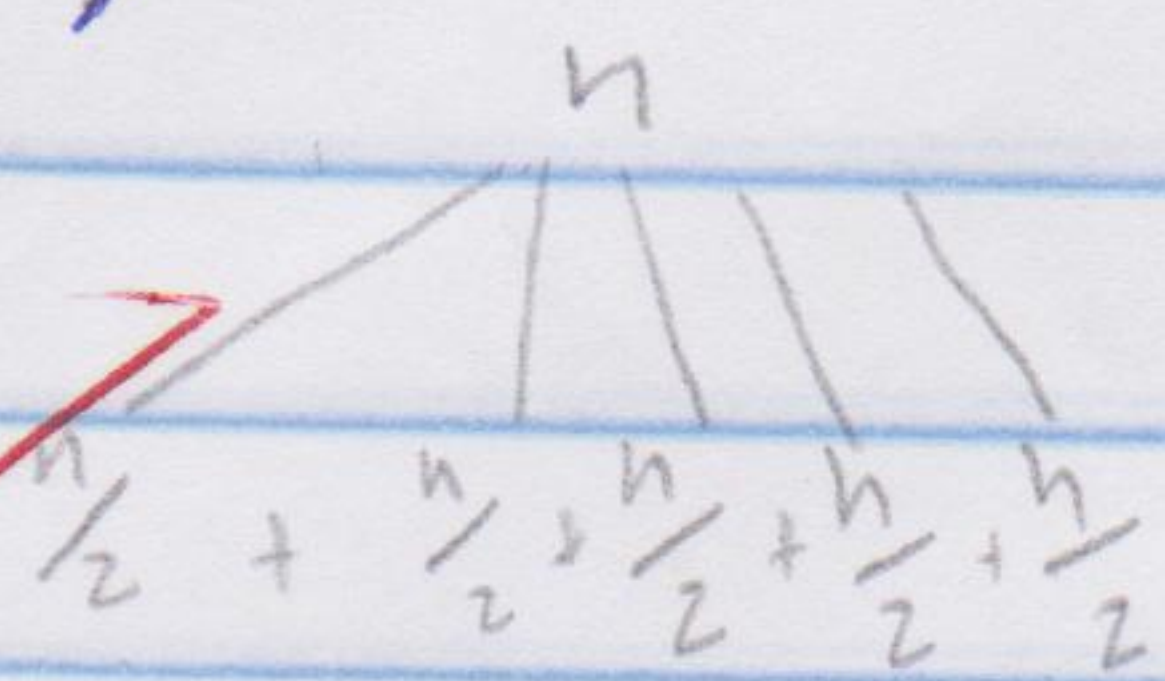
Podemos concluir que:

- ☒ a) Todas as afirmativas são verdadeiras.
- ☐ b) Todas as afirmativas são falsas.
- ☒ c) Apenas as afirmativas I e II são verdadeiras.
- ☐ d) Apenas as afirmativas II e III são verdadeiras.
- ☐ e) Apenas a afirmativa III é verdadeira.

PENSE!

1) A

Complexidade: $O(n \cdot \log n)$



Abordagem: Divisão e Conquista.

2) AlgJogo(W, n, d) {

// $W \rightarrow$ lista de números

// $n \rightarrow$ quantidades de números

// $d \rightarrow$ quantidade de dígitos removidos

$W = \text{Ordemacao_Heap_sort}(W)$ // ordena lista de forma decrescente } $O(n \log n)$

Resultado = "" // variável de saída com a resposta

For $K = 1$ to n : // realiza um loop pela lista

If $n - K < d$: // remove os d últimos dígitos da lista } $O(n)$

Resultado += $W[K]$

retorna resultado

$O(n \log n) + O(n) \Rightarrow O(n \log n)$

3) Refere-se aos Algoritmos gulosos, pois eles buscam sempre o ótimo para aquele momento, sem verificar os próximos passos, ou seja, o melhor caminho para o momento, sem verificar o restante do trajeto.

4) Resolvendo utilizando a Abundância de Dicionário.

DADOS = {}

Fib(n)

If Isset(DADOS[n]): // verifica se existe o valor armazenado

return DADOS[n]

else

// se não está armazenado, utiliza o método comum

If $n \leq 2$

X = 1

Complexidade: $O(n \log n)$

else

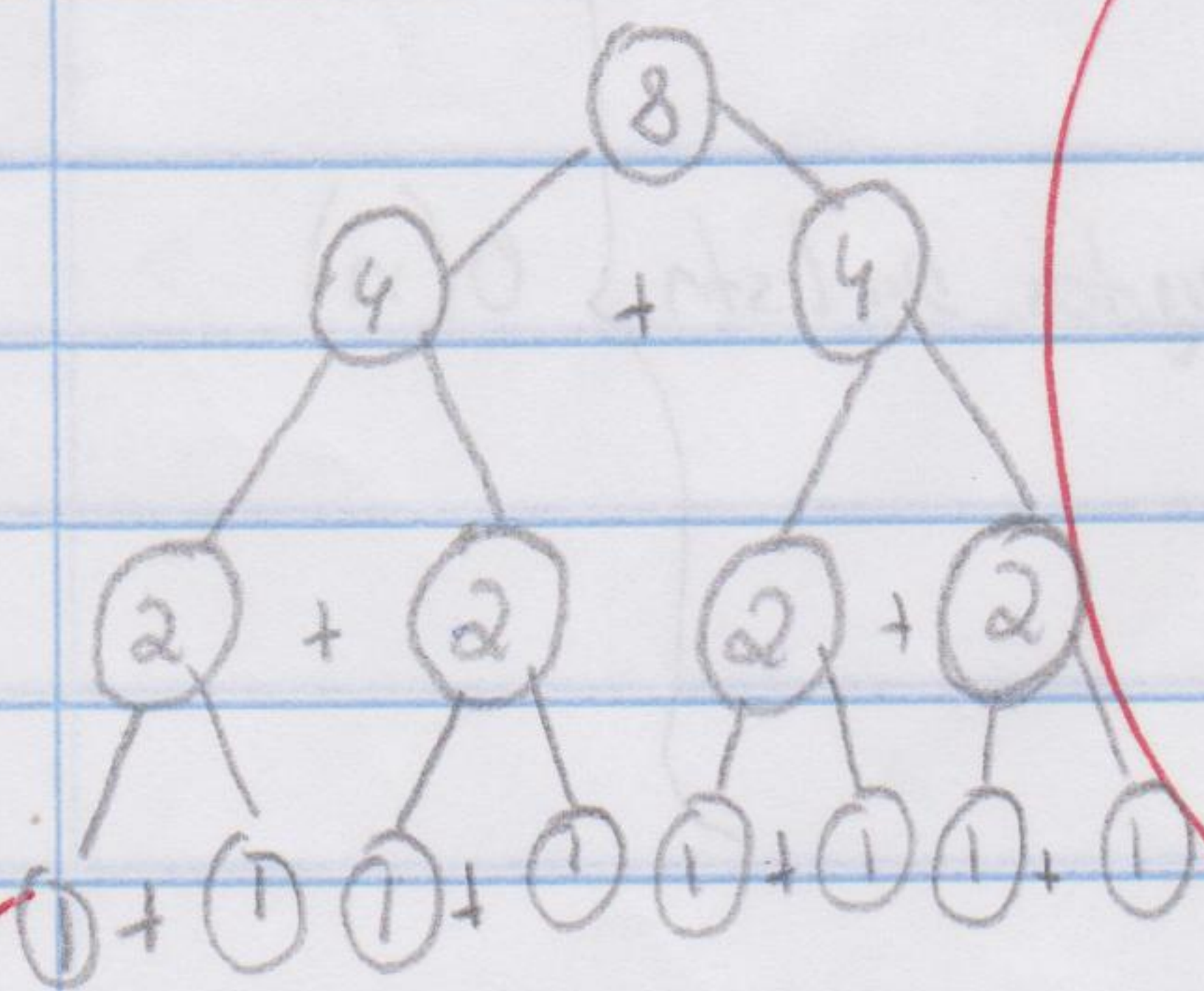
X = Fib(n-1) + Fib(n-2)

DADOS[n] = X

// Armazena o resultado para reutilizar depois.

return X

5) Ex: n = 8



Alysona(n)

If $n > 1$

$n1 = \text{Alysona}(\lfloor \frac{n}{2} \rfloor)$

$n2 = \text{Alysona}(\lceil \frac{n}{2} \rceil)$

return $n1 + n2$

return n

$O(n \log n)$

R: A Adição tradicional é mais eficiente, pois executa em uma complexidade $O(n)$, enquanto o algoritmo criado executa em uma complexidade $O(n \log n)$