Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Projeto e Análise de Algoritmos – 2º Semestre de 2018
Profª. Raquel Mini

2ª PROVA

Nome:____
Valor: 35 pontos
Todas as respostas devem ser dadas nos locais indicados.

1. (8 pontos) Considere o problema da colocação ótima de parênteses para multiplicação das seguintes matrizes A[5][14], B[14][3], C[3][10], D[10][8], E[8][50], F[50][6].

a) Execute o algoritmo e preencha os campos destacados indicados abaixo (utilize a folha em branco no final da prova para as execuções):

Matriz cost

0		360	570	2400	2640
	0		576	3540	2592
		0		1440	2340
			0		2880
				0	
					0

Matriz best

		3	3	3	3
			3	3	3
				5	6
					5

b) Qual é a colocação de parênteses com menor custo para realizar a multiplicação dessa cadeia de matrizes? Justifique sua resposta.

2. (8 pontos) Prove se as seguintes afirmativas são verdadeiras ou falsas:

a) Se **A** e **B** são dois problemas pertencentes à classe NP-Completo, então **A** é polinomialmente redutível a **B** e **B** é polinomialmente redutível a **A**. Justifique.

b) Se **A** é um problema de decisão localizado em NP, uma das formas de provarmos que **A** pertence à classe P é encontrarmos uma redução em tempo polinomial em máquina determinística de **A** para algum problema em NP-Completo. Justifique.

3. (10 pontos)

a) Apresente um algoritmo que utiliza a técnica de divisão e conquista para calcular a soma dos elementos de um vetor `A[0..n-1]` de números inteiros.

b) Considerando que a operação relevante seja o número de somas realizadas, escreva a equação de recorrência dessa versão recursiva. Utilizando o Teorema Mestre, encontre o comportamento assintótico da função de custo.

4. (9 pontos) O Quicksort é um dos algoritmos de ordenação mais utilizados na computação. A sua implementação recursiva consiste da chamada do procedimento partição que realiza `n+1` comparações com os elementos de A e de duas chamadas recursivas como mostrado abaixo:

```
Quicksort(A, p, r) {
  if (p < r) {
    q = Partition(A, p, r)
    Quicksort(A, p, q-1)
    Quicksort(A, q+1, r)
  }
}
```

onde `A` é um vetor, `p` e `r` são os índices da esquerda e direita do subvetor a ser particionado e `q` é o ponto de partição. A ordenação do vetor se dá pela chamada `Quicksort(A,1,A.length)`.

a) Considerando que a operação relevante seja o **número de comparações com os elementos de A**, mostre a fórmula fechada para a função de custo do Quicksort para a ordenação de um vetor de tamanho `n` no melhor caso. Para isso, construa e resolva a equação de recorrência (não pode ser utilizado o Teorema Mestre).

b) Para a execução das chamadas recursivas, os compiladores em geral utilizam uma pilha. A cada chamada de um procedimento recursivo, um registro de ativação é empilhado sendo posteriormente desempilhado ao final da execução do procedimento. Considerando que a operação relevante seja o **número de vezes que o registro de ativação é empilhado**, encontre a fórmula fechada para a função de custo do Quicksort no melhor caso. Para isso, construa e resolva a equação de recorrência (não pode ser utilizado o Teorema Mestre).

Fórmulas:

$\log n = \log_2^n$	$n! \approx \left(\frac{n}{e}\right)^n$	$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
$a = b^{\log_b^a}$	$\ln n = \log_e^n$	$\sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4}$
$n^{\frac{1}{\log n}} = n^{\log_n^2} = 2$	$a^{\log_b^n} = n^{\log_b^a}$	$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
$4^{\log n} = 2^{2\log n} = 2^{\log n^2} = n^2$	$2^{\log n} = n$	Para $a \neq 1$ $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$

$$1 \prec \log\log n \prec \log n \prec n^{\varepsilon} \prec n \prec n^{c} \prec n^{\log n} \prec c^{n} \prec n^{n} \prec c^{c^{n}}$$

onde ε e c são constantes arbitrárias com $0 < \varepsilon < 1 < c$

Multiplicação de uma Cadeia de Matrizes

```
for( i=1; i <= N; i++ )
    for( j = i+1; j <= N; j++ ) cost[i][j] = INT_MAX;
for( i=1; i <= N; i++ ) cost[i][i] = 0;
for( j=1; j < N; j++ )
    for( i=1; i <=  N-j; i++ )
        for( k= i+1; k <= i+j; k++ )
          {
            t = cost[i][k-1] + cost[k][i+j] +
            r[i]*r[k]*r[i+j+1];
            if( t < cost[i][i+j] )
                { cost[i][i+j] = t; best[i][i+j] = k;
                }
          }
```

Teorema Mestre

- Sejam as constantes $a \geq 1$ e $b > 1$ e $f(n)$ uma função definida nos inteiros não-negativos pela recorrência:

$$T(n) = aT(n/b) + f(n)$$

onde a fração n/b pode significar $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. A equação de recorrência $T(n)$ pode ser limitada assintoticamente da seguinte forma:

1. Se $f(n) = O(n^{\log_b a - \varepsilon})$ para alguma constante $\varepsilon > 0$, então $\boxed{T(n) = \Theta(n^{\log_b a})}$
2. Se $f(n) = \Theta(n^{\log_b a})$, então $\boxed{T(n) = \Theta(n^{\log_b a} \log n)}$
3. Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ para alguma constante $\varepsilon > 0$ e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e para n suficientemente grande, então $\boxed{T(n) = \Theta(f(n))}$