Universidade Federal do ABC
Pós-Graduação em Ciência da Computação - 14/03/2020
Prova de Seleção - Mestrado em Ciência da Computação

NOME/RG : .....

Instruções

1. Escreva com caneta o seu nome completo e o RG (ou outro documento com foto e de validade nacional, e, neste caso indicar qual documento utilizado) na **primeira** folha. Nas **demais** folhas escreva apenas o RG, pois a correção da prova será às cegas;
2. As respostas devem ser transcritas com caneta esferográfica;
3. A prova tem duração de duas horas;
4. A prova consiste de **8 questões**. Cada questão vale 1,25 ponto;
5. Só serão consideradas para correção as respostas transcritas nas folhas indicadas;
6. As questões serão corrigidas considerando corretude, rigor técnico, clareza, ortografia e gramática;
7. **Respostas sem explicação e justificativa não serão consideradas.**

Questões

Questão 1. Determine o valor da expressão:

$$L = \int_0^4 \log_2 2^x dx$$

Questão 2. No jogo de tabuleiro *Colonizadores de Catan* (*Settlers of Catan*), a produtividade de um dado terreno é proporcional à probabilidade da soma de dois dados de 6 faces ser igual ao número que está contido no terreno. Ou seja, em um determinado lançamento de dois dados, apenas os terrenos que possuirem o número que for igual ao resultado da soma dos dois dados produzirá recursos. No início do jogo, para cada terreno poderá ser atribuido qualquer número dentre os possíveis resultados da soma de dois dados, **com exceção do número 7**. Dessa forma, determine a probabilidade de cada possível resultado da soma de dois dados e, com base nessas probabilidades, determine também uma ordenação decrescente das produtividades correspondentes a essas somas.

Questão 3. Determine os valores de $x \in R$ que satisfazem a inequação:

$$\frac{x^4 - 10x^2 + 9}{x^4 - 16} \geq 0$$

Questão 4. Na implementação de um determinado algoritmo, você terá que fazer uso de uma lista. Você precisa decidir entre implementá-la como um vetor (*array*) sequencial ou como uma lista ligada. Quais fatores seriam levados em consideração para a escolha de cada uma?

Questão 5. Implemente um algoritmo que verifica se uma árvore binária é uma árvore binária de busca. Lembrando que uma árvore binária de busca é aquela em que, dado um nó n, todos os nós da sub-árvore esquerda possuem um valor menor que ele e todos os nós da sub-árvore direita possuem um valor maior que ele. Considere que a árvore contém apenas valores únicos.

Questão 6. Sejam A e B dois algoritmos que realizam a mesma tarefa. Os tempos de execução de pior caso de A e B são, respectivamente, $O(n \log_2 n)$ e $\Omega(n^{1,01})$. É possível dizer que no pior caso o algoritmo A é assintoticamente mais eficiente que o algoritmo B? Justifique sua resposta.

Questão 7. Uma palavra ou sentença é considerada um palíndromo se sua leitura da esquerda para a direita é idêntica à leitura da direita para a esquerda. Exemplos de palíndromos: OVO, OSSO, RADAR. Implemente uma função **recursiva** denominada `palindromo` que recebe como entrada um vetor de caracteres e devolve True (verdadeiro) se o vetor é palíndromo ou False (falso) caso contrário.

Questão 8. Um dos algoritmos de ordenação mais famosos é o *Quicksort*. O cerne da implementação do *Quicksort* é uma outra função (cuja implementação é dada abaixo), que tem uma única tarefa: separar um vetor em duas partes. A primeira parte (esquerda) contém todos os elementos "pequenos" e a segunda parte (direita) todos os elementos "grandes". Para determinar se um elemento é "pequeno" ou "grande", o algoritmo toma um elemento (qualquer) como base de comparação. Este elemento é chamado de *pivô*. Os elementos do vetor que forem maiores que `pivo` serão considerados grandes e os demais serão considerados pequenos.

Exemplo:

Pequenos							**Pivô**	**Grandes**	
6	2	1	7	5	4	5	7	9	8
							i		

A função abaixo rearranja um vetor $v[0..n-1]$ e devolve i tal que $v[0..i-1] \leq v[i] < v[i+1..n-1]$ para algum i em $0..n-1$ (neste caso, $v[i]$ é o pivô).

```
/* Recebe um vetor v[0..n-1] com n elementos. Rearranja os elementos do
vetor e devolve i, 0 <= i < n, tal que v[0..i-1] <= v[i] <= v[i+1..n-1] */
int separa(int v[], int n) {
    /* Toma como pivô o último elemento do vetor */
    int pivo = v[n - 1];
    int aux, k, i = 0;
    for (k = 0; k < n - 1; k++)
        if (v[k] <= pivo) {
            aux = v[i];
            v[i] = v[k];
            v[k] = aux;
            i++;
        }
    aux = v[i];
    v[i] = v[n - 1];
    v[n - 1] = aux;
    return i;
}
```

Modificando apenas uma das linhas da função `separa` fornecida acima, é possível fazer com que ela rearranje o vetor v de modo que tenhamos a primeira parte (esquerda) do vetor com todos os elementos **pares** e a segunda parte (direita) com todos os elementos **ímpares**. Realize essa modificação. Atenção: assim como a implementação original, a sua implementação não pode fazer uso de um vetor auxiliar.