Estruturas de Dados 2017/2018 – Época Normal (15 de Junho de 2018)

Versão A

Duração: 2h30m + 30m

Número mecanográfico:
Nome completo do estudante:

Grupo 1 - Fundamentos de Java

1.1. Escreva pequenos **excertos de código** para cada uma das seguintes alíneas:

a) Extrair o algarismo das dezenas de um inteiro n:

b) Criar um array v com espaço para n valores *true* ou $false$:

c) Ir buscar o número de colunas de uma matriz `m[][]`:

d) Expressão booleana para: c é uma vogal minúscula (sem acentos)?

e) Ir buscar a letra do meio de uma string s (assuma que tem tamanho ímpar):

f) Desligar o antepenúltimo bit de um inteiro b (colocar a 0) :

Grupo 2 - Implementação de uma classe simples

2.1. Implemente uma **classe** `Matrix` que representa uma **matriz quadrada de caracteres**, com atributos públicos para representar a matriz em si, bem como o tamanho do lado do quadrado. Inclua um construtor que receba o tamanho k e cria uma matriz de $k \times k$ (inicialmente com todas as posições com '.').

2.2. Implemente um **método** `clone()` da classe `Matrix` que cria e devolve uma novo objecto *Matrix* que é uma cópia da matriz original (mesmo número de linhas e colunas e mesmo conteúdo). Note que depois de criada a cópia, mexer numa das matrizes não deve implicar mudanças na outra matriz.

2.3. Implemente um **método** `verify(c)` que devolve *true* se alguma das linhas ou colunas é constituída apenas pelo caracter c. Por exemplo, uma chamada a `m.verify('X')` para uma das seguintes matrizes m, deveria devolver *true*:

$$\begin{bmatrix} . & . & . \\ X & X & X \\ . & . & . \end{bmatrix} \quad \begin{bmatrix} O & . & X \\ O & X & X \\ O & . & X \end{bmatrix} \quad \begin{bmatrix} X & A \\ X & X \end{bmatrix}$$

Já se a matriz m fosse uma das seguintes, uma chamada `m.verify('X')` deveria devolver $false$:

$$\begin{bmatrix} X & . \\ . & X \end{bmatrix} \quad \begin{bmatrix} . & . \\ . & X \end{bmatrix} \quad [A]$$

Se necessitar, pode criar métodos auxiliares.

Grupo 3 - Tipos Abstratos de Dados (TADs)

3.1. Escreva um **interface** Java para representar um conjunto de números inteiros (sem repetições), contendo métodos para inserir e remover um inteiro, verificar se um inteiro está no conjunto e devolver o número de inteiros do conjunto (use nomes indicativos para os métodos).

3.2. Suponha que quer criar uma classe X que **concretiza uma implementação** do interface anterior. Como deveria ser a primeira linha que declara essa classe?

3.3. Suponha que tem uma implementação do interface anterior que usa como base um array ordenado onde são guardadas as palavras (uma por cada posição). Seja n o número de palavras do conjunto. Indique a **melhor complexidade temporal** (notação $\mathcal{O}$) que conseguiria obter para os seguintes métodos (justificando sucintamente):

a) Devolver número de inteiros no conjunto:

b) Verificar se um inteiro está no conjunto:

c) Inserir um inteiro no conjunto:

3.4. Explique como poderia obter uma **implementação mais eficiente** para verificar se a palavra está no conjunto, indicando a nova complexidade e outras potenciais vantagens e desvantagens em comparação com o array ordenado.

Grupo 4 - Complexidade Algorítmica

4.1. Para os pares de funções seguintes, escreva na caixa em branco **Θ**, $\mathcal{O}$ ou **Ω** para tornar a afirmação verdadeira. Note que se as funções em causa tiverem uma relação Θ, não deverá escrever $\mathcal{O}$ ou Ω.

$5n \in \square\ (n)$ $\quad 2n \in \square\ (n^2)$ $\quad 2^n \in \square\ (n^2)$ $\quad n \log n \in \square\ (n^2)$ $\quad \log_2 9 \in \square\ (42)$ $\quad n! \in \square\ (n^n)$

4.2. Descreva o pior caso em termos de tempo de execução para os seguintes pedaços de código, usando notação $\mathcal{O}$ para a variável n. Use o limite mais "apertado"possível (ex: indicar todos como sendo $\mathcal{O}(n!)$ resultará em 0 pontos).

a)

```
for (int i=0; i<n; i++)
    for (int j=0; j<n; j++)
        for (int k=0; k<n; k++)
            count++;
```

b)

```
for (int i=24; i<42; i++)
    for (int j=2; j<n-2; j++)
        for (int k=0; k<n/2; k++)
            for (int l=0; l<n; l++)
                count++;
```

c)

```
int f1(int n) {
    if (n==0) return 0;
    return 1 + f1(n/3);
}
```

d)

```
int f2(int n) {
    if (n==0) return 0;
    return 2 + f2(n/2) + f2(n/2);
}
```

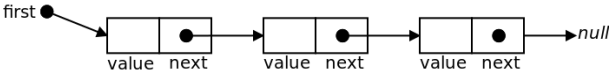
4.3. Justifique a sua resposta para a alínea (*d*) da pergunta anterior, **escrevendo a recorrência e desenhando a respectiva árvore de recorrência**, indicando porque dá origem à complexidade indicada.

4.4. Previsão de Tempo de Execução.
Complete a tabela seguinte usando a informação já preenchida em cada linha. Nos tempos, use uma previsão baseada no tempo já preenchido, tendo em conta a proporção entre n_1 e n_2 (ms = milisegundos).

Programa	Complexidade	Nome Comum	Tempo para $n_1 = 10$	Tempo para $n_2 = 20$
A		constante	60ms	
B		linear		30ms
C	$\Theta(n^2)$		15ms	
D		cúbica		40ms
E	$\Theta(2^n)$		10ms	

Grupo 5 - Listas, Pilhas e Filas

5.1. Considere uma implementação `SinglyLinkedList<T>` de uma lista ligada simples genérica tal como dada nas aulas com atributos `size` e first, e `Node<T>` representando um nó com atributos `value` e `next` (com *getters* e *setters*).



a) Implemente um **método** `contains(x)` devolve *true* se o elemento x está na lista e $false$ caso contrário. Não pode usar qualquer outro método da classe `SinglyLinkedList<T>`. Indique a **complexidade temporal** do método que descreveu.

b) Implemente um **método** `remove(i)` que remove o i-ésimo elemento da lista (elemento na posição i). As posições começam em 0. Se a posição i não existir, o método não deve fazer nada. Não pode usar qualquer outro método da classe `SinglyLinkedList<T>`. Indique a **complexidade temporal** do método que descreveu.

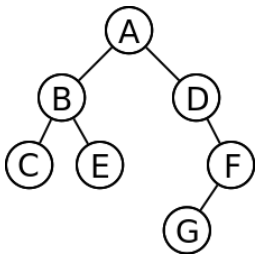
c) Implemente um **método** `noDuplicates()` que devolve uma nova lista que é igual à lista inicial sem elementos repetidos, mantendo apenas a primeira ocorrência de cada elemento (ex: $1 \to 1 \to 2 \to 3 \to 2 \to 1$ resultaria na lista $1 \to 2 \to 3$). Pode destruir a lista original e usar a API das listas (ex: removeFirst(), removeLast(), addFirst(), addLast()) ou os métodos anteriores (contains(x) e remove(i)). Indique a **complexidade temporal** do método que implementou.

5.2. Indique, justificando, uma **vantagem** e uma **desvantagem** de uma lista ligada em relação a um array.

5.3. Explique como poderia **inverter uma palavra** usando uma pilha, sendo que para interagir com a pilha apenas pode usar a sua API (*push*, *pop* e *size*). Indique a **complexidade temporal** do método que descreveu.

Grupo 6 - Árvores

6.1. Considerando a **árvore da figura** seguinte, responda às seguintes alíneas:



a) Quais os nós **folha**?

b) Qual a **altura** da árvore?

c) Os nós da árvore em **preorder:**

d) Os nós da árvore em **inorder:**

e) Os nós da árvore em **postorder:**

6.2. Quantos nós tem no mínimo uma árvore de altura 6? E no máximo?

6.3. Consider uma implementação `BTree<T>` de uma árvore binária genérica tal como dada nas aulas com um atributo `root` a apontar para a raíz que é um `BNode<T>` representando um nó com atributos `value`, `left` e `right`.

a) Implemente **um método** `leaves()` da classe `BTree<T>` que devolve o **número de folhas** da árvore (sem usar a API das árvores). Pode criar métodos auxiliares. Indique a **complexidade temporal** do método que implementou.

b) Implemente **um método** `bfs()` da classe `BTree<T>` que devolve um array contendo os elementos da árvore numa **pesquisa em largura**. Por exemplo, para a árvore da figura 6.1, deveria ser devolvido um array $[A, B, D, C, E, F, G]$. Pode usar as APIs de listas, pilhas e filas (não há problema se não souber de cor os nomes dos métodos, desde que use nomes indicativos da operação que pretende). Indique a **complexidade temporal** do método que implementou.

Grupo 7 - Árvores Binárias de Pesquisa

7.1. Suponha que quer inserir (por esta ordem) os seguintes 6 números numa árvore binária de pesquisa: $24, 42, 12, 5, 40, 16$.

Para as duas alíneas ao lado, **desenhe a árvore** (depois de completa a operação respetiva).

a) Como fica a **árvore original** depois de inseridos os 6 números?

b) Como fica a árvore (original) depois de **removermos o 24**?

7.2. Em que nó fica guardado o **elemento máximo** numa árvore binária de pesquisa? Justifique sucintamente.

7.3. Suponha que quer inserir os números de 1 a n numa árvore binária de pesquisa. **Qual a mínima altura possível da árvore?** Indique por qual ordem deve inserir os elementos para garantir essa altura mínima.

7.4. Suponha que tem um **TAD Dicionário** `BSTMap<K,V>` que mapeia chaves do tipo K em valores do tipo V implementado com uma **árvore binária de pesquisa** e que esta está **sempre equilibrada**. Suponha também que tem acesso a uma lista de n reviews de restaurantes. Cada review atribui uma nota de 0 a 100 a um restaurante, que é identificado pelo nome. Usando a API dos dicionários (*put*, *get*, *keys*, *size*) explique com algum detalhe (pode ser só por palavras) um algoritmo que descubra quais os restaurantes com pelo menos 42 reviews e nota média positiva (≥ 50). Indique, justificando, a a **complexidade temporal e espacial** (memória) do método que descreveu.

Grupo 8 - Filas de Prioridade e Heaps

8.1. Imagine que tem uma **minHeap** descrita pelo seguinte array

2	3	4	8	6	9

Para as 3 alíneas de baixo **desenhe a árvore** (a heap) correspondente (sempre já com o invariante reposto):

a) Qual a heap representada pelo **array original**?

b) Como fica a heap (original) depois de **adicionar um 1**?

c) Como fica a heap (original) depois de **remover o mínimo**?

8.2. Explique como poderia usar uma *minHeap* para **ordenar um conjunto de n números**. Indique que operações usava e qual seria a **complexidade temporal** do método que descreveu.

8.3. (pergunta de valorização) Explique como poderia implementar uma **min-max-heap**, uma estrutura de dados que suporte inserção de elementos em $\mathcal{O}(\log n)$, remoção quer do mínimo quer do máximo em $\mathcal{O}(\log n)$ e leitura (sem remoção) quer do mínimo quer do máximo em $\mathcal{O}(1)$.

(pode usar o resto desta página para continuar respostas que necessitem de mais espaço para além do que foi dado)

(pode usar esta página para continuar respostas que necessitem de mais espaço para além do que foi dado)

(pode usar esta página para continuar respostas que necessitem de mais espaço para além do que foi dado)