INSTITUTO SUPERIOR TÉCNICO

AED - Algoritmos e Estruturas de Dados 2010/2011 - 2º Semestre

2º Exame, 28 Junho 2011, 11:30h Duração: 3 horas

Prova escrita, individual e sem consulta

NOME:_____ NÚMERO:_____

PARTE I - Questões de Escolha Múltipla

Preencha as respostas na tabela (usando apenas letras maiúsculas). Se nenhuma opção servir, escreva **NENHUMA**. Se pretender alterar a sua resposta, risque e escreva ao lado a sua nova opção. Todas as questões de escolha múltipla valem 0.75 valores. As questões de escolha múltipla não respondidas são cotadas com 0 valores, mas por cada resposta errada são descontados 0.75/4 valores.

Questão	1	2	3	4	5	6	7	8
Resposta								

1. Considere uma função que realiza a procura de um item numa estrutura de dados. Qual das afirmações é verdadeira?

   A. A procura numa tabela ordenada tem maior complexidade do que numa árvore binária.
   B. A procura numa lista ordenada tem maior complexidade do que numa lista simples.
   C. A procura de um inteiro numa tabela simples exige menos comparações do que numa lista ordenada.
   D. A procura de uma palavra numa tabela ordenada tem menor complexidade do que a procura de um inteiro numa lista ordenada.

2. Considere um grafo ponderado com N vértices para o qual se pretende determinar a existência ou não de caminhos entre qualquer par de vértices. Para o fazer, assuma que se usam dois procedimentos: P1: (preparação) constrói-se um circuito eléctrico resistivo, em que cada vértice do grafo é um nó desse circuito; se dois vértices são adjacentes coloca-se uma resistência de valor R (valor proporcional ao peso da aresta) não nulo entre os respectivos nós do circuito; (execução) corre-se o algoritmo A1 para determinar da existência ou não de ligação, que consiste em aplicar uma tensão V a um par de nós (A,B) do circuito e medir a corrente I debitada pela bateria (ver figura). P2: (preparação) escreve-se um ficheiro com a descrição do grafo, listando os pares de vértices adjacentes; (execução) corre-se o algoritmo A2 que é implementado por um programa que lê o ficheiro, recebe como entrada o par de vértices (A,B), retornando 1 se A e B estiverem ligados e 0 se não estiverem.

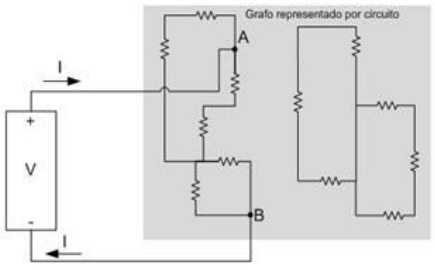


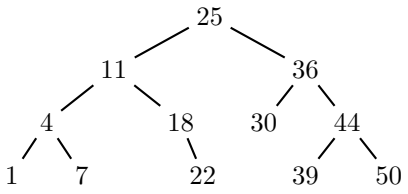
Figura 1: Circuito eléctrico.

Qual das seguintes afirmações é verdadeira:

   A. Se não existir um caminho entre os vérices a corrente é muito baixa.
   B. Quanto maior for a corrente mais curto é o caminho no grafo.
   C. A complexidade do algoritmo A1 é $\mathcal{O}(1)$.
   D. Quanto maior for a corrente mais longo é o caminho no grafo.

3. Considere a árvore binária ordenada e balanceada AVL representada na figura.

Assuma que após inserção de qualquer elemento, são efectuadas operações de balanceamento se necessário. Indique a opção que completa correctamente a seguinte afirmação.



Podem ser inseridos **sucessivamente e sem necessidade de operações de balanceamento** os elementos:

A. 38, 17 e 29.	B. 17, 53 e 29.	C. 15, 28 e 33.	D. 15, 28 e 26.

4. Considere a árvore binária ordenada e balanceada AVL do exercício anterior. Assumindo que nela é inserido o elemento 52 indique as operações que são necessárias efectuar para a rebalancear.

A.	Uma rotação dupla à direita no vértice 36.
B.	Uma rotação simples à esquerda no vértice 25.
C.	Uma rotação simples à esquerda no vértice 36.
D.	Uma rotação simples à direita no vértice 44.

5. Considere o seguinte acervo ("heap"): $\{35, 25, 33, 18, 23, 28, 30, 3, 10, 8, 20, 5, 15, 11\}$.

   Na representação em árvore deste acervo qual o nó pai do nó 28?

A. O nó 33.	B. O nó 23.	C. O nó 25.	D. O nó 35.

6. Suponha que numa tabela de dispersão ("hash table") são introduzidos 6 números. A função de dispersão usada é o resto da divisão inteira por 10.

   A tabela de dispersão obtida é $\{-, -, 902, -, 844, 514, 6, 54, 17, -\}$. O símbolo "-" significa que a posição da tabela correspondente está vazia.

   As colisões são resolvidas por procura linear.

   Quais os números, e por que ordem, foram introduzidos?

A.	$\{844, 6, 514, 54, 17, 902\}$	B.	$\{902, 844, 6, 514, 17, 54\}$
C.	$\{844, 514, 54, 6, 17, 902\}$	D.	$\{902, 514, 6, 844, 54, 17\}$

7. Considere a seguinte tabela (1ª linha) sobre a qual são listados alguns passos executados por um algoritmo de ordenação (restantes linhas). Qual é o algoritmo usado?

6	13	10	14	9	3	7	12	8	5	15	11
3	6	13	10	14	9	5	7	12	8	11	15
3	5	6	13	10	14	9	7	8	12	11	15
3	5	6	7	13	10	14	9	8	11	12	15
3	5	6	7	8	13	10	14	9	11	12	15
3	5	6	7	8	9	13	10	14	11	12	15
3	5	6	7	8	9	10	13	11	14	12	15

A.	Inserção
B.	*Shellsort (h=4, 2, 1)*
C.	*Bubblesort*
D.	*Quicksort*

8. Utilizando a notação assimptótica estudada, determine a ordem da solução da seguinte recorrência (escolha o menor majorante):

$$C_N = 3C_{N/2} + N$$

A. $\mathcal{O}(N3^{\lg_2 N})$	B. $\mathcal{O}(N \lg_2 N)$	C. $\mathcal{O}(N3^{N/2})$	D. $\mathcal{O}(3 \lg_2 N)$

PARTE II - Questões de Desenvolvimento

Responda a cada uma das questões de desenvolvimento em **folhas de exame separadas** e devidamente identificadas com nome e número.

[5.0] 9. O Triângulo de Sierpinsky é uma forma geométrica auto-semelhante. A figura junta apresenta várias versões correspondentes a diferentes números de iterações.

Figura 2: Triângulo de Sierpinsky.

A função apresentada permite desenhar esse triângulo. O argumento `pt` contém as coordenadas do canto inferior esquerdo do triângulo. O argumento `N` contém o comprimento da base do triângulo. O tipo de dados `Point` é uma estrutura com duas variáveis do tipo float. A função `DesenhaTriangulo` desenha no ecrã um triângulo preto. Os seus argumentos têm o mesmo significado dos argumentos da função `Sierpinski`.

```
void Sierpinski(Point pt, float N) {
 Point pt1, pt2, pt3;

 if (N == 1) {
  DesenhaTriangulo(pt, N);
  return;
 }

 pt1.x = pt.x;
 pt1.y = pt.y;
 pt2.x = pt.x + N/2;
 pt2.y = pt.y;
 pt3.x = pt.x + N/4;
 pt3.y = pt.y + sqrt(3)/4*N;

 Sierpinski (pt1, N/2);
 Sierpinski (pt2, N/2);
 Sierpinski (pt3, N/2);
}
```

Resolva as seguintes alíneas (não se esqueça que $x^{log_a(y)} = y^{log_a(x)}$):

[1.5] a) Escreva a expressão da recorrência que traduz a complexidade temporal desta função.

[1.0] b) Resolva a expressão da recorrência da alínea anterior. Utilize o Master Theorem, se lhe parecer adequado.
Se e somente não tenha resolvido a alínea a) use nesta alínea a recorrência $C_N = 2C_{N/3} + 2$.

[1.5] c) Escreva a expressão da recorrência que traduz a complexidade espacial desta função.

[1.0] d) Resolva a expressão da recorrência da alínea anterior usando a propriedade telescópica.
Se e somente não tenha resolvido a alínea b) use neste alínea a recorrência $C_N = C_{N-1} + \lg N$.

[5.0] 10. O triângulo de Pascal, ilustrado na figura 3, à esquerda, é formado começando num apex 1. Cada número abaixo no triângulo corresponde à soma dos dois números diagonalmente acima, à esquerda e à direita, com as posições fora do triângulo a contarem como zero. Note que o triângulo pode ser representado na forma de matriz, como ilustrado na figura 3, à direita.

O triângulo de Pascal possui diversas propriedades invulgares, das quais se enumeram as seguintes, ilustradas na figura 4:

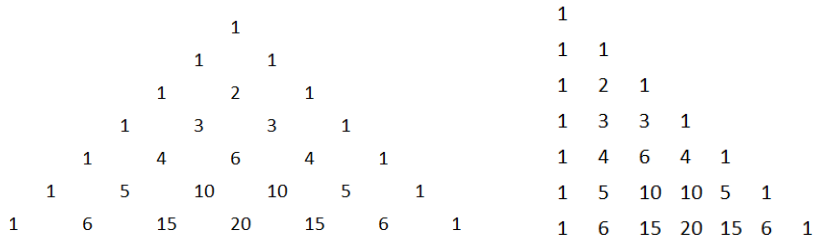


Figura 3: Triângulo de Pascal, à esquerda, e respectiva representação matricial.

(i) A soma dos números nas diagonais do triângulo produzem a série de Fibonacci (indicada na linha superior da figura).

(ii) A soma de cada linha corresponde a uma potência de 2 (i.e., 1, 2, 4, 8, 16 - coluna à direita na figura).

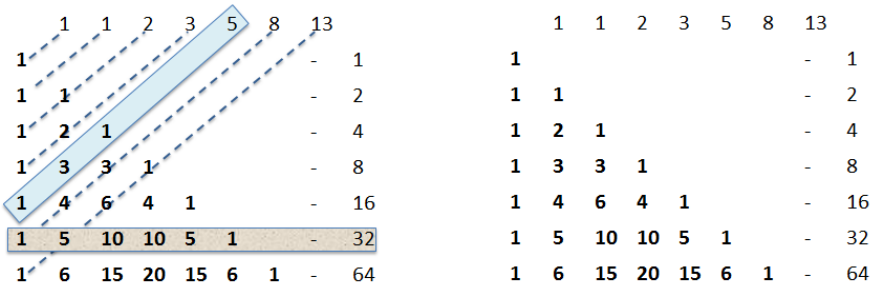


Figura 4: Esquerda: Triângulo de Pascal, série de Fibonacci (linha superior) e potências de 2. (coluna à direita). Direita: Ouput do programa para $N = 7$.

Pretende-se fazer um programa que ilustre estas propriedades, através de uma função que, recebendo um inteiro N, determine o triângulo de Pascal de altura N, bem como a série de Fibonacci e as potências de 2 correspondentes, produzindo como resultado o *output* no formato na figura 4 à direita. Para tal:

[1.0] a) Triângulo de Pascal: Escreva o código em C de uma função que receba como argumento um inteiro N e que determina o triângulo de Pascal de altura N, de acordo com a assinatura:

```
int ** trianguloPascal(int N);
```

[1.0] b) Série de Fibonacci: Escreva o código em C de uma função que receba como argumento o ponteiro para uma matriz de inteiros *mat* (representando o Triângulo de Pascal) e um inteiro N e que determina a série de Fibonacci associada (de comprimento N), de acordo com a assinatura:

```
int * Fibonacci(int ** mat, int N);
```

[1.0] c) Potências de 2: Escreva o código em C de uma função que receba como argumento o ponteiro para uma matriz de inteiros *mat* (representando o Triângulo de Pascal) e um inteiro N, e que determina a série de potências de 2 associada (de comprimento N), de acordo com a assinatura:

```
int * potencias2(int ** mat, int N);
```

[1.0] d) Escreva o código em C de uma função que receba como argumento um inteiro N, que chame as funções anteriores e imprima no ecrã o output indicado à direita da figura 4. Esta função tem a assinatura:

```
void pascal_props(int N);
```

[0.5] e) Determine justificadamente a **complexidade espacial** da função que escreveu em a), como função de N. Explicite a expressão de recorrência que caracteriza a referida complexidade.

[0.5] f) Determine justificadamente a expressão que carateriza a **complexidade temporal** da função `pascal_props`.

[4.0] 11. Considere um grafo representado por listas de adjacências, como se ilustra abaixo.

A →	B, 6	D, 3	E, 7	H, 5		
B →	E, 8	C, 4	F, 7	A, 6		
C →	H, 9	F, 5	B, 4	D, 8	E, 4	
D →	A, 3	C, 8	E, 9	G, 8	H, 2	
E →	B, 8	C, 4	A, 7	D, 9	H, 8	F, 5
F →	C, 5	B, 7	H, 1	G, 10	E, 5	
G →	D, 8	F, 10	H, 6			
H →	F, 1	C, 9	E, 8	A, 5	D, 2	G, 6

Figura 5: Grafo ponderado não direccionado.

[3.0] a) Determine a árvore de mínima de suporte (MST) usando o algoritmo de Prim e tomando o vértice **A** como fonte. Apresente os seus cálculos de forma clara, detalhada e completa para cada iteração do algoritmo. Por exemplo, mas sem se restringir a estes aspectos, identifique a franja da procura e pesos, assim como deverá indicar por que ordem entra cada vértice na árvore e o estado da franja em cada momento.

[1.0] b) Considere a seguinte afirmação: *Para qualquer grafo ponderado, direccionado ou não, a aresta de maior peso nunca faz parte da árvore Mínima de Suporte.*

Indique qual o valor lógico da afirmação e apresente uma demonstração clara e rigorosa que fundamente a sua opção.