INSTITUTO SUPERIOR TÉCNICO

AED - Algoritmos e Estruturas de Dados 2011/2012 - 1º Semestre

1º Exame, 4 Fevereiro 2012, 08:00h 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. Admita que se pretende listar algumas funções de N de forma crescente quanto ao seu crescimento assimptótico. Indique qual das seguintes sequências é **falsa**:

   A. $\log N,\ N\log N,\ N^5,\ 1.1^N$ B. $2\log N,\ N(\log N)^2,\ N^2\log N,\ N^5$

   C. $2\log N,\ N\log N,\ 1.1^N,\ 2^5$ D. $\log N,\ N\log N,\ N(\log N)^2,\ 1.1^N$

2. Considere o código apresentado à direita.

   Considerando que `n = n2-n1` determina a dimensão do problema, qual dos pares de equações abaixo representa as recorrências associadas com o tempo de computação, (T_n), e a memória usada, (M_n), pela função `function`, respectivamente? Assuma que a função é inicialmente chamada com `n1` $= 0$ e `n2` > 0.

```
int * function(int * a, int n1, int n2)
{
 int b, *c, n;
 n = n2-n1;
 if (n == 1) return a;
 c = (int *) malloc(sqrt(n)*sizeof(int));
 {outras instruções básicas}
 a = function(a, n1, n1+n/2);
 {mais instruções básicas}
 c = function(a, n1+n/2, n2);
 return c;
}
```

   A. $T_n = 2T_{n/2} + 1;\ M_n = 2T_{n/2} + \sqrt{n}$ B. $T_n = 2T_{n/2} + \sqrt{n};\ M_n = T_{n/2} + \sqrt{n}$

   C. $T_n = 2T_{n/2} + 1;\ M_n = T_{n/2} + \sqrt{n}$ D. $T_n = 2T_{n/2} + \sqrt{n};\ M_n = 2T_{n/2} + \sqrt{n}$

3. Diga qual das seguintes afirmações é **verdadeira** relativamente aos algoritmos de ordenação. Assuma em cada caso a melhor implementação de cada algoritmo (adaptativa, optimizada, etc):

   A. Mesmo no pior caso, o Algoritmo *QuickSort* tem sempre uma complexidade de $\mathcal{O}(N\log N)$

   B. O Algoritmo de Selecção é especialmente interessante porque faz sempre $\mathcal{O}(N)$ comparações

   C. A versão adaptativa do algoritmo de Inserção nunca tem melhor desempenho que a versão não adaptativa do mesmo algoritmo

   D. O desempenho do algoritmo *ShellSort* depende da sequência de passos utilizada.

4. Considere a seguinte tabela

05	08	12	10	09	08	04	05	06	08	09	11	07	08

e suponha que aplica o algoritmo de ordenação *quicksort*. Qual das tabelas abaixo se obtém após a primeira chamada à função que executa a partição? Assuma que o pivot é o último elemento da tabela original.

A.

05	08	07	06	05	08	04	08	10	08	09	11	12	09

B.

05	07	08	06	05	04	08	09	10	12	09	11	08	08

C.

05	08	07	08	06	08	04	05	08	10	09	11	12	09

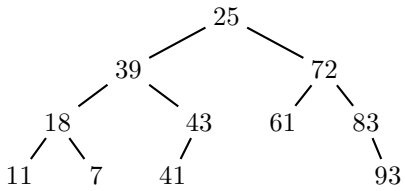
D.

05	07	06	05	04	08	09	10	12	08	09	11	08	08

.

5. Considere a árvore binária representada na figura.

Qual das sequências se obtém se a árvore for percorrida em modo in-fixado ?



A.

07	18	11	39	41	43	25	61	72	83	93

B.

11	18	07	39	43	41	25	61	72	83	93

C.

11	18	07	39	41	43	25	61	72	93	83

D.

11	18	07	39	41	43	25	61	72	83	93

6. Considerando a condição de acervo por valor, ou seja, maior valor corresponde a maior prioridade, qual das seguintes tabelas **não** representa um acervo?

A.

93	43	83	41	25	61	72	39	18	07	11

B.

93	83	72	41	25	43	61	18	39	07	11

C.

93	83	72	39	25	43	61	18	41	07	11

D.

93	83	72	25	41	43	61	07	11	18	39

7. Assuma que após a aplicação de algumas iterações do algoritmo de Kruskal, a tabela representando as sub-árvores que constituem a MST em construção é a que se apresenta abaixo. Suponha que os índice são contados a partir de zero e que **das arestas ainda não consideradas** para inclusão na MST, as quatro arestas de valor mais baixo são: (7, 8) com peso 13; (10, 5) com peso 14; (5, 13) com peso 16; e (4, 14) com peso 18. Quais destas quatro arestas serão incluídas na MST?

07	06	07	03	05	06	06	03	03	09	01	01	12	09	13	12	12	16

A. Só a terceira. B. A primeira e a quarta. C. Só a quarta. D. A segunda e a terceira.

8. Considere uma tabela de dispersão, ("hash table"), são introduzidos 6 números pela seguinte ordem: {360, 100, 37, 606, 99, 28}. As colisões resolvem-se por **procura linear**. As primeiras posições da tabela são: {360, 37, 606, 99, 100, 28, ...}. Não deverá assumir nada sobre o tamanho da tabela. Qual a função de dispersão usada?

A. Resto da divisão inteira por 8. B. Resto da divisão inteira por 9 da soma dos dígitos.
C. Resto da divisão inteira por 6. D. Resto da divisão inteira por 5 da soma dos dígitos.

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.

[4.0] 9. Considere um grafo ponderado não direccionado cuja matriz de adjacências se apresenta abaixo.

$$\begin{pmatrix} 0 & \infty & 21 & 13 & 15 & 17 & \infty & \infty & \infty & \infty \\ \infty & 0 & 53 & 22 & 8 & 6 & 18 & \infty & \infty & \infty \\ 21 & 53 & 0 & \infty & \infty & \infty & \infty & 18 & \infty & 4 \\ 13 & 22 & \infty & 0 & 18 & \infty & \infty & 2 & \infty & 14 \\ 15 & 8 & \infty & 18 & 0 & 7 & \infty & 9 & 15 & \infty \\ 17 & 6 & \infty & \infty & 7 & 0 & \infty & \infty & 12 & 14 \\ \infty & 18 & \infty & \infty & \infty & \infty & 0 & \infty & 17 & 21 \\ \infty & \infty & 18 & 2 & 9 & \infty & \infty & 0 & \infty & 17 \\ \infty & \infty & \infty & \infty & 15 & 12 & 17 & \infty & 0 & 6 \\ \infty & \infty & 4 & 14 & \infty & 14 & 21 & 17 & 6 & 0 \end{pmatrix}$$

[2.5] a) Assumindo que a numeração dos vértices começa em zero, tome o vértice de índice 0 como ponto de partida para determinar a Árvore de Mínima de Suporte ("MST"), usando o algoritmo de Prim. Indique justificadamente todos os passos que executar e descreva em detalhe os cálculos realizados.

[0.75] b) A partir dos cálculos da alínea anterior, trace a "MST" que obteve.

[0.75] c) Qual o custo da "MST" que obteve?

[3.0] 10. Suponha que se insere um elemento x numa árvore de procura binária, ordenada e balanceada AVL. Assuma que o elemento x não existia anteriormente na árvore. Imediatamente após a inserção, removemos x da árvore.

[2.0] a) A árvore resultante, depois da remoção é sempre idêntica à árvore inicial antes da inserção? Justifique a sua resposta indicando se as árvores são sempre idênticas, nunca o são, ou são-no apenas em certas condições (nesse caso discuta quais).

[1.0] b) Dê um exemplo de uma árvore de procura binária, ordenada e balanceada AVL cujos elementos são números inteiros positivos, que contenha pelo menos 10 elementos, e introduza na mesma um elemento maior do que todos os lá contidos, outro menor que todos os elementos já na árvore e outro que seja próximo da média dos elementos já na árvore mas distinto de todos eles. Em todos os casos, a árvore resultante deve continuar ordenada e balanceada AVL.

[3.0] 11. Suponha um acervo, H e dois números, v_1 e v_2, distintos entre si e distintos de todos os elementos já contidos no acervo. Seja H_{12} o acervo resultante de introduzir primeiro v_1 e depois v_2 no acervo e H_{21} o acerto resultante de introduzir primeiro v_2 e depois v_1 no acervo.

[1.5] a) Dê um exemplo de um acervo H e de dois números v_1 e v_2 tais que os acervos H_{12} e H_{21} sejam iguais. Indique o acervo original e o acervo resultante após a introdução de cada um dos números. O acervo original, H, por si construído deverá ter, pelo menos, 10 elementos.

[1.5] b) Dê um exemplo de um acervo H e de dois números v_1 e v_2 tais que os acervos H_{12} e H_{21} sejam distintos. Indique o acervo original e o acervo resultante após a introdução de cada um dos números. O acervo original, H, por si construído deverá ter, pelo menos, 10 elementos.

[4.0] 12. Na implementação da ADT para grafos não ponderados com tabela de listas de adjacências (normalmente designada mais simplesmente apenas como listas de adjacências) suponha que pretende escrever a função de remoção de arestas, com a seguinte assinatura.

```
void GRAPHRemoveE(Graph *G, Edge *e);
```

De acordo com os acetatos da disciplina `Graph` e `Edge` são definidos na interface como

```
tpedef struct graph Graph;
typedef struct edge Edge;
```

Na implementação por listas de adjacências tem-se

```
typedef struct node link;
struct node {int v; link *next;};
```

```
struct graph {int V; int E; link **adj;};
struct edge {int v; int w;};
```

[1.5] a) Escreva em C uma implementação daquela função, assumindo que o grafo é não direccionado. Note que no argumento `Edge *e` se identifica a aresta a remover e que, como o grafo é não direccionado, o vértice `e->v` terá que ser retirado da lista do vértice `e->w`, assim como o vértice `e->w` terá que ser retirado da lista do vértice `e->v`. Se a aresta não existir, não poderá ser retirada, pelo que ambas as listas de adjacências permanecerão inalteradas.

[1.5] b) Escreva em C uma implementação daquela função, assumindo agora que o grafo é direccionado. Neste caso apenas o vértice `e->w` deverá ser retirado da lista do vértice `e->v`. Naturalmente, se a aresta não existir, não poderá ser retirada.

[1.0] c) Indique de forma justificada e precisa qual a complexidade da função escrita na alínea a). Pretende-se que explicite a análise de pior caso e de caso médio.

Mesmo que não tenha implementado aquela função, deverá ser capaz de indicar a sua complexidade, como função dos parâmetros que entender adequados, desde que justificados.