INSTITUTO SUPERIOR TÉCNICO

1º Exame, 29 Junho 2009, 13:00h 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 0.25 valores. As questões de escolha múltipla não respondidas são cotadas com 0 valores, mas por cada resposta errada são descontados 0.25/4 valores.

Questão	1	2	3	4	5	6	7	8
Resposta								

1. 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?

2	11	6	10	7	4	1	5	9	8	3	12
2	11	6	10	7	4	1	5	9	8	3	12
2	4	6	10	7	8	1	5	9	11	3	12
2	4	1	10	7	8	3	5	9	11	6	12
2	4	1	5	7	8	3	10	9	11	6	12
1	4	2	5	3	8	6	10	7	11	9	12
1	4	2	5	3	8	6	10	7	11	9	12

A. Inserção

B. *Shellsort (h=4, 2, 1)*

C. *Shellsort (h=4, 3, 1)*

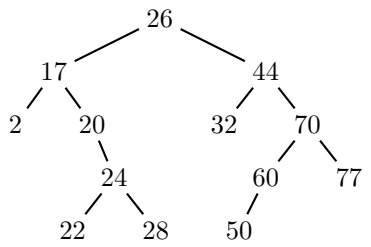
D. *Quicksort*

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

$$C_N = C_{N-1} + \lg(N^2)$$

A. $\mathcal{O}(\lg N)$ B. $\mathcal{O}(N)$ C. $\mathcal{O}(N \lg N)$ D. $\mathcal{O}(N^2)$

3. Considere a árvore **binária** representada na figura à direita.



Qual das afirmações é **verdadeira** relativamente à árvore?

A. É ordenada e balanceada AVL.
B. É ordenada e não balanceada AVL.
C. Não é ordenada mas é balanceada AVL.
D. Não é ordenada nem balanceada AVL.

4. Relativamente à árvore da questão anterior, suponha que se lhe aplicou varrimento pós-fixado. Qual das seguintes sequências se obteria se o processamento de um nó implicasse imprimir o seu valor?

A. [2 22 24 28 20 17 32 50 60 77 70 44 26]
B. [2 22 28 24 20 17 32 77 50 60 70 44 26]
C. [2 22 28 24 20 17 32 50 60 77 70 44 26]
D. [2 22 28 24 20 17 32 44 50 60 77 70 26]

5. Considere o seguinte acervo ("heap"), no qual a prioridade é mais elevada quanto menor for o valor.

3	7	10	13	14	11	23	17	30	41	20	70	24	35

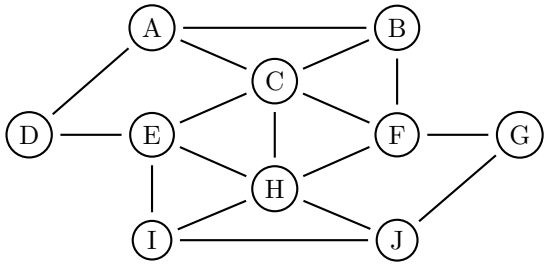
No total **quantas trocas** entre elementos da tabela são efectuadas nas rotinas de `FixUp()` e `FixDown()` durante a execução da seguinte sequência de operações (Nota: após cada operação é sempre necessário repôr a condição de acervo; conte **apenas** as trocas feitas no contexto das rotinas indicadas): (i) **Inserir** no acervo o valor 8; (ii) **Baixar** a prioridade do valor 7 para 18; (iii) **Retirar** o elemento mais prioritário.

A. 5	B. 6	C. 7	D. 8

6. Suponha que numa tabela de dispersão ("hash table") são introduzidas as seguintes palavras: **{amplexo, capacidade, computador, universidade, tremoço, capacidade, honestidade, complexo, cidade}**. Assuma que a função de dispersão (para indexação na tabela) usa os últimos três caracteres de cada palavra, que palavras repetidas são descartas quando tal é verificado e que colisões são resolvidas por lista, com inserção no **final**. Qual das afirmações é **verdadeira**?

A.	Ocorrem 5 colisões e 8 comparações.	B.	Ocorrem 7 colisões e 8 comparações.
C.	Ocorrem 5 colisões e 7 comparações.	D.	Ocorrem 7 colisões e 7 comparações.

7. Considere o grafo indicado em baixo à esquerda e assuma que o mesmo não é direccionado mas é ponderado, como se indica do lado direito do grafo. Assumindo que aplica o algoritmo de Prim tomando o vértice **D** como ponto de partida, indique qual das afirmações é **falsa**.



D↔A, B↔F, E↔H	2
B↔A, F↔C	3
A↔C, E↔D, G↔J	4
C↔B, C↔E	5
E↔I	6
H↔C, I↔J	7
I↔H, H↔F	8
J↔H, F↔G	9

A.	O custo da MST que se obtém é 33 e tem 9 arestas.
B.	A aresta que une os vértices **D** e **E** é a quinta a entrar na MST.
C.	A aresta que une os vértices **C** e **F** é a quarta a entrar na MST.
D.	A aresta que une os vértices **H** e **I** é a oitava a entrar na MST.

8. Indique qual das afirmações seguintes é **falsa**:

A.	$\sqrt{\lg N} \in \mathcal{O}(\lg \sqrt{N})$	B.	$\lg^2 \sqrt{N} \in \mathcal{O}(\lg N)$
C.	$\sum_{i=1}^{N} 1/i \in \mathcal{O}(\lg N)$	D.	$\lg^2 N \notin \mathcal{O}(\lg \lg N)$

PARTE II - Questões de Desenvolvimento

Responda às questões de desenvolvimento em folhas de exame devidamente identificadas com nome e número.

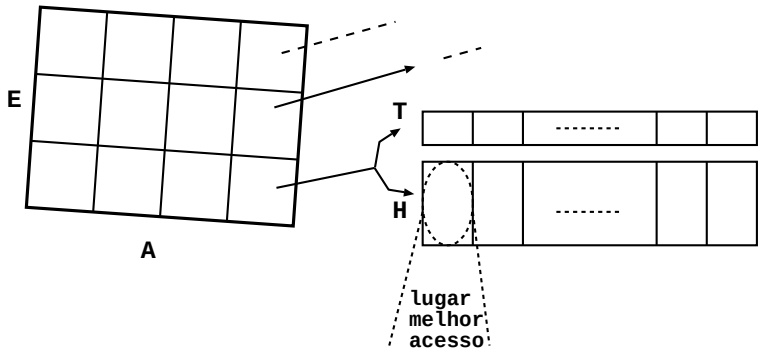
[3.0] 9. Considere que o seu programa de Gestão de Garagens calcula todas as SPT's de entradas para lugares de estacionamento e de acessos para lugares de estacionamento. Considere ainda que para cada lugar de estacionamento e para cada par entrada/acesso calcula o custo total de seguir desde essa entrada de carro até ao lugar em questão, seguindo depois a pé para esse acesso.

Suponha que calculou e armazenou estes custos totais numa tabela, para cada par entrada/acesso. Em seguida transformou cada uma destas tabelas num acervo, sendo que a condição de acervo indica que um vértice pai possui custo mais baixo que qualquer dos seus dois filhos. Como precisa saber em que posição do acervo se encontra cada lugar de estacionamento, necessitou criar uma tabela auxiliar por acervo onde indica a posição de cada lugar de estacionamento no acervo respectivo.

Imagine que numerou os N lugares de estacionamento de 0 a $N-1$, que numerou todas as E entradas de 0 a $E-1$ e que numerou todos os A acessos de 0 a $A-1$. Assim, para a entrada i e acesso j, existem duas tabelas T_{ij} e H_{ij} tais que para o lugar k se tem $H_{ij}[T_{ij}[k]] \rightarrow lugar = k$, com $k = 0, 1, 2, \ldots, N-1$ (ou seja para um lugar k, $T_{ij}[k]$ indica em que posição do acervo H_{ij} o lugar está).

Cada tabela T_{ij} contém apenas inteiros e cada tabela H_{ij} contém os seguintes objectos:

```
typedef struct {
 int lugar;
 double melhor;
 double actual;
} Lugar;
```



O campo `lugar` indica o índice do lugar de estacionamento, o campo `melhor` contém o valor de wt para o par entrada/acesso e o campo `actual` contém o mesmo valor que `melhor` se o lugar estiver disponível ou ∞ se o lugar estiver ocupado.

O acervo é construído usando como chave o campo `actual`. Sempre que um lugar disponível é atribuido a um carro, existe uma função que coloca ∞ em `actual` para todos os acervos existentes e os actualiza, repondo a condição de acervo. Sempre que um carro sai da garagem, existe outra função que coloca o valor de `melhor` em `actual` para o lugar que estava ocupado em todos os acervos existentes e os actualiza, repondo a condição de acervo.

[1.0] a) Para ser simples encontrar os vários acervos existentes, suponha que existe uma tabela bi-dimensional, onde se guardam os endereços do par H/T associado com cada par entrada/acesso. A figura acima, à direita ilustra esta situação. Escreva a função que aloca essa tabela e preenche-a com ponteiros para `NULL`. A função possui a seguinte assinatura:

`HT *** aloca_tabela(int entradas, int acessos)`,

onde o tipo `HT` é assim definido `typedef struct {int *T; Lugar * H;}HT;`

NOTA: Pretende-se apenas alocar a tabela de apontadores para `HT` e não as tabelas T e H. Assuma que essas serão tratadas posteriormente à medida que forem sendo calculadas.

[0.5] b) Como função de N, E e A, indique qual a complexidade da operação abstracta

`encontrar_melhor_lugar(HT *** Acervos, int entrada, Type acesso)`,

em que `Acervos` é a tabela onde estão os endereços de todas as tabelas, `entrada` indica qual a entrada onde que se encontra o carro e `acesso` indica o tipo de acesso pretendido.

[0.75] c) Como função de N, E e A, indique qual a complexidade da operação abstracta

`atribuir_lugar(HT *** Acervos, int k)`,

em que se pretende marcar o lugar k como ocupado e actualizar os acervos.

[0.75] d) Como função de N, E e A, indique qual a complexidade da operação abstracta

`libertar_lugar(HT *** Acervos, int k)`,

em que se pretende marcar o lugar k como livre e actualizar os acervos.

[2.5] 10. Abaixo encontram-se duas implementações alternativas para calcular x^n. Poderá experimentar calcular 2^9 usando cada uma das duas funções para verificar que são ambas correctas.

```
double power1(double x, int n)
{
  double result = 1;

  while (n > 0) {
    result = result*x;
    n = n-1;
  }
  return result;
}
```

```
double power2(double x, int n)
{
  double result = 1;

  while (n > 0) {
    if (n%2 == 1) { /* Se n é ímpar, multiplicar por x
                                          e decrescer n */
      result = result*x;
      n = n-1;
    }
    x = x*x;  /* Se n é par, então basta levantar
                 o quadrado de x a metade de n */
    n = n/2;
  }
  return result;
}
```

[0.5] a) Indique justificadamente, através da formulação de uma recorrência e sua resolução, qual a complexidade de `power1` como função de n.

[1.0] b) Altere a função `power2` para formato recursivo. A técnica de desenho usada pode identificar-se como *divide-and-conquer* ou programação dinâmica? Justifique.

[0.5] c) Indique justificadamente, usando notação assimptótica, qual a complexidade temporal de `power2`.

[0.5] d) Qual das duas funções escolheria para calcular x^n? Porquê?

[2.0] 11. No contexto das árvores binárias, imagine que se pretende remover da árvore todas as folhas cujo valor seja superior a um determinado valor. Suponha que a árvore binária está assim implementada

```
typedef struct _BTree{
 Item node;
 _BTree * esquerda;
 _BTree * direita;
} BTree;
```

[1.5] a) Escreva a função `int prune(BTree *a, double bound)`, em que `bound` é o valor de corte e assuma que existe uma função que para cada `Item` devolve o valor da chave de interesse, com a seguinte assinatura `double key_value(Item i)`. Note que se pretende apenas podar vértices que sejam folhas. Como função do número de vértices, qual a complexidade da sua função?
Sugestão: Para uma implementação recursiva, se achar útil, a função pode retornar 1 quando um vértice é folha e tem que ser removido, retornando 0 em todos os restantes casos.

[0.5] b) Ambas as funções acima admitem uma implementação recursiva ou iterativa. Independemente da opção que tomou, se optarmos por uma implementação recursiva, para cada problema qual das duas técnicas de decomposição de problemas que estudou é aplicável e porquê?

[2.5] 12. Considere de novo o grafo da Questão 7.

[1.75] a) Determine a SPT 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. Deverá ainda indicar qual o valor final do vector st e do vector wt.

[0.5] b) Indique quais os vértices que fazem parte do caminho mais curto entre o vértice **A** e o vértice **J**, bem como qual o seu valor de custo.

[0.25] c) Diga se a SPT que calculou lhe permite ou não dizer qual o caminho mais curto entre **C** e **F**. Porquê?