DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

AED - 2014-2015 - 1° Semestre
Algoritmos e Estruturas de Dados

1º Exame, 6 Janeiro 2015, 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								

[0.75] 1. Considere o código à esquerda, utilizado no problema da conectividade (correspondente a uma das soluções que foi proposta e analisada nas aulas). Se for introduzida a sequência de pares indicada à direita, qual a tabela que se obtém no final?

```
void main (int argc, char *argv[]) {
 int i, p, q, a, N=10, id[10];

 for (i = 0; i < N ; i ++) id[i] = i ;

 while (scanf("%d %d", &p, &q) == 2) {
   if (id[p] == id[q]) continue;
   a = id[p];
   for (i = 0; i < N; i ++)
     if (id[i] == a) id[i] = id[q];
   printf ("%d %d\n", p, q);
 }
}
```

1-6, 7-3,4-2, 5-3, 0-7, 5-1

A. 6 6 2 6 2 6 6 6 8 9

B. 6 6 2 3 2 6 6 6 8 9

C. 7 6 2 3 2 1 6 3 8 9

D. 7 6 2 3 2 3 6 3 8 9

[0.75] 2. Uma das virtudes da construção de funções recursivas assenta na possibilidade de se resolver um problema como composição da resolução de problemas de menor dimensão. Neste enquadramento, estudou duas técnicas principais de projecto de funções recursivas.

Sobre essas indique qual das seguintes afirmações é **verdadeira**.

A. A programação dinâmica ascendente é indicada para problemas em que a decomposição produz problemas independentes, enquanto que a programação dinâmica descendente se aplica a casos em que os sub-problemas possuem interdependências.

B. A programação dinâmica descendente é indicada para problemas em que a decomposição produz problemas independentes, enquanto que divide-and-conquer se aplica a casos em que os sub-problemas possuem interdependências.

C. A programação dinâmica descendente é indicada para problemas em que a decomposição produz problemas independentes, enquanto que a programação dinâmica ascendente se aplica a casos em que os sub-problemas possuem interdependências.

D. A programação dinâmica ascendente é indicada para problemas em que a decomposição produz problemas com interdependências, enquanto que divide-and-conquer se aplica a casos em que os sub-problemas são independentes.

[0.75] 3. Considere um procedimento efectuado sobre N dados guardados numa **tabela** em que é necessário:
(1) examinar todos os elementos **uma vez**, com custo $\mathcal{O}(1)$ para cada, e como resultado dessa operação dividir os dados em dois subconjuntos, A e B, o que pode implicar **uma** troca para cada elemento examinado;
(2) repetir **recursivamente** o procedimento para os dados no subconjunto A e depois para os dados no subconjunto B.

Indique qual das seguintes respostas é **verdadeira** relativamente à complexidade computacional deste procedimento.

A. No melhor caso é $\mathcal{O}(N)$ porque no passo (1) cada elemento só é examinado uma vez.
B. É sempre $\mathcal{O}(N^2)$ porque para cada elemento temos de examinar todos os restantes.
C. No caso médio é $\mathcal{O}(N \log N)$ porque no passo (2) os dois subconjuntos são processados separadamente.
D. É sempre $\mathcal{O}(N)$ se não precisar de memória adicional.

[0.75] 4. Suponha que tem uma pilha inicialmente vazia, na qual é efectuada a sequência de operações de `push()` e `pop()` indicadas à esquerda. Assuma que cada `pop()` imprime o resultado dessa operação. Indique das sequências à direita qual é a sequência que é impressa.

```
push (0), push (1), pop(), push (2), push (3), pop(),
pop(), push (4), push (5), pop(), pop(), pop(), push (6),
pop(), push (7), push (8), pop()
```

A. 0 1 2 3 4 5 6 7.
B. 1 3 5 7 2 4 6 8.
C. 0 2 3 4 5 1 7 8.
D. 1 3 2 5 4 0 6 8.

[0.75] 5. Considere os algoritmos de ordenação básicos estudados (seleção, inserção e *bubble*) e assuma em cada caso a melhor implementação de cada algoritmo (adaptativa, optimizada, etc.) Relativamente a estes algoritmos, indique qual das seguintes afirmações é **falsa**.

A. No pior caso, todos os algoritmos têm desempenho assintótico semelhante.
B. A complexidade computacional da ordenação por selecção depende da ordenação inicial dos dados.
C. Quando os dados estão quase ordenados, o *bubble sort* e a ordenação por inserção são quase lineares.
D. Todos os algoritmos são estáveis.

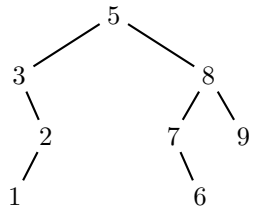
[0.75] 6. Considere a seguinte tabela (1ª linha) sobre a qual são listados alguns passos executados por um algoritmo de ordenação elementar. Indica-se o resultado após a conclusão do *loop* interior. Qual é o algoritmo usado?

10	18	19	16	11	14	15	14	12	13
10	18	19	16	11	14	15	14	12	13
10	11	19	16	18	14	15	14	12	13
10	11	12	16	18	14	15	14	19	13
10	11	12	13	18	14	15	14	19	16
10	11	12	13	14	18	15	14	19	16

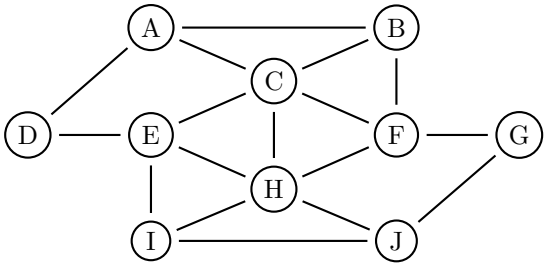
A. Seleção
B. Inserção
C. *Shellsort (h=4, 2, 1)*
D. *Bubble*

[0.75] 7. Considere a árvore binária da figura à direita. Indique qual das seguintes afirmações é **verdadeira**.

A. A árvore é balanceada AVL e o varrimento pré-fixado produz: 5-3-2-1-8-7-6-9.
B. A árvore é ordenada e o varrimento in-fixado produz: 3-1-2-5-7-6-8-9.
C. A árvore não é balanceada AVL e o varrimento em profundidade produz: 5-3-8-2-7-9-1-6.
D. A árvore não é ordenada e o varrimento pós-fixado produz: 1-2-3-6-7-9-8-5.



[0.75] 8. 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 Kruskal** para cálculo da árvore de suporte mínima do grafo, indique qual das seguintes 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. A aresta que une os vértices **D** e **E** é a quinta a entrar na MST.
B. O custo da MST que se obtém é 33 e tem 9 arestas.
C. A aresta que une os vértices **C** e **H** não faz parte da MST.
D. Das arestas de peso ímpar, só 3 delas fazem parte da MST.

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.

[6.0] 9. Uma empresa financeira dispõe de uma forma muito eficaz de prever o valor futuro das ações da companhia XCom, cotada em bolsa, e quer usar essa informação para efectuar nos próximos N dias **uma** compra e **uma** venda de ações. Para isso precisa de determinar o **melhor dia para comprar** ações e o **melhor dia para vender** de forma a **maximizar o lucro**. A empresa abre um concurso internacional para ver quem a pode ajudar.
Assuma que a empresa pode fornecer uma tabela, `int price[]`, de dimensão N, com o valor futuro das ações da XCom para os próximos N dias, tal que na k-ésima posição é guardado o valor no dia k (assuma que estamos no dia 0).
Assuma ainda que a empresa pode fazer *short sell*, ou seja, que pode vender antes de ter comprado, desde que venha a comprar no prazo de N dias.

[2.0] a) A primeira resposta que a empresa recebe vem de um concorrente que claramente não teve AED no curso. Ele sugere usar o código à direita para determinar o dia em que se deve comprar, `buy`, e vender, `sell`.
Nota: o valor de retorno das chamadas a `malloc()` deve ser testado mas esse código não é mostrado para simplificar.

Qual é a complexidade do algoritmo proposto pelo seu concorrente em função de N, tanto em termos de memória como em termos computacionais?

```
int **lucro, i, j;

/* constroi matriz lucro; testa malloc() */
lucro = (int**) malloc(N * sizeof(int*));
for (i = 0; i < N; i++) {
 lucro[i] = (int*) malloc(N * sizeof(int));
 for (j = 0; j < N; j++)
   lucro[i][j] = price[j] - price[i];
}

max = 0;
buy = sell = 0;
for (i = 0; i < N; i++)
 for (j = 0; j < N; j++)
   if (lucro[i][j] > max) {
     max = lucro[i][j]; buy=i; sell=j
   }
```

[1.5] b) **Sem escrever código**, descreva um algoritmo para resolver o problema que tenha uma **complexidade inferior** à do seu concorrente tanto em memória como em computação e que lhe permita saber quando a empresa deve comprar e quando deve vender as ações da XCom. Indique justificando, em função de N, qual a complexidade da sua solução.
Sugestão: Pense numa solução em que só usa a tabela `price[]` original (não usa matriz).

[2.5] c) Quando já julgava que o contrato era seu, a empresa descobre que afinal não pode fazer *short sell* e que tem de comprar **antes** de vender. Determinado a ficar com o contrato **escreva código** que implemente um algoritmo para resolver o problema e indicar quando a empresa deve comprar e quando deve vender as ações da XCom, assumindo que **só pode vender depois de ter comprado**. O seu algoritmo deve ter complexidade idêntica à do proposto em b) (ou melhor) e portanto ainda inferior à que o seu concorrente propôs em a).
Indique, justificando, qual a complexidade da sua solução em função de N.

[2.5] 10. Considere uma tabela de dispersão (Hash-table) de dimensão M=10 na qual são introduzidas sequencialmente as seguintes *strings*: "BOA", "SORTE", "PARA","O", "EXAME","DE","AED" (assuma que cada elemento da tabela pode armazenar uma *string* de qualquer tamanho).

Assuma que a função de dispersão utilizada é $f(x) = x \bmod 10$ em que x é o número (ver tabela abaixo) da última letra da *string* e "mod"calcula o resto da divisão, neste caso por 10. Assuma ainda que as colisões são resolvidas por procura linear na tabela.

[1.0] a) Determine o conteúdo da tabela de dispersão. Justifique.

[1.5] b) Identifique o problema da função de dispersão utilizada e proponha uma alternativa melhor.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

[3.0] 11. Você acaba de ser contratado por uma empresa que dispõe de N centros espalhados pela Europa nos quais conta com recursos especializados capazes de responder a qualquer solicitação. Cada centro k tem um número total de recursos, $\mathbf{R}_k$, que é conhecido, dos quais $\mathbf{D}_k \leq \mathbf{R}_k$ estão disponíveis num dado momento. O valor de $\mathbf{D}_k$ está constantemente a ser atualizado à medida que o trabalho efetuado em cada centro termina e uma nova tarefa lhe é atribuída. O seu trabalho como gestor dos recursos é de definir para cada nova tarefa que lhe é pedida qual o centro onde essa tarefa deve ser realizada. Quando uma tarefa lhe é apresentada, é também indicado o número de recursos, P, que necessita para a sua realização.

[0.5] a) Rapidamente você descobre a solução, muito simples, que a empresa tem implementada para o problema: para determinar onde executar uma dada tarefa que requer P recursos, percorre-se a tabela $\mathbf{D}_k$ e determina-se o primeiro centro onde essa tarefa pode ser realizada, actualizando-se a estrutura de dados depois de atribuir essa tarefa ao centro escolhido.
Descreva, em função de N, a complexidade desse algoritmo.

[1.5] b) A solução anterior não satisfaz pois você percebe que o sucesso da empresa passa por **encontrar o mais rapidamente possível para cada tarefa o centro onde ela pode ser realizada**. Para resolver o problema você propõe que a empresa passe a guardar a informação dos recursos disponíveis D_K de outra forma. Descreva, justificando, que estrutura de dados você sugere para guardar a informação D_k dos recursos disponíveis em cada centro k de forma a determinar o mais rapidamente possível qual o centro que tem disponíveis P recursos.
Discuta a complexidade, em função de N, da sua solução em termos de memória ocupada e número de operações para encontrar a solução e manter a estrutura de dados atualizada.

[1.0] c) O seu chefe, uma pessoa muito íntegra, diz-lhe que além de ter uma solução rápida, ela também tem de ser justa, isto é, todos os centros devem ter aproximadamente o mesmo nível de trabalho atribuído. Para isso pede-lhe que invente um algoritmo que minimize a diferença de trabalho em cada centro, ou seja, que você tente fazer uma atribuição que procure igualar a carga em todos os centros. De que forma mudaria a estrutura de dados que propos em b), ou alteraria a procura para otimizar a solução de acordo com esta solicitação do seu chefe?
Discuta a complexidade, em função de N, da sua solução em termos de memória ocupada e número de operações para encontrar a solução e manter a estrutura de dados atualizada.

[2.5] 12. Considere a árvore do problema 7.

[0.5] a) Indique a sequência de elementos resultante de um varrimento **pré-fixado** na árvore.

Utilizando agora os elementos por essa sequência obtenha:

[1.0] b) Uma árvore binária **ordenada**, com esses elementos (que não precisa de ser balanceada).

[1.0] c) Uma árvore binária **ordenada e balanceada AVL** com esses elementos.