DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

AED - 2018-2019 - 1º Semestre
Algoritmos e Estruturas de Dados

1º Exame, 9 de janeiro de 2019, 18h30m Duração: 3 horas

Prova escrita, individual e sem consulta

NOME:____________________________________ NÚMERO:____________

PARTE I - Questões de Escolha Múltipla (A/B/C/D)

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 seguintes valem 0.75 valores. Estas 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
Resposta						

1. Seja um acervo com 10 elementos, todos distintos: $1 - x - 9 - 16 - 20 - 13 - y - 34 - 19 - 22$. Após a operação de remoção do elemento mais prioritário o acervo resultante foi: $9 - x - 13 - 16 - 20 - 22 - y - 34 - 19$. Qual das seguintes afirmações é verdadeira?

   A. $x < 9$ e $y > 13$ B. $x < 13$ e $y < 34$
   C. $x < 16$ e $y > 22$ D. $x > 9$ e $y > 13$

2. Considere a seguinte função recursiva, em que a função `quicksort` se refere à implementação padrão do algoritmo de ordenação "Quicksort"(i.e., sem qualquer melhoria para tratamento de casos especiais) e a sua execução deixa `table` ordenada crescentemente:

```
int geometric(float *table, int l, int r)
{
   if (r < l) return 1;
   if (r == l) return table[r];
   else {
      quicksort(table, l, r);
      return sqrt(geometric(table, l, (l+r)/2) * geometric(table, (l+r)/2+1, r));
   }
}
```

   Qual dos seguintes pares, recorrência e respectiva solução, descreve a complexidade temporal da função `geometric`?

   A. $C_N = 2C_{N/2} + N \lg N$, $\mathcal{O}(N \lg^2 N)$ B. $C_N = 2C_{N/2} + N^2$, $\mathcal{O}(N^2)$
   C. $C_N = 2C_{N/2} + N \lg N$, $\mathcal{O}(N^2)$ D. $C_N = 2C_{N/2} + N^2$, $\mathcal{O}(N^2 \lg N)$

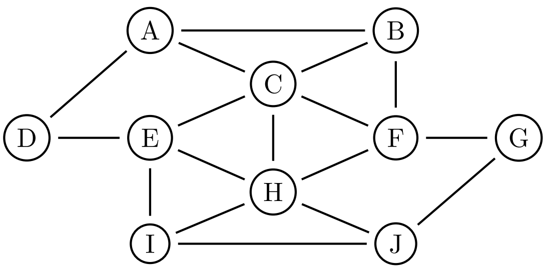
3. Suponha que se pretende ordenar alfabeticamente um conjunto de N strings acessíveis **indirectamente** através de um conjunto de ponteiros disponíveis numa tabela de dimensão N. Assuma que as strings têm em média comprimento M. O custo computacional médio desta operação utilizando o algoritmo de *Selection Sort* é de

   A. $\mathcal{O}(N^2 M)$ B. $\mathcal{O}(MN \log(N))$ C. $\mathcal{O}(NM^2)$ D. $\mathcal{O}(N^2 M^2)$

4. Um dos algoritmos propostos para solução do problema da conectividade foi o algoritmo de *Quick Find*. Este algoritmo resolve o problema original dividindo-o em dois sub-problemas: procura (*find*) e união (*union*). A complexidade computacional associada a N execuções do sub-problema de procura num universo de N objectos é, no pior caso, de:

A. $\mathcal{O}(1)$	B. $\mathcal{O}(N)$	C. $\mathcal{O}(N \log(N))$	D. $\mathcal{O}(N^2)$

5. Um **ciclo de Euler** num grafo é um caminho que partindo de um dado nó do grafo percorre todas as arestas do grafo exactamente uma vez e termina de novo no nó de partida.
Considere o grafo indicado em baixo à esquerda e assuma que o mesmo não é direccionado mas é ponderado (mas nesta pergunta ignore as ponderações indicadas à direita).



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

Diga qual das seguintes afirmações é **falsa**:

A. Não é possível encontrar um ciclo de Euler que parta de A nem de B.
B. Num ciclo de Euler há nós que poderão ser visitados apenas uma vez.
C. É possível que haja um ciclo de Euler partindo do nó H.
D. Um ciclo de Euler pode passar mais do que uma vez pelo mesmo nó.

6. Suponha um grafo direccionado e ponderado com 1000 nós e 4000 arestas. Numa máquina de 64 bits (inteiros e ponteiros ocupam 8 bytes), quantos bytes serão no mínimo necessários se o grafo for representado por uma lista de arestas:

A. 96 kB	B. 104 kB	C. 128 kB	D. 8 MB

PARTE II - Questões de Escolha Binária (V/F)

Preencha as respostas na tabela (usando apenas letras maiúsculas – **V**(erdadeira) ou **F**(alsa)). Todas as questões de escolha múltipla seguintes valem 0.50 valores. Estas questões de escolha múltipla não respondidas ou erradas são cotadas com 0 valores.

Questão	7	8	9	10	11	12	13	14
Resposta								

7. Dadas duas funções $f(N)$ e $g(N)$, ambas não decrescentes com N, quando $f(N)$ é limitada superiormente por $g(N)$ diz-se que $f(N) \in \Omega(g(N))$.

8. Como $\lg_2 3$ e $\lg_3 2$ se relacionam através de uma constante multiplicativa, pode afirmar-se que $N^{\lg_2 3} \in \Theta(N^{\lg_3 2})$.

9. Numa pesquisa numa árvore ordenada e balanceada com N nós, nunca são examinados mais do que $\mathcal{O}(\log N)$ nós.

10. No problema da conectividade, o algoritmo de *Quick Union* tem, no melhor caso, complexidade linear, mesmo sem compressão de caminho ou qualquer outro tipo de melhoramento.
11. Existem algoritmos recursivos que não gastam recursos de memória quando são executados.
12. Considere uma tabela de N elementos. Se a tabela for ordenada com *Selection Sort* o número máximo de vezes que um qualquer elemento pode mudar de posição é $N-1$ e o mínimo é 0.
13. Inserir um elemento x numa árvore binária ordenada e balanceada e retirar logo em seguida o mesmo elemento deixa sempre a árvore com a mesma configuração que antes da inserção de x.
14. Numa tabela de dispersão cuja resolução de colisões é feita por procura linear, para o mesmo número de objectos, o tempo médio de procura aumenta com o aumento da dimensão da tabela.

PARTE III - 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] 15. Pretende-se escrever uma função `int NRooks(Place *positions, int n)` em que `positions` é uma tabela com n coordenadas identificando a localização de n torres num tabuleiro de xadrez de $n \times n$, não havendo mais peças nesse tabuleiro. O tipo `Place` está definido como

```
typedef struct _place {
   int x;
   int y;
} Place;
```

em que (x, y) representa a posição de uma torre no tabuleiro (tanto x como y são inteiros entre 1 e n, inclusive).

A função deverá determinar se todas as torres estão colocadas em posições tais que nenhum par se ataca mutuamente (duas torres atacam-se mutuamente se estiverem localizadas na mesma linha ou coluna). Se esse for o caso, a função deverá devolver -1. Caso contrário deverá devolver o índice da tabela `positions` onde estão as coordenadas da primeira torre nessa tabela que esteja a atacar outra peça, e.g., um número não negativo entre 0 e $N-2$.

Por exemplo, para $n = 8$, se a entrada for $[(2,3),(4,5),(6,5),(8,4),(3,4),(1,6),(7,7),(5,8)]$ a resposta deverá ser 1, porque a segunda torre está na mesma coluna, que a terceira (5).

[2.5] a) Proponha e implemente uma solução para este problema. Descreve detalhadamente a solução proposta. O algoritmo proposto deverá ser o mais eficiente possível, sendo que a cotação desta alínea será atribuída em função da eficiência da implementação proposta e a cotação máxima apenas será dada à implementação correta que tiver uma eficiência óptima.

[1.0] b) Como função de n, discuta de forma clara e detalhada a complexidade temporal do algoritmo por si proposto.

[1.5] c) Suponha agora que se pretendia desenvolver uma outra função

`int NQueens(Place * positions, int n)`

em que as torres são substituídas por rainhas (as rainhas atacam-se mutuamente se estiverem na mesma linha, coluna ou diagonal). Indique o que modificaria no seu algoritmo e em que medida essa modificação alteraria a complexidade temporal resultante.

Nota: Para o exemplo acima, esta segunda função deverá devolver 0 (índice da primeira rainha na tabela), porque a primeira rainha está na mesma diagonal que a segunda.

Responda em folhas separadas. Não se esqueça de mudar de folha

[4.5] 16. Considere o grafo do Problema 5 e assuma que o mesmo é não dirigido mas ponderado, com as ponderações indicadas na figura.

[2.5] a) Utilizando o **Algoritmo de Kruskal** determine a *Minimum Spanning Tree* (MST) do grafo. Indique todos os passos do seu algoritmo de forma apropriada que permita ver porque ordem e porque razão cada uma das arestas entra para a MST. No final represente a MST e indique o seu custo.

[1.0] b) Diga, justificando, qual a complexidade de execução do algoritmo de Kruskal num grafo genérico com V nós e E arestas. Distinga os casos em que o grafo é esparso ou quando é denso.

[1.0] c) É sabido que a MST de um grafo não é necessariamente única. Se no grafo indicado em vez do Algoritmo de Kruskal tivesse sido utilizado o **Algoritmo de Prim** com início no nó I, indique se a aresta I↔H faria ou não parte dessa MST? Justifique a sua resposta.

Responda em folhas separadas. Não se esqueça de mudar de folha

[2.0] 17. Para um dado grafo direccionado e ponderado com 12 vértices, representado por matriz de adjacências, aplicou-se o algoritmo de Dijkstra tomando o vértice 0 como fonte e o vértice 11 como fonte. O vector `st` que se obteve quando se usou o vértice 0 como fonte foi:

```
st0 = [-1 0 0 0 10 7 1 3 10 3 0 5]
```

[1.0] a) Trace a *Shortest Path Tree* (SPT) que está representada naquele vector. Justifique o que entender necessário para que a sua resolução seja facilmente compreendida por terceiros.

[1.0] b) Abaixo apresentam-se três possíveis vectores `st` (contendo informação sobre os antecessores de um dado nó na SPT) resultantes da aplicação do algoritmo de Dijkstra ao mesmo grafo tomando o vértice 11 como fonte. Indique se são todos, apenas dois, só um ou nenhum consistentes com o vector `st` obtido para o vértice 0 como fonte. Fundamente as suas conclusões de forma clara.

```
st1 = [11 0 0 11 9 3 9 5 11 11 0 -1]
st2 = [11 0 0 11 9 7 9 3 11 11 0 -1]
st3 = [11 0 3 11 9 7 9 3 11 11 0 -1]
```