Departamento de Ciência de Computadores FCUP

Algoritmos (CC4010) 2019/20

Exame (07.02.2020) *duração: 3h (+30')*

N.º Nome

1. [2.0] **Usando a definição matemática** das ordens de grandeza, **prove** a veracidade ou falsidade de:

a) $n^2 + 3n \log n \in O(500n)$

b) $\Omega(n^2) \cup \Omega(n^2 \log n) = \Omega(n^2 \log n)$

2. [1.0] A Seja **A** o problema de decidir se existe um caminho de s para t de comprimento maior ou igual a k num grafo $G = (V, E, \omega)$, dados G, s, t e k, com $\omega(e) \in \mathbb{Z}^+$, para $e \in E$. Seja **B** o problema de decidir se existe um caminho de s para t de comprimento menor ou igual a k em G, dados G, s, t e k. Seja **C** o problema de decidir se existe um ciclo de Hamilton num grafo G dado.

Assumindo que **P**≠**NP**, quais dos problemas **A**, **B**, **C** e 3-SAT se podem reduzir polinomialmente: (i) a **A**? (ii) a **B**? (iii) a **C**? (iii) a 3-SAT?
Refira apenas os **distintos** do indicado. Justifique sucintamente a resposta.

3. [1.0] Prove que o custo amortizado por operação quando incrementamos um *contador binário* de 0 a n, efetuando uma sequência de n incrementos de uma unidade é $\Theta(1)$. O contador é definido por um *array* de k bits, a contagem é efetuada módulo 2^k e o custo real de cada *flip* ($0 \to 1$ ou $1 \to 0$) é 1.

4. Sejam $A_1, A_2, \ldots, A_n$ matrizes de inteiros, tendo A_k dimensão $d_k \times d_{k+1}$, para $1 \leq k \leq n$. Pretendemos efetuar o produto $A_1 A_2 \cdots A_n$ com **número mínimo de multiplicações**, usando a **definição usual de produto de matrizes** mas explorando a associatividade. Por exemplo, para $A_1 : 2 \times 3$, $A_2 : 3 \times 5$ e $A_3 : 5 \times 2$, se usarmos $A_1(A_2 A_3)$ efetuamos $30 + 12 = 42$ multiplicações e se usarmos $(A_1 A_2)A_3$ efetuamos $30 + 20 = 50$. Recorde que se $C = A_1 A_2$ então C tem dimensão $d_1 \times d_3$ e no cálculo de $C[i,j] = \sum_{p=1}^{d_2} A_1[i,p] \times A_2[p,j]$ efetuamos d_2 multiplicações, para $1 \leq i \leq d_1$, $1 \leq j \leq d_3$.

a) [1.2] Apresente a recorrência que define o número mínimo de multiplicações para calcular $A_k A_{k+1} \ldots A_m$, com $1 \leq k \leq m \leq n$, e o número de soluções ótimas alternativas. Represente-os por N_{km} e S_{km}.

b) [0.3] Use a recorrência para calcular o número de soluções ótimas de uma instância com $d = [2,2,2,2,2]$ e $n = 2$.

c) [1.0] Escreva uma função (em pseudocódigo) para calcular o valor do mínimo para $A_1 A_2 \cdots A_n$ e o número de soluções ótimas alternativas, sendo dados n e o *array* d de dimensões como parêmetros. A função imprime os valores pedidos. Pode implementar funções auxiliares. Deve seguir uma abordagem de **programação dinâmica**.

5. [2.5] Caraterize a complexidade temporal de algoritmos/implementações (eficientes), no **melhor** caso (caixa à esquerda) e no **pior** caso (caixa à direita) para:

a) extrair o valor mínimo de uma *heap* de mínimo, com n inteiros;

b) localizar um ponto relativamente a um polígono convexo com n vértices;

c) ordenar um vetor de n inteiros por um método comparativo eficiente;

d) *Jarvis march* (*gift-wraping*) para obter o invólucro convexo de n pontos em $\mathbb{R}^2$.

N.º Nome

Justifique *sucintamente* a resposta para uma das alíneas da questão **5.**.

6. [3.0] Recorde o problema *unit task scheduling*, em que todas as tarefas têm duração 1, prazos limite e penalizações se forem realizadas para lá desses prazos.

a) Considere a instância seguinte, sendo d_i o prazo limite para execução da tarefa i sem penalização e p_i o montante a pagar se executar a tarefa após esse prazo.

i	1	2	3	4	5	6	7	8	9	10	11	12
d_i	3	4	4	1	2	3	5	9	2	5	9	9
p_i	21	20	22	15	20	25	10	5	18	30	1	15

Apresente as tarefas que serão **realizadas dentro do prazo** pela ordem em que são escolhidas no algoritmo dado nas aulas (em caso de empate, opte pela tarefa com índice menor) . Indique o calendário para execução das 12 tarefas, com penalização mínima, que resulta de alocar cada tarefa o mais tarde possível, à medida que **o algoritmo** as vai processando, sem ultrapassar 12 slots de tempo . Indique a penalização .

b) Escreva **o algoritmo** referido em **6a)** em pseudocódigo. Deve ter **complexidade** $O(n^2)$. Indique **as propriedades estruturais** que o algoritmo explora e que permitem justificar a sua correção.

7. [2.0] A recorrência $T(n) = T(n/5) + T(7n/10) + cn$ surge na análise de complexidade do algoritmo de seleção ordinal determinístico *"mediana das medianas de 5"*. Explique porquê e apresente a prova de que $T(n) \in \Theta(n)$ usando a árvore de recursão. Pode assumir que n é múltiplo de 5.

8. [2.0] Considere a função QUICKSORT definida por

```
QUICKSORT(x, a, b)
    Se a < b então
        t ← MYPARTITION(x, a, b)
        QUICKSORT(x, a, t − 1)
        QUICKSORT(x, t + 1, b)
```

sendo x é um *array* de n inteiros $x[1], \ldots, x[n]$, e a e b inteiros tais que $1 \leq a \leq b \leq n$.

Escreva MYPARTITION em pseudocódigo, de modo que QUICKSORT$(x, 1, n)$ ordene o vetor por **ordem decrescente**. Demonstre a **correção** do programa.

N.º Nome

9. [1.5] Considere o algoritmo para cálculo do produto de dois inteiros positivos, $\mathbf{x}$ e $\mathbf{y}$, representados numa base B (fixa) com k dígitos, que aplica divisão e conquista (*divide-and-conquer*), e usa o facto de

$$\mathbf{xy} = (B^{k/2}\mathbf{x_2} + \mathbf{x_1})(B^{k/2}\mathbf{y_2} + \mathbf{y_1}) = B^k(\mathbf{x_2y_2}) + B^{k/2}(\mathbf{x_2y_1} + \mathbf{x_1y_2}) + (\mathbf{x_1y_1})$$

e de $\mathbf{x_2y_1} + \mathbf{x_1y_2} = (\mathbf{x_1} + \mathbf{x_2})(\mathbf{y_1} + \mathbf{y_2}) - \mathbf{x_1y_1} - \mathbf{x_2y_2}$ para reduzir o número de produtos auxiliares, sendo $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{y}_1$ e $\mathbf{y}_2$ inteiros com $k/2$ dígitos base B. Os produtos entre inteiros com $k/2$ dígitos são obtidos de forma análoga (se $k \geq 4$) e os produtos por $B^{k/2}$ e B^k são efetuados por deslocamento à esquerda (*left shift*). Admita que tal deslocamento é realizado *dígito a dígito* e que k potência de 2.

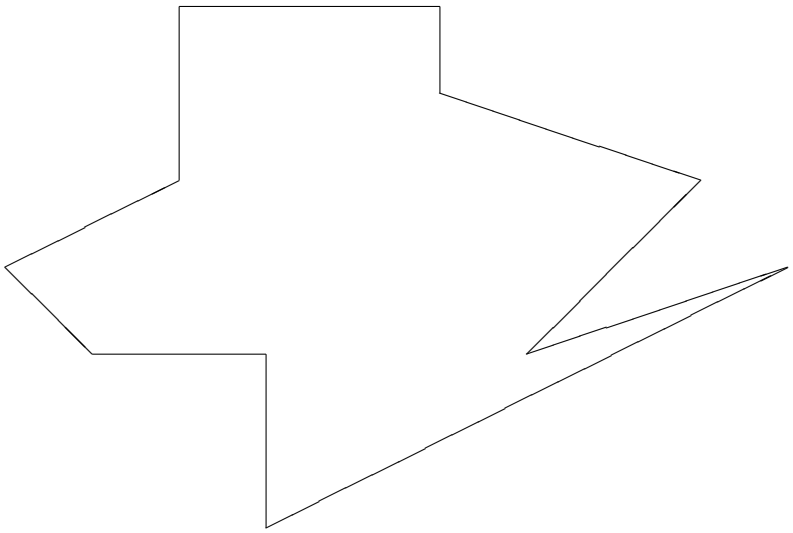
Indique a recorrência que define a complexidade do método e resolva-a aplicando o Master Theorem.

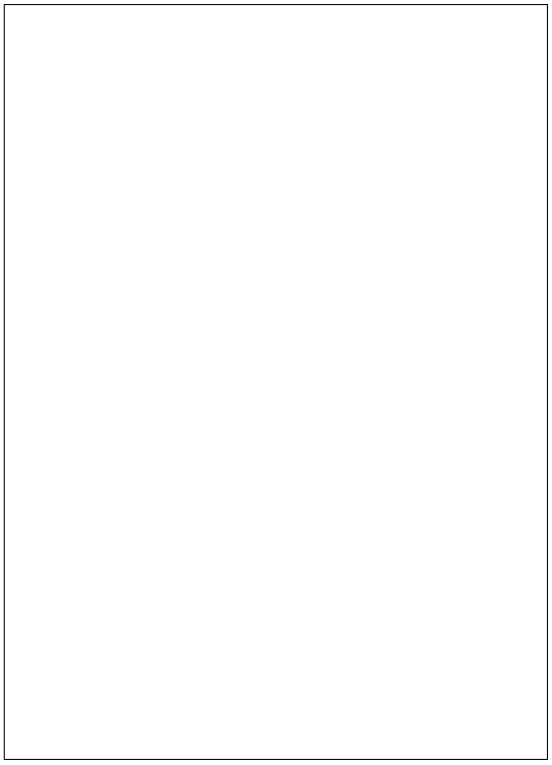
10. Para ordenar n inteiros positivos, $a_1, \ldots, a_n$, por ordem crescente, dadas as suas representações base B, com k dígitos, vamos aplicar *radix sort* (partindo do dígito menos significativo), usando *counting sort* como algoritmo auxiliar. O inteiro a_i está na linha i de uma matriz A, sendo $A[i][1]$ o dígito mais significativo.

a) [0.5] Explique o que é um algoritmo de ordenação estável e porque é que a estabilidade é relevante.

b) [1.0] Indique a complexidade de *radix sort* se: (i) se trocar os elementos da matriz A; (ii) se não trocar explicitamente, mas usar um *array* p para indicar a posição final de a_i. Justifique sucintamente, descrevendo os passos principais de *counting sort* nessa aplicação e a sua complexidade.

11. [1.0] Explique em que consiste o algoritmo de Ghosh para o problema de cobertura de polígonos com número mínimo guardas (colocados em vértices) e ilustre a sua aplicação ao polígono indicado, cujos vértices são $(3,0), (9,3), (6,2), (8,4), (6,5), (6,6), (2,6), (2,4), (0,3), (1,2), (3,2)$.





Master theorem:

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence $T(n) = aT(n/b) + f(n)$, where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log_2 n)$.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$, for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Stirling's approximation:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1 + \Theta(1/n)) = \sqrt{2\pi n}\left(\frac{n}{e}\right)^{\alpha_n}, \quad \text{with } 1/(12n+1) < \alpha_n < 1/(12n)$$

Some useful results:

$$\log(\prod_{k=1}^{n} a_k) = \sum_{k=1}^{n} \log a_k \qquad \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}, \quad \text{for } |x| < 1$$

If $(u_k)_k$ is an arithmetic progression (i.e., $u_{k+1} = r + u_k$, for some constant $r \neq 0$), then $\sum_{k=1}^{n} u_k = \frac{(u_1+u_n)n}{2}$.

If $(u_k)_k$ is a geometric progression (i.e., $u_{k+1} = ru_k$, for some constant $r \neq 1$), then $\sum_{k=1}^{n} u_k = \frac{u_{n+1}-u_1}{r-1}$.

If $f \geq 0$ is continuous and a monotonically increasing function, then

$$\int_{m-1}^{n} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m}^{n+1} f(x)dx$$
