Departamento de Ciência de Computadores — FCUP

Algoritmos (CC4010) — 2016/17

Exame (23.01.2017) — *duração: 3h*

Das três questões assinaladas por **(***)**, resolva apenas duas.

1. Considere as funções MYPARTITION e QUICKSORT assim definidas, sendo x é um *array* de n inteiros $x[1], \ldots, x[n]$, e a e b inteiros tais que $1 \leq a \leq b \leq n$.

```
MyPartition(x, a, b)
1.  z = x[a]
2.  j = a
3.  for i = a + 1 to b do
4.      if x[i] < z then
5.          Exchange x[i] with x[j + 1]
6.          j = j + 1
7.  Exchange x[a] with x[j + 1]
8.  return j + 1
```

```
QuickSort(x, a, b)
8.  if a < b then
9.      t = MyPartition(x, a, b)
10.     QuickSort(x, a, t − 1)
11.     QuickSort(x, t + 1, b)
```

a) Apresente o valor de retorno e o estado do *array* x depois de executar MYPARTITION(x, a, b) com:
(i) $x = [7, 4, 5, 9, 2, 7, 9, 8, 7, 10, 4]$, $a = 1$, $b = 11$; (ii) $x = [7, 4, 5, 9, 2, 7, 9, 8, 7, 10, 4]$, $a = 7$, $b = 9$.

b) No caso geral, qual é o estado de x e o valor de retorno após a execução MYPARTITION(x, a, b)? Na justificação da resposta, indique e demonstre o invariante de ciclo que carateriza o estado das variáveis a, b, x, j e i e z no momento em que a condição de paragem de ciclo é avaliada em cada iteração.

c) Sem alterar as instruções 1. a 4., corrija a função MYPARTITION de modo que QUICKSORT(x, a, b) ordene $x[a], \ldots, x[b]$ por ordem crescente (i.e., não decrescente).

d) Na continuação da alínea anterior, justifique a terminação e correção da função QUICKSORT(x, a, b).

2. **(***)** Baseando-se em MYPARTITION (versão corrigida) e QUICKSORT, apresente em pseudocódigo a função de QUICKSELECT(x, a, b, k), para seleção do k-ésimo menor elemento de $x[a], \ldots, x[b]$, com complexidade temporal esperada $O(b - a + 1)$. Justifique a correção e complexidade sucintamente.

3. Seja v um vetor (*array*) com n com elementos, $v[1], \ldots, v[n]$, em que cada elemento tem dois campos: o campo key, que é um inteiro positivo não superior a k; o campo $name$ que é uma sequência de carateres. **O vetor v encontra-se ordenado por ordem alfabética (crescente).**

Pretende-se imprimir o conteúdo de v por ordem decrescente de valor *key*, mantendo os elementos que têm a mesmo valor de key ordenados por ordem alfabética (crescente).

a) Apresente em pseudocódigo uma função ORDENA(v, n, b, k) para obter em b uma permutação de $1, \ldots, n$ que define a ordem pela qual os elementos de v devem ser escritos. A função deve ter complexidade temporal e espacial $O(n + k)$. Se necessário, use $v[i].name$ e $v[i].key$ para aceder aos campos. (Estamos sempre a supor que os vetores são passados por referência.)

b) Justifique a correção e complexidade da função que apresentou.

(Continua, v.p.f.)

4. **(***)** Apresente a ideia central da prova de que o algoritmo determinístico de seleção ordinal "mediana das medianas de 5" (*medians of 5*) tem complexidade temporal $O(n)$.

5. **(***)** Seja $T(n)$ uma função não negativa e crescente tal que $T(n) = T(a_1 n) + T(a_2 n) + \cdots T(a_k n) + bn$, onde k é um inteiro positivo fixo, $b \geq 1$ e $a_1 + a_2 + \cdots + a_k < 1$, sendo $a_1, a_2, \ldots a_k, b$ constantes reais positivas. Assuma que $T(n) \in O(1)$ para valores suficientemente pequenos de n.

a) Prove que $T(n) \in \Omega(n)$, diretamente a partir da definição formal de $\Omega(n)$.

b) Por análise da árvore de recursão, prove que $T(n) \in \Theta(n)$.

6. Recorde que no problema *unit task scheduling* são dadas n tarefas e há que definir a sequência pela qual as n tarefas são executadas de forma a minimizar a penalização total. A tarefa i tem duração unitária, um prazo limite d_i e uma penalização p_i se não for executada dentro do prazo limite. Em cada unidade de tempo só se pode executar uma tarefa.

a) Recordando a prova de que o problema tem estrutura de matróide pesado, explique sucintamente porque é que é equivalente dizer que um conjunto de tarefas A é independente (ou seja, é formado por tarefas *livres*) e se ter $N_t(A) \leq t$, para todo $t \geq 0$.

b) Admita que todas as tarefas devem ser executadas (ainda que com penalização). Sendo $A \subseteq \{1, \ldots, n\}$ um conjunto *ótimo* formado por tarefas livres, indique a ordem pela qual as tarefas em A devem ser executadas e também a ordem pela qual as restantes tarefas serão executadas.

7. Considere uma instância do problema "Minimum Edit Distance", para transformação de uma sequência $X = x_1 \ldots x_n$ numa sequência $Y = y_1 \ldots y_m$, em que os custos das operações de *inserção* e *remoção* são iguais a 2 e o custo de uma *substituição* é 3.

a) Apresente a recorrência que define o custo ótimo da transformação.

b) Apresente um programa que construa uma tabela S em que a entrada S_{ij} contém o custo ótimo de edição de $X_i = x_1 \ldots x_i$ em $Y_j = y_1 \ldots y_j$ e além disso também uma informação que permite recuperar uma solução com esse custo. Use dois campos *cost* e *sol*, ambos com complexidade espacial $O(1)$, para tal.

8. No problema BIN PACKING são dados n itens com pesos $p_1, \ldots, p_n$, tais que $0 < p_i \leq 1$, para todo i, e há que os distribuir por latas de capacidade unitária, usando o menor número de latas possível.

No problema PARTITION são dados n inteiros positivos $a_1, a_2, \ldots, a_n$, e há que decidir se existe uma partição $\{S, T\}$ do conjunto de índices $\{1, 2, \ldots, n\}$ tal que $\sum_{i \in S} a_i = \sum_{i \in T} a_i$. Sabe-se que PARTITION é um problema **NP-completo**.

a) Prove que as instâncias de PARTITION com $a_j > \sum_{j \neq i} a_i$, para algum j, são trivialmente decidíveis.

b) Dada uma instância de PARTITION, com $a_j \leq \sum_{j \neq i} a_i$, para todo j, definimos uma instância de BIN PACKING com $p_j = 2a_j / \sum_{i=1}^{n} a_i$, para todo j. Justifique que se trata de uma redução polinomial que permite decidir PARTITION em tempo polinomial se algum dos algoritmos seguintes existir e conclua que não podem existir a menos que P=NP: (i) um algoritmo polinomial que calcule uma solução ótima para BIN PACKING; (ii) um algoritmo de aproximação polinomial de razão c para BIN PACKING, com $c < 3/2$.

(FIM)

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:

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

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

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