Departamento de Ciência de Computadores FCUP

Algoritmos (CC4010) 2015/16

Exame (10.02.2016) *duração: 3h*

N.º | Nome

1. Sejam $f(n) = 2n^2 - n + 1$ e $g(n) = 7n^2 + 3n$, para $n \geq 1$. Prove que $n^2 \in \Theta(f(n)) \cap \Omega(g(n))$, usando **diretamente** as definições das ordens de grandeza $\Theta(f(n))$ e $\Omega(g(n))$.

2. Considere a função MyPartition definida abaixo, sendo x um *array* de n inteiros, com $x[1], \ldots, x[n]$ **distintos**, e a e b inteiros tais que $1 \leq a \leq b \leq n$. Assuma que RandomInt(a, b) retorna um inteiro em $[a, b]$ e tem complexidade temporal $O(1)$.

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

a) O que se pode garantir sobre o estado de x após a chamada MyPartition$(x, 1, n)$, se $n > 1$ e a primeira chamada da função RandomInt retornar 1? Explique.

b) Descreva em pseudocódigo uma versão MyQuickSort de *quicksort*, que use MyPartition. Qual é o resultado da chamada MyQuickSort$(x, 1, n)$? Que propriedade assume sobre o resultado das chamadas a MyPartition(x, a, b)?

c) Indique a condição que carateriza o estado das variáveis x, a, b, y, i e j em MyPartition(x, a, b) no final da iteração i do ciclo 5.-8., antes de incrementar i, e que permite justificar a propriedade que referiu na alínea anterior. Prove que tal condição é um invariante de ciclo.

d) É possível o tempo de execução de uma dada chamada de MyQuickSort$(x, 1, n)$, para um certo x ser da ordem $\Omega(n^2)$? Que importância tem o bloco de instruções 1.-2. de MyPartition na análise da complexidade de MyQuickSort? Que implicações tem a sua remoção?

3. Considere o problema da seleção do **k-ésimo elemento <u>máximo</u>** de um *array* x constituído por n elementos $x[1], \ldots, x[n]$ **distintos**.

a) Apresente em pseudocódigo uma adaptação de *QuickSelect* para resolver tal problema, com tempo esperado da ordem $O(n)$. Deve usar a função MyPartition definida na questão **2.**.

b) Justifique a correção e diga qual é a principal razão para a diferença de complexidade entre MyQuickSort (descrito na questão **2.**) e este algoritmo?

(Continua, v.p.f.)

4. Seja $T = \{t_1, t_2, \ldots, t_n\}$ o conjunto de tarefas que uma empresa teria de realizar num certo dia e seja k o número máximo de tarefas que pode realizar, com $k \leq n$, fixo. Seja p_i a penalização que tem se não realizar a tarefa t_i, para $1 \leq i \leq n$. O valor de p_i é sempre um **múltiplo positivo de 10 e não superior a 100**. Pretendemos encontrar um conjunto de tarefas a realizar de forma a minimizar a penalização total. Caso haja **empates**, deve realizar a que tiver identificador menor.

a) Resolva a instância seguinte e justifique que a solução determinada é correta.

$k = 5$

t_i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	20	30	40	50	80	70	60	20	40	60	50	30

b) No caso geral, designe por I_k o conjunto dos subconjuntos de T com no máximo k elementos.

1. Prove que (T, I_k) define um matróide.
2. Usando pseudo-código, apresente um algoritmo para determinar uma solução ótima, com complexidade temporal $O(n)$.
3. Prove a correção do algoritmo que apresentou e justifique que tem complexidade $O(n)$.

Dos CINCO problemas seguintes, resolva apenas TRÊS

5. Aplique RadixSort (partindo do dígito menos significativo) para ordenar 5678, 5663, 2358, 2761, 5345, 5533, 6783, 5195, apresentando o resultado após cada iteração. Através desse exemplo, explique a necessidade de o algoritmo de ordenação utilizado nos passos intermédios ser estável.

6. Recorde o método de Graham para determinar o invólucro convexo (*convex hull*) de n pontos no plano. De que pressupostos depende a conclusão de que tem complexidade $O(n \log n)$?

7. Considere o problema da distribuição de N caixas de morangos por M lojas, sendo v_{ij} o lucro que obtém se entregar i caixas à loja j.

a) Defina uma recorrência para cálculo do lucro máximo que se pode obter se se distribuir k caixas pelas lojas $1..j$ (podendo algumas não receber qualquer caixa).

b) Escreva um algoritmo para calcular o lucro máximo usando programação dinâmica. Que complexidade tem?

8. Seja $G = (V, E)$ um grafo não dirigido. Considere o algoritmo seguinte para obter uma solução aproximada de *minimum vertex cover* (cobertura de G por vértices, com cardinal mínimo).

```
ApproxVertexCover(V, E)
  S := ∅
  while (E ≠ ∅) do
      remove an edge e = (u, v) from E
      remove all edges incident to u or v
      S := S ∪ {u, v}
  return S
```

a) Apresente a prova de que se trata de um algoritmo de aproximação **polinomial** de **razão 2**.

b) Comente a afirmação: "A menos que P=NP, não existe um algoritmo de aproximação polinomial de razão 1 (um) para tal problema".

9. Por análise da árvore de recursão, apresente a prova do **caso 1** do Master Theorem, assumindo que n é potência de b (de expoente natural).

(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:

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

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

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