Departamento de Ciência de Computadores — FCUP

Algoritmos (CC4010) — 2016/17

Teste (13.12.2016) — *duração: 3h*

---

1. Sejam $f(n) = 2n\log_2(n)$ e $g(n) = 5n + \log_2 n$, para $n \geq 1$.

a) Usando a classe $O(g(n))$, $\Theta(g(n))$ ou $\Omega(g(n))$ caraterize a ordem de grandeza de $f(n)$.

b) Prove a relação indicada, usando **diretamente** as definições das ordens de grandeza.

2. 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 inteiro não negativo fixo. Seja p_i a penalização que terá se não realizar a tarefa t_i, para $1 \leq i \leq n$, sendo p_i inteiro positivo. Pretendemos selecionar as tarefas a realizar de modo a minimizar a penalização total.

a) Apresente em linhas gerais um algoritmo para resolver o problema com complexidade temporal $O(n \log n)$. Justifique a complexidade desse algoritmo.

b) Aplique o algoritmo à instância seguinte:

$k = 5$

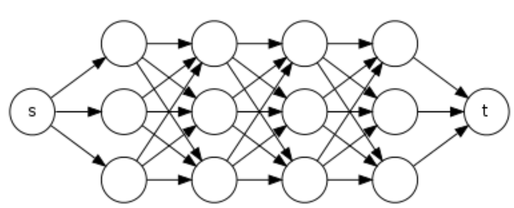
t_i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	22	31	41	41	80	70	61	20	50	50	50	32

c) No caso geral, prove que: (i) o número de tarefas realizadas em qualquer solução ótima é $\min(n, k)$ e (ii) a solução obtida pelo algoritmo que apresentou é ótima.

3. Em que consiste o problema da determinação do *comprimento* da subsequência comum mais longa? Explique sucintamente as ideias subjacentes à recorrência analisada nas aulas para sua determinação. Como pode ser calculado usando programação dinâmica e espaço linear (no tamanho da instância)?

4. Seja $G = (V, E)$ um grafo dirigido acíclico (DAG). Designe por $Adj[v]$ o conjunto dos vértices adjacentes a v, para $v \in V$. Assuma que esse conjunto é representado por uma lista ligada simples. Pretendemos contar o número de caminhos em G de s para t, sendo s e t dois nós dados.

a) Resolva o problema para a instância representada abaixo e explique as vantagens da utilização de programação dinâmica.



b) Apresente, em pseudocódigo, uma função $\textsc{Conta}(s, t)$ para resolver o problema, que use programação dinâmica (com memoização). Assumindo que os cálculos podem ser efetuados em $O(1)$, caraterize a complexidade temporal e espacial da função.

c) Face à instância dada em **4a)**, é razoável assumir que os cálculos podem ser efetuados em $O(1)$?

(Continua, v.p.f.)

5. Considere uma *stack* suportada por um *array* V e uma variável top que designa o índice da primeira posição livre de V, com as operações usuais $\textsc{Push}(x)$ e $\textsc{Pop}()$ assim definidas:

$\textsc{Push}(x)$	$\textsc{Pop}()$
$V[top] := x$	$top := top - 1$
$top := top + 1$	retorna $V[top+1]$

Considere um modelo de custos em que a inserção de um elemento e a remoção de um elemento têm custo real 1. Suponha que inicialmente V tem 1 posição e considere a possibilidade de a operação de $\textsc{Push}(x)$ ser substituída para permitir aumentar a capacidade da stack de 1 (uma) posição se esta se encontrar cheia (realocando espaço e inserindo os elementos que estavam na stack no novo array). O custo real da expansão da estrutura de dados é igual ao número de elementos que é necessário transferir.

a) Suponha que usa o método contabilístico (*accounting*) com custo amortizado de 3 unidades para a operação de PUSH e de 0 unidades para POP e para a expansão. Prove que esses valores não são adequados e justifique que o custo amortizado de uma sequência de n operações quaisquer não pode ser constante.

b) Considere a alternativa em que se *duplica* a dimensão do vetor em cada expansão. Explique a interpretação que pode ter um custo amortizado de 3 unidades para $\textsc{Push}(x)$ e de 0 para as restantes operações. Mostre ainda que neste caso o custo total amortizado nunca é inferior ao custo real.

6. Considere o problema *unit task scheduling* e a prova de que 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 $t \geq 0$.

7. 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 1, usando o menor número de latas possível. Por exemplo, se $n = 5$ e os pesos fossem 0.9, 0.8, 0.5, 0.3, 0.3 seriam necessárias quatro latas.

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

a) Mostre que PARTITION se reduz polinomialmente a BIN PACKING se se tomar $p_i = 2a_i / \sum_{i=1}^n a_i$. Quantas latas são usadas se a decisão for "sim" para a instância de PARTITION? E quantas latas são usadas se for "não"?

b) Sabendo que PARTITION é NP-completo, prove que se existir um algoritmo aproximação polinomial de razão c para BIN PACKING, para algum $c < 3/2$, então P=NP.

c) Suponha que para se obter uma solução aproximada de BIN PACKING, se tomam os itens de 1 a n, sucessivamente, colocando o item corrente na primeira lata em que couber. Se não existir, abre-se uma nova e coloca-se lá. A ordem é definida pela ordem de abertura. Esta estratégia designa-se por "first-fit". Justifique que:

1. Qualquer que seja a estratégia, o número de latas necessárias seria sempre maior ou igual a $\sum_{i=1}^n p_i$.
2. Usando "first-fit", não se pode ter duas latas pelo menos semi-vazias (ou seja, prenchidas a 50% ou menos), pelo que se "first-fit" usar m latas, então $\sum_{i=1}^n p_i > \frac{m-1}{2}$.
3. Aplicando a estratégia "first-fit" usa-se quando muito o dobro de latas que usaria uma solução ótima.

d) Justifique que BIN PACKING pertence à classe APX e é APX-hard.

(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

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