Departamento de Ciência de Computadores FCUP

Algoritmos (CC4010) 2020/21

Teste (25.11.2020) *duração: 2h*

N.º ______ Nome ______

Das perguntas 4 a 9, resolva quatro apenas.

1. Recordando que $n^2 < \; n^2 + n\log n < 2n^2$, para todo $n \geq 2$, **prove** a veracidade ou a falsidade de $n^2 + n\log n \;\in\; \Omega(500n^2) \;\wedge\; n^2 + n\log n \;\notin O(n^2/100)$. Deve usar **diretamente** a definição matemática.

2. Admitindo que n é potência de 4, resolva a recorrência $T(1) = 1$, $T(n) = 2T(n/4) + 1$, se $n > 1$:

a) por aplicação do Master Theorem.

b) por análise da árvore de recursão.

3. Considere o problema da ordenação de um *array* de n inteiros $x[1], \ldots x[n]$, por ordem decrescente, com $0 < x[i] < 100000$, para $1 \leq i \leq n$ (ou seja, $x[i]$ pode ser representado na base 10 com 5 dígitos). Recorde que para um inteiro não negativo v, o k-ésimo dígito menos significativo da representação base 10 é dannndo pela expressão $(v\%10^k)/10^{k-1}$, onde $\%$ é o resto da divisão inteira, com $k \geq 1$.

a) Apresente os passos principais do algoritmo de ordenação por inserção (*insertion sort*) dado nas aulas.

b) Recordando a definição das classes $O(\cdot)$, $\Omega(\cdot)$ e $\Theta(\cdot)$, explique e corrija as incorreções da afirmação: "o algoritmo de ordenação por inserção tem complexidade temporal assintótica $O(n^2)$ no pior caso e $\Theta(n)$ no melhor caso, sendo $\Omega(n^2)$."

c) Explique se se pode usar *insertion sort* como algoritmo auxiliar em *Radix Sort* para efetuar a ordenação, e quais seriam as vantagens ou desvantagens face a *counting sort*.

4. Seja P um polígono com n vértices, no plano, dado pela sequência dos seus vértices $v_1, \ldots v_n$, ordenados no sentido anti-horário (CCW). Como é que se pode verificar se P é convexo? Apresente os passos principais e a sua complexidade temporal.

5. Apresente os passos principais do algoritmo para verificar se um ponto $p \in \mathbb{Z}^2$ pertence ao interior de um polígono **convexo** com n vértices, com tempo de execução $O(\log n)$. Como se justifica essa complexidade?

6. Por aplicação do método contabilístico (*accounting*), apresente a prova de que o custo amortizado por operação de uma sequência de n operaçoes (PUSH/POP) sobre uma stack de inteiros, suportada por um *array*, com duplicação do espaço se necessário (por realocação) é $O(1)$.

7. Comente a afirmação: *o algoritmo de seleção ordinal "mediana das medianas de 5" (medians of 5) determina a mediana de um* array *de n elementos, sendo menos geral do que o algoritmo* quickselect *(aleatorizado), que se aplica para determinar o k-ésimo menor elemento do* array. Apresente ainda a ideia central da prova de que o algoritmo "mediana das medianas de 5" (medians of 5), tem complexidade temporal $\Theta(n)$.

(Continua, v.p.f.)

8. Dado um *array* $a[1] \ldots a[n]$, de n inteiros distintos, *ordenado* por ordem crescente, e dado um inteiro x, pretendemos determinar k tal que $|x - a[k]| = \min_{1 \leq i \leq n} |x - a[i]|$, isto é, $a[k]$ está a distância mínima de x. Usando como modelo uma árvore de decisão (binárias), prove que qualquer algoritmo comparativo que resolva o problema tem complexidade temporal $\Omega(\log n)$, no pior caso.

9. Explique que problemas os algoritmos de Karatsuba e de Strassen, dados nas aulas resolvem. Porque é que são interessantes?

10. Considere a aplicação da função seguinte para compactar um array x de n inteiros positivos, por substituição de blocos com k ou mais inteiros iguais (em posições contíguas) por `0 v c`, sendo `v` o valor que se repete e `c` o comprimento do bloco. Assuma que $k \geq 4$ e as posições do *array* são indexadas a partir de 1. Por exemplo, se $k = 4$, e a sequência inicial fosse

`9 9 7 7 7 7 7 6 5 5 7 7 7 2 2 2 2 2 2 2 2 2 2 2 2 5 1 1 2 3 4 5 2 2 2 2`

a sequência final seria `9 9 0 7 5 6 5 5 7 7 7 0 2 12 5 1 1 2 3 4 5 0 2 4`.

```
COMPACTAR(x, n, k) :
    if n < k then return n;
    t := 0;  c := 1;  v := x[1];  i := 2;
    while i ≤ n do
        if x[i] = v then  c := c + 1;
        else  t := REESCREVER(x, t, c, v, k);  c := 1;  v := x[i];
        i := i + 1;
    t := REESCREVER(x, t, c, v, k);
    return t;

REESCREVER(x, t, c, v, k) :
    if c ≥ k then
        x[t + 1] := 0;   x[t + 2] := v;   x[t + 3] := c;  return t + 3;
    while (c > 0) do
        t := t + 1;  x[t] := v;  c := c − 1;
    return t;
```

a) Caraterize a complexidade temporal assintótica das funções REESCREVER e COMPACTAR, no pior caso e no melhor caso. Justifique sucintamente as suas respostas.

b) Indique o invariante de ciclo que permite concluir que a função COMPACTAR resolve corretamente o problema. Explique como é que se conclui a correção a partir dessa condição.

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 $T(n) = \Theta(f(n))$.

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

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