Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Estruturas de Dados

LISTA DE EXERCÍCIOS 2

Recorde que, nas funções de complexidade trabalhadas neste curso, o domínio é sempre o conjunto dos naturais, e o contradomínio é sempre o conjunto dos reais não-negativos.

Exercício 1. Resolva o problema URI #1472 implementando a busca binária com recursão.

Exercício 2. Mostre que se, sendo k um inteiro positivo, $2^n + n^k + \sqrt{n \log n} = \theta(2^n)$.

Exercício 3. Livro do Cormen, Exercício 3.1-3.

Exercício 4. Escreva um algoritmo que, recebendo um inteiro positivo n, um *array* $\boldsymbol{A}[0..n-1]$ de n elementos inteiros ordenados em ordem não-decrescente, e um elemento x, insere o elemento x no *array* preservando a ordenação e imprime o *array* resultante. Qual a complexidade de tempo do seu algoritmo no melhor caso? E no pior caso? Implemente seu algoritmo em linguage C ANSI.

Exercício 5. Como visto em sala de aula, sendo $f, g: \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$, escrever $f(n) = o(g(n))$ significa que $f(n) = O(g(n))$, mas $f(n) \neq \theta(g(n))$. Equivalentemente, podemos definir que $f(n) = o(g(n))$ se para todo $\varepsilon > 0$ existe um natural n_0 tal que $f(n) < \varepsilon g(n)$ para todo $n \geq n_0$. Mostre que, para todo k, $n^k = o(2^n)$ (ou seja, mostre que toda função polinomial é assintoticamente inferior à função 2^n).

Resolução. Sendo ε uma constante real positiva *qualquer*, queremos mostrar que existe um natural n_0 tal que $n^k < \varepsilon 2^n$ para todo $n \geq n_0$. Como $n^k < \varepsilon 2^n$ é equivalente a

$$\frac{1}{2^n} < \frac{\varepsilon}{n^k},$$

que é equivalente a

$$\lg \frac{1}{2^n} < \lg \frac{\varepsilon}{n^k},$$

que é equivalente a

$$-n < \lg \varepsilon - \lg n^k,$$

que, por sua vez, é equivalente a

$$n > k \lg n - \lg \varepsilon.$$

Ou seja, queremos encontrar um n_0 natural tal que $n > k \lg n - \lg \varepsilon$ para todo $n \geq n_0$. Temos dois casos:

1. Se $\lg \varepsilon \geq 0$, então basta mostrar que existe um n_0 tal que para todo $n \geq n_0$ valha que $n > k \lg n$, pois isto implica $n > k \lg n - \lg \varepsilon$, já que $-\lg \varepsilon < 0$. Neste caso, podemos simplesmente tomar n_0 como qualquer inteiro tal que $n_0 / \lg n_0 > k$, já que a função $n / \lg n$ é crescente.

2. Se $\lg \varepsilon < 0$ (o que acontece quando $\varepsilon < 1$), então basta mostrar que existe um n_0 tal que para todo $n \geq n_0$ valha que

   $$n > k \lg n - \lg \varepsilon \lg n = (k - \lg \varepsilon) \lg n,$$

   pois isto implica $n > k \lg n - \lg \varepsilon$, já que $-\lg \varepsilon > 0$. Neste caso, podemos simplesmente tomar n_0 como qualquer inteiro tal que $n_0 / \lg n_0 > k - \lg \varepsilon$, também porque a função $n \lg n$ é crescente. □

Exercício 6. Mostre que $1/n = o(1)$.

Exercício 7. É verdade que, se $f(n) = c^n$ para alguma constante $c > 1$, então $f(n) = O(2^n)$?

Exercício 8. Mostre que, se $f(n) = 2^{O(\log n)}$, então existe um inteiro positivo k tal que $f(n) = O(n^k)$.

Exercício 9. Escreva um algoritmo que, recebendo um inteiro positivo n, utiliza busca binária para devolver $\lfloor \sqrt{n} \rfloor$. Qual a complexidade de tempo do seu algoritmo? Implemente seu algoritmo em linguagem C ANSI.

Exercício 10. Livro do Cormen, Exercício 3-2.

Exercício 11. Livro do Cormen, Exercício 3-4.