Construção e Análise de Algoritmos

lista de exercícios 11

1. Como vimos nessa aula, o algoritmo `Seleção-DC2.0()` é baseado na decomposição dos elementos do vetor em grupinhos de 5.

   Mas, porque 5?

   Quer dizer, porque não 3 ou 7?

   Verifique se as variantes do algoritmo baseadas na decomposição do vetor em grupinhos de 3 ou 7 também possuem a garantia de tempo $O(n)$.

2. **Seleção em listas ordenadas**

   Consider a seguinte variante do problema da seleção.

   Nós temos duas listas ordenadas

   

   e o problema consiste em encontrar o k-ésimo menor elemento dentre todos os elementos das duas listas.

   Apresente um algoritmo de divisão e conquista para esse problema, e analise a sua complexidade.

   Como as duas listas estão ordenadas, a ideia é que desta vez a seleção pode ser feita em tempo menor que $O(n)$.

3. (OPCIONAL)

   Imagine que alguém lhe dá uma rotina `Med()` que encontra a mediana (i.e., o elemento do meio) de um vetor de inteiros `V[1..n]` (desordenado) em tempo $O(n)$.

   Apresente o pseudo-código de um procedimento `Terc()` que encontra o $n/3$-ésimo menor elemento do vetor `V[1..n]`, utilizando a rotina `Med()`.

   Qual o tempo de execução do seu procedimento `Terc()`? Explique.

   Agora, apresente o pseudo-código de um outro procedimento `Med2()` que encontra a mediana do vetor `V[1..n]`, utilizando o seu procedimento `Terc()`.

   Qual o tempo de execução do seu procedimento `Med2()`? Explique.

4. (OPCIONAL)

   Implemente o algoritmo `Seleção-DC2.0()` na sua linguagem de programação favorita.

   A seguir, compare os tempos de execução das seguintes versões do algoritmo Quicksort:

   - versão que utiliza um pivô aleatório no procedimento de partição
   - versão que utiliza o melhor pivô possível, encontrado por `Seleção-DC2.0()`

   Apenas a segunda versão possui a garantia de tempo $O(n \log n)$ no pior caso.

   Mas, será que isso vale a pena na prática?