MC448 – Projeto e Análise de Algoritmos
Lista de Exercícios 6

Orlando Lee

A numeração abaixo nos exercícios refere-se à **segunda edição** (CLRS).

1 Ordenação em tempo linear

1. (CLRS 8.2-3) Suponha que no algoritmo Counting-Sort abaixo

   CountingSort(A, B, n, k)
   1 **para** $i \leftarrow 0$ **até** k **faça**
   2 $\quad C[i] \leftarrow 0$
   3 **para** $j \leftarrow 1$ **até** n **faça**
   4 $\quad C[A[j]] \leftarrow C[A[j]] + 1$
   5 **para** $i \leftarrow 1$ **até** k **faça**
   6 $\quad C[i] \leftarrow C[i] + C[i-1]$
   7 **para** $j \leftarrow n$ **decrescendo até** 1 **faça**
   8 $\quad B[C[A[j]]] \leftarrow A[j]$
   9 $\quad C[A[j]] \leftarrow C[A[j]] - 1$

   a linha 7 seja trocada pela linha

   7 $\quad$ **para** $j \leftarrow 1$ **até** n **faça** .

   Mostre que o algoritmo ainda funciona. Este algoritmo modificado é estável?

2. (CLRS 8.2-4) Descreva um algoritmo que dado n inteiros no intervalo $[0\,..\,k]$, pré-processa a entrada de modo que qualquer consulta do tipo: "quantos desses n inteiros caem no intervalo $[a\,..\,b]$?" pode ser respondida em tempo constante. O algoritmo deve ter tempo de processamento $\Theta(n+k)$.

3. (CLRS 8.3-3) Use indução para provar que Radix-Sort funciona. Em que ponto da prova você usou o fato que a ordenação dos dígitos é estável?

4. (CLRS 8.3-4) Mostre como ordenar n inteiros no intervalo $[0\,..\,n^2-1]$ em tempo $\emptyset(n)$.

2 Estatísticas de ordem

Você pode usar o fato que existe um algoritmo *linear* que resolve o problema da seleção.

1. (CLRS 9.3-1) No algoritmo linear Select visto em aula para o Problema da Seleção, os elementos de entrada são divididos em grupos de 5. O algoritmo continua sendo linear se dividirmos em grupos de 7? Argumente por que o algoritmo **não** é linear se dividirmos em grupos de 3.

2. (CLRS 9.3-3) Mostre como QuickSort pode ser modificado de modo que tenha complexidade de tempo $O(n \lg n)$ no pior caso.

3. (CLRS 9.3-5) Suponha que você possui um algoritmo linear do tipo "caixa-preta" que resolve o problema de encontrar a mediana. Descreva um algoritmo linear que resolve o problema da seleção (encontrar o i-ésimo menor) para todo i.

4. (CLRS 9.3-6) O k-ésimo **quantiles** de um vetor de n inteiros são os $k-1$ elementos que dividiriam o vetor, se este estivesse ordenado, em k partes de tamanho quase iguais (diferindo de no máximo 1). Por exemplo, a mediana é o segundo *quantiles*. Descreva um algoritmo de complexidade $O(n \lg k)$ que determina o k-ésimo *quantiles* de um vetor de n inteiros.

5. (CLRS 9.3-7) Descreva um algoritmo de complexidade $O(n)$ que dado um conjunto S de n números reais *distintos* e um inteiro positivo $k \leq n$, determina os k números em S que estão mais próximos da mediana de S. **Observação:** a proximidade entre dois elementos é dada pela diferença, em valor absoluto, entre ambos.

6. (CLRS 9.3-8) Sejam $X[1\,..\,n]$ e $Y[1\,..\,n]$ dois vetores **ordenados**. Descreva um algoritmo de complexidade $O(\lg n)$ para determinar a mediana dos $2n$ elementos de X e Y. Escreva um pseudo-código. Note que a mediana de $2k$ elementos é o elemento x que é maior ou igual a $k-1$ elementos e menor ou igual a k elementos.

7. (CLRS Problema 9-1) Dado um conjunto de n números, queremos listar os i maiores elementos em ordem crescente usando um algoritmo baseado em comparações. Considere os algoritmos baseados nas seguintes idéias:

   (a) Ordene os números e liste os i maiores.

   (b) Construa uma fila de prioridade com os n números e chame EXTRACT-MAX i vezes.

   (c) Use um algoritmo para encontrar o i-ésimo maior número e particione o conjunto em torno dele. Então ordene e imprima os i maiores elementos.

   Analise a complexidade desses algoritmos em função de n e i. Determine para que valores de i (em função de n) um algoritmo é melhor que os outros.