Lista de exercícios 06

Os seguintes exercícios são para aprofundar no estudo da matéria. Nos casos em que fala Exercícios e problemas (com números), se referem aos exercícios/problemas da 3^{ra} edição do livro de texto principal (T. Cormen, C. Leiserson, R. Rivest, C. Stein. "Introduction to Algorithms").

Ordenação linear

1. Exercícios: 8.1-1, 8.1-2, 8.2-1, 8.2-4, 8.3-1, 8.3-3, 8.4-1, 8.4-2,

2. Suponha que o vetor $A[1..n]$ tenha apenas números de $\{1, \ldots, n^2\}$, mas que no máximo $\log\log n$ desses números aparecem no vetor. Projete um algoritmo que ordene A em tempo substancialmente menor que $O(n \log n)$. Calcule a complexidade e demonstre a correção do algoritmo.

3. Considere o mesmo problema do exercício anterior, mas sem a restrição de que no máximo haja que no máximo $\log\log n$ dos números, isto é, o vetor pode ter quaisquer n inteiros de $\{1, \ldots, n^2\}$ com repetições ou sem. Projete um algoritmo que ordene A em tempo linear. Demonstre a correção do algoritmo.

Estatísticas de ordem

4. Exercícios: 9.1-1, 9.2-1, 9.2-2, 9.2-4, 9.3-1, 9.3-2, 9.3-5, 9.3-7, 9.3-9,

5. Problemas: 9-1

6. Suponha que usamos RandomizedSelect para selecionar o menor elemento de um vetor $\langle 3; 2; 9; 0; 7; 5; 4; 8; 6; 1\rangle$. Descreva uma sequência de partições que resulte em uma execução de pior caso de RandomizedSelect.

7. Projete um algoritmo de divisão e conquista para encontrar o menor e o maior elementos de um conjunto. O algoritmo deve usar no máximo $3n/2$ comparações (para $n = 2^k$). Você pode apontar a razão desse algoritmo requerer menos que $2n - 3$ comparações do algoritmo trivial?

8. A entrada é um conjunto S contendo n números reais e um número real x.

(a) Projete um algoritmo para determinar se há dois elementos de s cuja soma é exatamente x. O algoritmo deve executar em tempo $O(n \log n)$.

(b) Suponha agora que o conjunto S é dado de forma ordenada. Projete um algoritmo para resolver esse problema em tempo $O(n)$.

9. Diz-se que um vetor $A[1..n]$ tem um elemento majoritário se mais da metade de suas entradas forem iguais. Dado um vetor, a tarefa é projetar um algoritmo eficiente para determinar se o vetor possui um elemento majoritário e, em caso afirmativo, para encontrar esse elemento. Os elementos do vetor não são necessariamente de algum domínio ordenado como os inteiros e, portanto, não pode haver comparações da forma "$A[i] > A[j]$?". (Pense nos elementos da matriz como arquivos GIF, digamos). No entanto, você pode responder perguntas do tipo: "$A[i] = A[j]$?" em tempo constante.

(a) Mostre como resolver este problema em tempo $O(n \log n)$. Para isso, divida o vetor A em dois vetores A_1 e A_2 com a metade do tamanho. Conhecer um elemento majoritário de A_1 e A_2 ajuda você a descobrir o elemento maioritário de A? Se for assim, então você pode usar a abordagem de divisão-e-conquista.

(b) Você pode dar um algoritmo de tempo linear? Aqui está outra abordagem de divisão e conquista:

- Pareie elementos de A e obtenha $\frac{n}{2}$ pares.
- Olhe para cada par: se os dois elementos forem diferentes, descarte os dois; se eles são os mesmos, mantenha apenas um deles.

Formalize e mostre que, após este procedimento, há no máximo $\frac{n}{2}$ elementos restantes e que entre eles existe um elemento maioritário se e somente se A tiver. Depois mostre como resolver o problema em tempo linear.

10. Definimos como os k-ésimos quantis de um conjunto de elementos, os $k-1$ elementos que dividem o conjunto ordenado em k conjuntos de tamanhos quase iguais (a diferença de tamanho entre quaisquer dois conjuntos é no máximo um). Dê um algoritmo de tempo $O(n \log k)$ para listar os k-ésimos quantis de um conjunto.