Ordenação

Visão geral de algoritmos de ordenação

Questão 1. (FKM) Dada uma matriz de números retangular, ordene cada linha da matriz. Em seguida, ordene cada coluna da matriz. Mostre que as linhas da matriz continuam ordenadas.

Questão 2. (Manber) (6.12) Em alguns casos, a entrada de um algoritmo de ordenação já está quase ordenada, o que significa que o número de elementos fora de ordem é pequeno. Descreva como os algoritmos de ordenação que você conhece se comportam com sequências quase ordenadas. Que algoritmo você usaria? (Você é encorajado a projetar o seu próprio algoritmo)

Questão 3. Suponha que queremos ordenar um conjunto de n registros armazenados em disco. Se n for relativamente pequeno, a melhor estratégia costuma ser carregar todos os dados na memória RAM, ordená-los e gravá-los de volta. Se n for muito grande, então estratégias diferentes precisam ser consideradas. Considere duas situações: quando os dados são comparados por uma rotina dada como caixa-preta; e quando se deve ordenar por identificadores únicos, que são números de 1 até n. Pesquise e disserte brevemente (200/400 palavras) sobre algoritmos de ordenação nessas situações. Escreva um texto coeso (i.e., escreva um texto contínuo evitando dar duas respostas independentes) e considere as operações necessárias para os algoritmos citados e as implicações da sua estratégia com relação ao tempo de latência e cópia em disco, quantidade de memória RAM disponível, etc.

Questão 4. (Manber) (6.29) A entrada são d sequências de elementos tais que cada sequência já está ordenada e há um total de n elementos. Projete um algoritmo que executa em tempo $O(n \log d)$ para juntar todas as sequências em uma única sequência ordenada.

Questão 5. (Sedgewick) Explique como você classificaria um baralho de cartas com a restrição de que as únicas operações permitidas são olhar os valores das duas primeiras cartas, inverter a ordem das duas cartas no topo e mover a carta mais acima para o fundo do baralho.

(a) Escreva um algoritmo para ordenar as cartas em ordem decrescente de valor e demonstre que o algoritmo está correto.

(b) Analise a complexidade do algoritmo e mostre que sua análise é justa, i.e., construa uma instância do problema e mostre que ela corresponde a um pior caso.

Fila de prioridade e Heapsort

Questão 6. (CLRS) Exercícios: 6.1-1, 6.1-2, 6.1-3, 6.1-4, 6.1-5, 6.1-6, 6.2-1, 6.2-2, 6.2-6, 6.3-2, 6.4-1, 6.4-3,

Questão 7. (Manber) A entrada é um *heap* de tamanho n (em que o maior elemento está no topo), dado como um vetor, e um número real x. Projete um algoritmo para determinar se o k-ésimo maior elemento no heap é menor ou igual a x. No pior caso, seus algoritmo deve executar em tempo $O(k)$, independente do tamanho do heap. Você pode usar espaço de tamanho $O(k)$. (Note que você não tem que encontrar o k-ésimo maior elemento; você só precisa determinar sua relação com x.)

Questão 8. (Dasgupta et al., adaptado) Uma árvore binária completa com n nós pode ser representada por um vetor B indexado por elementos $1, 2, \ldots, n$. Nesta questão, queremos representar uma árvore d-ária completa usando um vetor D, indexado de 1 a n. Uma árvore d-ária completa representada em um vetor tem as seguintes propriedades:

- cada nó que não é folha tem um número constante d de filhos;

- todas as folhas estão no nível mais abaixo;
- percorrer o vetor da esquerda para a direita é equivalente a percorrer a árvore em largura.

(a) Dado um índice j, qual o índice de D corresponde ao pai de j? E quais índices correspondem aos filhos de j? Demonstre isso.

(b) Escreva um algoritmo linear para, dado um vetor D com n elementos, criar um *maxheap* d-ário. Defina a propriedade de *heap* correspondente e demonstre a correção do seu algoritmo e sua complexidade.