USP-ICMC-BInfo
Introdução à Ciência de Computação II - SCC-0501
Prof. João Luís - 26/09/2011

Lista de Exercícios 4

1. Re-escreva a rotina da classificação por bolha com sucessivas passagens em direções opostas.

2. Modifique o programa do *quicksort* de modo que, se um subvetor for pequeno, a classificação por bolha seja usada.

3. Explique por que a classificação por seleção direta é mais eficiente do que a classificação por bolha.

4. Escreva uma rotina *combine(x)* que aceite um vetor x no qual as sub-árvores enraizadas em $x[1]$ e $x[2]$ sejam *heaps* e que modifique o vetor x de maneira que ele represente um único *heap*.

5. Modifique a rotina *quicksort* para usar uma classificação por inserção simples quando um sub-arquivo tiver um tamanho menor que s. Determine através de experimentação, qual valor de s deve ser usado para obter a eficiência máxima.

6. Determine qual das seguintes classificações é mais eficiente:

   i. a classificação por inserção simples,

   ii. a classificação por seleção direta,

   iii. a classificação por bolha.

7. Escreva um algoritmo para a rotina *merge(x,i,j,k)*, que presuma que $x[i]$ até $x[j]$ e $x[j+1]$ até $x[k]$ estão classificados e intercale os dois em $x[i]$ até $x[k]$.

8. Implemente os métodos *bubblesort* (bolha), *selectsort* (seleção simples), *insertsort* (inserção simples) e *quicksort*, comparando o número de comparações e trocas que realizam na ordenação de seqüências.

9. Explique como seria possível melhorar o método *bubblesort*, armazenando não apenas a informação da troca, mas também a posição do vetor onde ocorreu a troca. Implemente essa modificação.

10. No método *insertsort*, a cada passo, o menor elemento é procurado para que seja inserido na seqüência já ordenada. Essa procura pode ser realizada seqüencialmente ou por busca binária. Analise o desempenho de ambas as abordagens.

11. Discuta como a escolha do pivô pode influenciar no desempenho do método *quicksort*. Proponha estratégias para a escolha do pivô, visando melhorar seu desempenho.

12. Faça um teste de mesa com cada método de ordenação estudado, utilizando as seguintes seqüências de dados de entrada:

Veja a próxima página...

i.

2	4	6	8	10	12

ii.

11	9	7	5	3	1

iii.

5	7	2	8	1	6

iv.

2	4	6	8	10	12	11	9	7	5	3	1

v.

2	4	6	8	10	12	1	3	5	7	9	11

13. Compare o desempenho dos métodos estudados, identificando em que casos você utilizaria cada um deles. Justifique suas afirmações.

14. Considere a ordenação de n números armazenados no arranjo A, localizando primeiro o menor elemento de A e permutando esse elemento contido em $A[1]$. Em seguida, encontre o segundo menor elemento de A e o troque pelo elemento $A[2]$. Continue dessa maneira para os primeiros $n-1$ elementos de A. Escreva o pseudocódigo para esse algoritmo conhecido como *ordenação por seleção*. Qual invariante do laço esse algoritmo mantém? Por que ele só precisa ser executado para os primeiros $n-1$ elementos, e não para todos os elementos? Forneça os tempos de execução do melhor caso e do pior caso da ordenação por seleção em notação $\mathcal{O}$.

15. Vamos supor que estamos comparando diferentes implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho n, a ordenação por inserção é executada em $8n^2$ etapas, enquanto a ordenação por intercalação é executada em $64n\ log\ n$ etapas. Para que valores de n a ordenação por inserção supera a ordenação por intercalação?

References

[1] Nakamiti, Gilberto, *Listas de Exercícios de Estruturas de Dados II*," Engenharia de Computação. PUC-Campinas, 2007.

[2] Oliveira, Maria Cristina Ferreira de, *Primeira Lista de Exercícios - Análise de Algoritmos e Notação Assintótica*," SCE0181 - Introdução à Ciência da Computação II. ICMC-USP, 2008.

[3] Tenenbaum, A. M., Langsam, Y., Augestein, M. J., *Estruturas de Dados Usando C*. Makron Books, 1995.