

**MAC 121 – Algoritmos e Estruturas de Dados I****Segundo semestre de 2016****Lista de exercícios—Ordenação**

1. Ordene a sequência abaixo usando os métodos Mergesort, Quicksort, Heapsort e Bubblesort:

12   23   5   9   19   0   24   4   1   12   21   2   5   14   -9   10   14

Em cada um dos casos, qual o número de comparações e de trocas feitas durante a ordenação?

2. Queremos ordenar um vetor de  $n$  elementos que é composto apenas por dois valores distintos que não conheço *a priori*. Escreva um algoritmo que faça no máximo  $n - 1$  comparações para ordenar o vetor. Use um vetor auxiliar se necessário, mas pense em uma versão sem usar um vetor auxiliar. Dica: pense no separa do quicksort.
3. Encontre permutações do vetor com elementos  $1, 2, \dots, 10$  que forcem:
- (a) o algoritmo Quicksort a executar o número máximo de comparações;
  - (b) o algoritmo Quicksort a executar o número máximo de trocas;
  - (c) o algoritmo Bubblesort a executar o número máximo de trocas.
  - (d) o algoritmo da versão “coquetel” do Bubblesort a executar o número máximo de comparações;
4. Faça uma função Merge que recebe vetores ordenados  $A$  (com  $m$  elementos) e  $B$  (com  $n$  elementos) e devolve um vetor ordenado  $C$  (com  $m + n$  elementos) que resulta da intercalação de  $A$  e  $B$ . Faça duas versões do algoritmo: uma iterativa e uma recursiva. Encontre instâncias (dados) em que o algoritmo acima faz:
- (a)  $m$  comparações;
  - (b)  $n$  comparações;
  - (c)  $m + n - 1$  comparações.
5. Considere o seguinte trecho de programa:

```
for (i = 0; i < n; i++) cont[i] = 0;
for (i=0; i< n; i++)
    for (j = 0; j < n; j++)
        if (V[j] < V[i]) cont[i]++;
```

- (a) Escreva um algoritmo que ordene um vetor  $V$  utilizando o trecho acima. Observe o que ocorre se  $V$  tiver elementos repetidos.
  - (b) Quantas comparações e quantas movimentações envolvendo os elementos do vetor  $V$  são feitas no melhor caso? Quantas no pior caso? Quantas no caso médio?
6. Considere o seguinte algoritmo (em pseudocódigo) para ordenação de um vetor  $V$ :

```

W = V;
enquanto W não está ordenado faça
    W := uma permutação de V que ainda não foi testada

```

- (a) Quantas comparações são necessárias para verificar se um vetor está ordenado?
  - (b) Quantas permutações deverão ser testadas no pior caso? E no caso médio?
  - (c) Baseado nas respostas aos itens acima, calcule quantas comparações esse algoritmo faz, no pior caso, para ordenar um vetor de  $n$  elementos. Repita para o melhor caso. Repita para o caso médio.
7. Em cada uma das situações abaixo, qual algoritmo de ordenação (visto em sala de aula ou não) é mais apropriado?
- (a) Uma lista de 10000 nomes parcialmente ordenada.
  - (b) Uma lista de 10000 nomes em ordem aleatória.
  - (c) Uma lista de 10000 inteiros positivos menores que 100.
  - (d) Uma lista de 10000 inteiros entre 1000 e 9999.
  - (e) Uma lista de 10000 números reais entre 0 e 1.
  - (f) Uma lista de nomes que não cabe na memória principal.
8. Considere as seguintes funções de ordenação de um vetor  $A$ :

```

void ordena (int A[], int n)
{
    int i, j, min;

    for(i = 0; i < n - 1; i++){ /* I */
        min = i;
        for (j = i+1; j < n; j++){
            if (A[j] < A[min]) /* II */
                min = j;
        }
        troca (A, i, min); /* III */
    }
}

```

```

void class (int A[], int n)
{
    int i, j;

    for (i = 1; i < n; i++)    /* I */
        for (j = i; j > 1; j--)
            if (A[j] < A[j-1]) /* II */
                troca (A, j, j-1); /* III */
}

```

Para cada um dos algoritmos responda às seguintes perguntas.

- (a) Para cada um dos algoritmos, qual a situação do vetor  $A$  a cada passagem pelo ponto I? Justifique.
  - (b) Baseado em sua resposta no item acima, mostre que cada uma das funções de fato ordena o vetor  $A$ .
  - (c) Qual o número máximo e mínimo de vezes que a comparação II é executada e em que situações ocorre?
  - (d) Idem para o comando III, que troca elementos.
9. Faça uma função recursiva que ordena um vetor  $A$  de  $n > 1$  elementos baseado no seguinte algoritmo: Obtenha um número  $p$  em  $[1..n]$ . Divida o vetor em duas partes, a primeira com  $p$  elementos e a segunda com  $n - p$ . Ordene cada uma das duas partes. Depois, supondo que  $A[1] \leq A[2] \leq \dots A[p]$  e  $A[p+1] \leq A[p+2] \leq \dots \leq A[n]$ , intercale as duas sequências de forma a completar a ordenação de  $A$ . Use vetores auxiliares se necessário.
  10. Considere o algoritmo do exercício anterior com  $p = 1$ , depois com  $p = \lfloor n/3 \rfloor$ , depois com  $p = \lfloor n/2 \rfloor$ . Você já viu antes o algoritmo correspondente a algum destes valores de  $p$ ? Qual das três escolhas sugeridas para  $p$  é a mais eficiente? Por quê?
  11. Considere uma matriz  $A_{m \times n}$  em que cada uma das linhas e cada uma das colunas estão ordenadas em ordem crescente (veja exemplo abaixo). Escreva um algoritmo que recebe uma matriz com esta propriedade e um elemento  $x$  e faz no máximo  $m + n$  comparações para verificar se  $x$  está ou não na matriz.

Exemplo:

$$\begin{pmatrix} 12 & 20 & 21 & 42 \\ 15 & 22 & 25 & 51 \\ 16 & 28 & 31 & 94 \\ 23 & 32 & 51 & 98 \\ 77 & 91 & 93 & 123 \end{pmatrix}$$

12. Dizemos que um vetor  $v$  tem uma inversão nas posições  $i$  e  $j$ , com  $i < j$  se  $v[j] > v[i]$ . Faça uma função que conta o número de inversões em um vetor  $v$  com  $n$  elementos em  $O(n \log n)$ . Dica: Pense no Mergesort.

13. Simule a execução do algoritmo `constroiHeap` dado em sala de aula com o vetor abaixo:

12   23   5   9   19   0   24   4   1   13   21   2   7   14   -9   10   14

14. Considere um vetor com 100 elementos organizado com um min-heap (o menor elemento na primeira posição). Em que posições do vetor pode estar:

- o segundo menor elemento.
- o quinto menor elemento.
- o maior elemento.

15. Faça uma função `void RemoveHeap(int *heap, int n, int k)` que remove o elemento na posição  $k$  do heap mantendo a estrutura de heap.

## Problemas de Online Judges

16. UVA 110 - Meta-Loopless Sorts
17. UVA 299 - Train Swapping
18. UVA 11495 - Bubbles and Buckets
19. UVA 11714 - Blind Sorting
20. URI 1252 - Sort! Sort! e Sort!!!
21. UVA 120 - Stacks of Flapjacks