Universidade Federal do Ceará - Campus Quixadá
QXD0010 – Estruturas de Dados – 2025.2
Prof. Atílio Gomes

Ordenação por Flutuação — BubbleSort

```
1 void bubblesort(int A[], int n) {
2   for (int i = 0; i < n-1; i++)
3     for (int j = n-1; j > i; j--)
4       if (A[j] < A[j-1])
5         int aux = A[j];
6         A[j] = A[j-1];
7         A[j-1] = aux;
8 }
```

Listing 1: Algoritmo de ordenação BubbleSort.

1. Escreva uma função que verifique se um vetor $A[0 \ldots n-1]$ de inteiros está em ordem crescente. Quantas comparações entre elementos do vetor sua função executa?

2. Ordene o vetor `A[0...5]`, que possui os elementos `20 12 28 05 10 18`, usando o algoritmo de ordenação por flutuação (BubbleSort). Mostre o vetor a cada passo do laço `for` mais externo.

3. Qual a complexidade de pior caso do BubbleSort? Qual a complexidade de melhor caso? Justifique sua resposta.

4. Um amigo lhe disse que é capaz de ordenar qualquer conjunto de três números com no máximo 2 comparações. O seu amigo está falando a verdade ou mentindo? Justifique sua resposta.

5. Um amigo lhe disse que é capaz de ordenar qualquer conjunto de quatro números com no máximo 5 comparações. O seu amigo está falando a verdade ou mentindo? Justifique sua resposta.

6. Um amigo lhe disse que é capaz de ordenar qualquer conjunto de cinco números com no máximo 7 comparações. O seu amigo está falando a verdade ou mentindo? Justifique sua resposta.

7. Um amigo lhe disse que é capaz de ordenar qualquer conjunto de seis números com no máximo 9 comparações. O seu amigo está falando a verdade ou mentindo? Justifique sua resposta.

8. Implemente um algoritmo que ordena um vetor de inteiros e retorna a quantidade de inversões que ocorreram.

9. Corretude do BubbleSort. Prove que o algoritmo Bubblesort está correto mostrando que os dois invariantes a seguir são verdadeiros:

   (a) No início de cada iteração do laço **for** mais interno (linhas 3-5), o subvetor $A[j \ldots n-1]$ consiste em uma permutação dos valores que estavam originalmente em $A[j \ldots n-1]$ antes do laço iniciar. Além disso, $A[j] = \min\{A[k] : j \leq k \leq n-1\}$, ou seja, o elemento na posição $A[j]$ é o menor dentre todos os elementos nesse subvetor.

(b) No início de cada iteração do laço **for** externo, o subvetor $A[0 \ldots i-1]$ consiste nos i menores valores originalmente em $A[0 \ldots n-1]$, ordenados em ordem crescente, e $A[i \ldots n-1]$ consiste nos $n-i$ valores restantes originalmente em $A[0 \ldots n-1]$.

Ordenação por Inserção — Insertion-Sort

```
1  void insertionsort(int A[], int n) {
2    int i, j, key;
3    for (j = 1; j < n; j++) {
4      key = A[j];
5      i = j-1;
6      while (i >= 0 && A[i] > key) {
7        A[i+1] = A[i];
8        i--;
9      }
10     A[i+1] = key;
11   }
12 }
```

Listing 2: Algoritmo de ordenação Insertion-Sort.

10. No código da função `insertionsort`, troque "`A[i] > key`" por "`A[i] >= key`". A nova função continua produzindo uma ordenação crescente de $A[0 \ldots n-1]$?

11. Escreva uma versão recursiva do algoritmo de ordenação por inserção.

12. Escreva uma função que rearranje um vetor `A[0...n-1]` de inteiros de modo que ele fique em ordem decrescente. Sua função deve se basear no algoritmo de ordenação por inserção. Escreva duas versões de sua função: uma iterativa e uma recursiva.

13. Pior caso. Descreva e analise uma instância de pior caso para o algoritmo de ordenação por inserção, ou seja, um vetor `A[0...n-1]` que leva o algoritmo a executar o maior número possível de comparações.

14. Melhor caso. Descreva e analise uma instância de melhor caso para o algoritmo de inserção, ou seja, um vetor `A[0...n-1]` que leva o algoritmo a executar o menor número possível de comparações.

15. Movimentação de dados. Quantas vezes, no pior caso, o algoritmo de ordenação por inserção copia um elemento do vetor de um lugar para outro? Quantas vezes isso ocorre no melhor caso?

16. Escreva uma versão do algoritmo de ordenação por inserção (Insertion-Sort) que tenha o seguinte invariante: no início de cada iteração, o vetor $A[j+1 \ldots n-1]$ é crescente.

17. O algoritmo de ordenação por inserção é estável? Explique.

18. No código da função `insertionsort`, troque a comparação "`A[i] > key`" por "`A[i] >= key`". A nova função faz uma ordenação estável de `A[0...n-1]`?

19. A **ordenação por inserção bidirecional** é uma modificação da ordenação por inserção simples da seguinte forma: um vetor de saída separado, de tamanho n, é reservado. Esse vetor de saída atua como uma estrutura circular. $x[0]$ é posicionado no elemento do meio

do vetor. Assim que um grupo contíguo de elementos estiver no vetor, será aberto espaço para um novo elemento deslocando todos os elementos menores um passo para a esquerda ou todos os elementos maiores um passo a direita. A escolha do deslocamento é feita para provocar o menor número de movimentações. Escreva uma função para implementar essa técnica.

Ordenação por Seleção — Selection-Sort

```
1  void selectionsort(int A[], int n) {
2    for (int i = 0; i < n-1; i++) {
3      int min = i;
4      for (int j = i+1; j < n; j++)
5        if(A[j] < A[min])
6          min = j;
7      int aux = A[i];
8      A[i] = A[min];
9      A[min] = aux;
10   }
11 }
```

Listing 3: Algoritmo de ordenação Selection-Sort.

20. Aplique a função `selectionsort` ao vetor `A[0...7]` que contém os elementos `88 99 33 99 44 77 99 11`, nesta ordem. Diga qual a sequência exata de pares (i, j) tais que `A[i]` é comparado com `A[j]` durante a execução do algoritmo. Repita o exercício para a função `insertionsort`.

21. Pior caso. Descreva e analise uma instância de pior caso para o algoritmo de ordenação por seleção, ou seja, um vetor `A[0...n-1]` que leva o algoritmo a executar o maior número possível de comparações.

22. Melhor caso. Descreva e analise uma instância de melhor caso para o algoritmo de seleção, ou seja, um vetor `A[0...n-1]` que leva o algoritmo a executar o menor número possível de comparações.

23. Movimentação de dados. Quantas vezes, no pior caso, o algoritmo de seleção copia um elemento do vetor de um lugar para outro? Quantas vezes isso ocorre no melhor caso?

24. Escreva uma versão recursiva do algoritmo de ordenação por seleção.

25. No código da função `selectionsort`, troque "`A[j] < A[min]`" por "`A[j] <= A[min]`". A nova função continua produzindo uma ordenação crescente de $A[0 \ldots n-1]$?

26. Os algoritmos de ordenação por seleção e ordenação por inserção rearranjam um vetor `A[0...n-1]` de modo que ele fique em ordem crescente. Qual dos dois é mais rápido no caso em que os elementos do vetor `A[0...n-1]` são todos iguais?

27. Escreva uma função que permute os elementos de um vetor $A[0 \ldots n-1]$ de modo que eles fiquem em ordem decrescente.

28. O algoritmo de ordenação por seleção é estável? Explique.

Ordenação — Questões Gerais

29. Implemente cada um dos algoritmos de ordenação estudados em sala de aula.

30. Implemente uma versão recursiva de cada um dos algoritmos de ordenação estudados em sala de aula.

31. Um vetor $v[p..r]$ está "arrumado" se existe $j \in [p, r]$ tal que $v[p..j-a] \leq v[j] < v[j+1..r]$. Escreva um algoritmo que decida se $v[p..r]$ está arrumado. Em caso afirmativo, seu algoritmo deve devolver o valor de j.

32. Dada a sequência de números: 13 7 11 2 5 17 7 13 4 6 7 3 7 10 54 13, ordene-a em ordem crescente segundo cado um dos algoritmo estudados em sala. Para cada algoritmo, mostre o número de comparações e trocas que realizam na ordenação de sequências.

33. Dos algoritmos estudados, quais são estáveis? Utilize a questão anterior para apoiar sua resposta.

34. Para cada um dos seis algoritmos de ordenação estudados em aula, responda as seguintes perguntas:

    1. Explique, resumidamente, o funcionamento do algoritmo.
    2. Qual a complexidade de melhor caso?
    3. Qual a complexidade de pior caso?

35. Usando lista duplamente encadeada, implemente cada um dos algoritmos de ordenação estudados em sala de aula.