Exercícios: Busca e ordenação

1. Defina formalmente o problema de ordenação.
2. Defina formalmente o problema de encontrar o menor valor de um vetor.
3. Forneça um exemplo de aplicação real que envolva o problema de ordenação e de encontrar o menor valor.
4. Escreva um algoritmo que receba valores em um vetor e imprima **ORDENADO** se o vetor estiver em ordem crescente.
5. Escreva um algoritmo que ordene de maneira descrescente (do maior para o menor).
6. Escreva um algoritmo que receba um vetor ordenado e um número extra e insira esse número na sua posição correta no vetor ordenado, deslocando os outros números se necessário.
7. Escreva um algoritmo que procure por um dado número em vetor ordenado.
8. Qual é o vetor resultante após as 4 primeiras trocas ao executar ordenação por seleção com o seguinte array inicial?
9. Implemente o algoritmo de ordenação por inserção visto em aula e conte o número total de cópias de valores do vetor dentro do while ao executar no seguinte array: 72 12 62 69 27 67 41 56 33 74
10. Escreva o vetor resultante ao aplicar o algoritmo de particionamento em duas partes no vetor seguinte: 26 65 45 73 10 18 78 93 70 49 23 22
11. Faça uma comparação entre todos os métodos de ordenação estudados em aula com relação a estabilidade (preservar ordem lexicográfica), ordem de complexidade levando em consideração comparações e movimentações.
12. Dada a sequência de números: 3 4 9 2 5 8 2 1 7 4 6 2 9 8 5 1, ordene-a em ordem não decrescente segundo os seguintes algoritmos, apresentando a sequência obtida após cada passo do algoritmo:
    (a) MergeSort
    (b) QuickSort
    (c) HeapSort
13. João diz ter desenvolvido um algoritmo que é capaz de ordenar qualquer conjunto de n números reais, fazendo apenas $O(n^{3/2})$ comparações. Você compraria este algoritmo? Justifique.
14. Uma ordenação por contagem de um vetor x de tamanho n é executada da seguinte forma: declare um vetor **count** e defina **count[i]** como o número de elementos menores que **x[i]**. Em seguida, coloque **x[i]** na posição **count[i]** de um vetor de saída (leve em consideração a possibilidade de elementos iguais). Escreva uma função para ordenar um vetor x de tamanho n usando esse método.

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

16. Faça um teste de mesa com cada método de ordenação estudado até o momento, utilizando as seguintes sequências de dados de entrada:

    (a) 2, 4, 6, 8, 10, 12

    (b) 11, 9, 7, 5, 3, 1

    (c) 5, 7, 2, 8, 1, 6

    (d) 2, 4, 6, 8, 10, 12, 11, 9, 7, 5, 3, 1

    (e) 2, 4, 6, 8, 10, 12, 1, 3, 5, 7, 9, 11

    (f) 8, 9, 7, 9, 3, 2, 3, 8, 4, 6

    (g) 89, 79, 32, 38, 46, 26, 43, 38, 32, 79

    Em cada caso, mostre o número de comparações e trocas que realizam na ordenação de sequências.

17. Faça um programa que leia n nomes inserindo-os em uma lista de forma ordenada utilizando a ideia do algoritmo insertion sort. No final, o programa deve mostrar todos os nomes ordenados alfabeticamente.

18. Crie um programa que dado uma string, coloque as letras dela em ordem crescente pelo algoritmo bubble sort.

19. Faça um programa que leia n nomes e ordene-os pelo tamanho utilizando o algoritmo selection sort

20. Crie um programa que dado uma string, coloque as letras dela em ordem decrescente usando o algoritmo quick sort.

21. Considere a seguinte estrutura:

```
struct pessoa{
int Matricula;
char Nome[30];
float Nota;
};
```

    Faça uma função que dado um array de tamanho N dessa estrutura, ordene o array pelo campo escolhido pelo usuário.