Algoritmos de Ordenação

1. Algoritmo de ordenação Bubble Sort

O Bubble Sort, ou Ordenação Bolha, é um algoritmo de ordenação de dados por troca, ou seja, ele compara e troca elementos adjacentes até que a lista esteja ordenada. O algoritmo é simples e comumente utilizado para a compreensão do processo de ordenação. Em termos computacionais, o Bubble Sort é ineficiente, pois implementa dois laços de repetição aninhados e possui complexidade de tempo quadrática – $O(n^2)$.

Exemplo de implementação:

```
#include <stdio.h>

// Função Bubble Sort
void bubbleSort(int array[], int t) {
  int aux;
  for (int i = 0; i < t-1; i++) {  // Percorre a lista t vezes (tamanho do array)
    for (int j = 0; j < t-1; j++) {  // Percorre a lista até o penúltimo elemento
      if (array[j] > array[j+1]) {  // Compara elementos adjacentes
        // Troca os elementos de posição caso estejam fora de ordem crescente
        aux = array[j];
        array[j] = array[j+1];
        array[j+1] = aux;
      }
    }
  }
}

int main() {
    int array[] = {67, 38, 21, 15, 29}; // Declara o vetor

    // Calcula o tamanho do vetor
    // sizeof(array) retorna a qtde de bytes alocados
    // sizeof(array[0] especifica quantos bytes um elemento ocupa (neste caso, um inteiro)
    int tamanho = sizeof(array) / sizeof(array[0]);

    printf("Vetor original: ");
    for (int i = 0; i < tamanho; i++) {
      printf("%d ", array[i]);
    }
    printf("\n");

    bubbleSort(array, tamanho);  // Chama a função de ordenação Bubble Sort
```

```
    printf("Vetor ordenado: ");
    for (int i = 0; i < tamanho; i++) {
      printf("%d ", array[i]);
    }
    printf("\n");

    return 0;
}
```

Exemplo de funcionamento:

I = 0

67	**38**	**21**	**15**	**29**	**67 > 38? Troca**
38	**67**	**21**	**15**	**29**	**67 > 21? Troca**
38	**21**	**67**	**15**	**29**	**67 > 15? Troca**
38	**21**	**15**	**67**	**29**	**67 > 29? Troca**
38	**21**	**15**	**29**	**67**	**Final da iteração**

I = 1

38	**21**	**15**	**29**	**67**	**38 > 21? Troca**
21	**38**	**15**	**29**	**67**	**38 > 15? Troca**
21	**15**	**38**	**29**	**67**	**38 > 29? Troca**
21	**15**	**29**	**38**	**67**	**38 > 67? Não Troca**
21	**15**	**29**	**38**	**67**	**Final da iteração**

I = 2

21	**15**	**29**	**38**	**67**	**21 > 15? Troca**
15	**21**	**29**	**38**	**67**	**21 > 29? Não Troca**
15	**21**	**29**	**38**	**67**	**29 > 38? Não Troca**
15	**21**	**29**	**38**	**67**	**38 > 67? Não Troca**
15	**21**	**29**	**38**	**67**	**Final da iteração**

I = 3

15	21	29	38	67	15 > 21? Não Troca

15	21	29	38	67	21 > 29? Não Troca

15	21	29	38	67	29 > 38? Não Troca

15	21	29	38	67	38 > 67? Não Troca

15	21	29	38	67	Final da iteração

2. Algoritmo de ordenação Selection Sort

O Selection Sort, ou Ordenação por Seleção, é um algoritmo de ordenação capaz de selecionar o menor (ou maior) elemento de um vetor e trocá-lo de posição até que o conjunto de valores esteja ordenado. Nele, cada número do vetor, a partir do primeiro, será eleito e comparado com o menor (ou maior, dependendo da ordenação desejada) número dentre aqueles que estão à direita do eleito. Nessas comparações procura-se um número menor que o eleito (quando a ordenação for crescente) ou um maior que o eleito (quando a ordenação for decrescente). Quando um número satisfaz as condições da ordenação desejada (menor ou maior que o eleito), ele será trocado de posição com o eleito. Dessa forma, todos os números à esquerda do eleito são ordenados. O Selection Sort é simples e de fácil compreensão, contudo ineficiente para listas com uma grande quantidade de valores ou listas que já estejam praticamente ordenadas.

Exemplo de implementação:

```
#include <stdio.h>


// Função Selection Sort (ordem crescente)
void selectionSort(int array[], int t) {
  int aux;
  for (int i = 0; i < t-1; i++) {  // Percorre a lista não ordenada
    int indiceMenor = i;  // Assume que o elemento atual é o menor
    for (int j = i+1; j < t; j++) {  // Procura o menor elemento na lista não ordenada
      if (array[j] < array[indiceMenor]) {
        indiceMenor = j;  // Atualiza o índice do menor elemento
      }
    }
    // Troca o elemento atual (array[i]) com o menor elemento encontrado (array[indiceMenor])
    aux = array[i];
    array[i] = array[indiceMenor];
    array[indiceMenor] = aux;
  }
}


// Função para impressão do vetor
void impressao(char msg[], int array[], int t) {
  printf("%s", msg);
```

```
  for (int i = 0; i < t; i++) {
    printf("%d ", array[i]);
  }
  printf("\n");
}

int main() {
    int array[] = {62, 35, 14, 32, 11};
    int tamanho = sizeof(array)/sizeof(array[0]);  // Calcula o tamanho do vetor

    impressao("Vetor original: ", array, tamanho);

    selectionSort(array, tamanho);  // Chama a função de ordenação Selection Sort

    impressao("Vetor ordenado: ", array, tamanho);

    return 0;
}
```

Exemplo de funcionamento:

I = 0

62	**35**	**14**	**32**	**11**

62 eleito; 11 menor; Troca

I = 1

11	**35**	**14**	**32**	**62**

35 eleito; 14 menor; Troca

I = 2

11	**14**	**35**	**32**	**62**

35 eleito; 32 menor; Troca

I = 3

11	**14**	**32**	**35**	**62**

35 eleito; 35 menor; Não Troca

3. Exercícios

3.1. O algoritmo Bubble Sort apresentado não interrompe as comparações mesmo quando o conjunto de números já está ordenado. Realize ajustes no código para tornar o algoritmo mais eficiente. Além disso, imprima a quantidade de iterações que cada uma das versões utiliza para ordenar o conjunto de dados fornecido.

3.2. Implemente um algoritmo para ordenação (ordem decrescente) da lista a seguir:

41	**-8**	**16**	**2**	**52**	**39**	**19**

Imprima o vetor resultante de cada passo do processo de ordenação e a quantidade de iterações necessárias para realizá-la.

3.3. Elabore um algoritmo que implemente o Insertion Sort. Inclua uma breve descrição do algoritmo e implemente um exemplo de ordenação.

3.4. Considerando os algoritmos de ordenação Merge Sort e Quick Sort, realize as seguintes atividades:

a) Elabore um breve texto descrevendo o algoritmo, seu propósito de utilização, principais vantagens e desvantagens.
b) Crie um exemplo no qual o algoritmo é utilizado.

3.5. Elabore um algoritmo que leia um conjunto de caracteres, realize a ordenação em ordem alfabética e exiba o resultado.