O Poder Oculto dos Algoritmos

Domine a Ordenação em C#

Aprenda quais são os principais algoritmos de ordenação e construa aplicações mais performáticas

Jefferson Mesquita

PRINCIPAIS ALGORITMOS DE ORDENAÇÃO

Uma Jornada pelos Algoritmos Essenciais

Se você é um desenvolvedor em C#, entender os algoritmos de ordenação é crucial para otimizar o desempenho dos seus programas. Neste ebook, vamos explorar os principais algoritmos de ordenação de uma maneira simples e prática, fornecendo exemplos de código.

01

Bubble Sort

O Início da Jornada

Como funciona?

O **Bubble Sort** é um dos algoritmos mais simples, mas não menos importante. Ele percorre a lista, comparando elementos adjacentes e os trocando se estiverem na ordem errada.

Troca de posição porque 7 é maior que 2

Não é necessário trocar a posição porque 7 é menor que 9

Troca de posição porque 9 é maior que 6

Troca de posição porque 9 é maior que 4

Exemplo de código

```
public void BubbleSort(int[] array)
{
    int n = array.Length;
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (array[j] > array[j + 1])
            {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}
```

- A função BubbleSort recebe como entrada um array de inteiros.
- Um loop externo percorre o array da esquerda para a direita, exceto pelo último elemento, já que ao final de cada iteração deste loop, o maior elemento da lista é colocado na sua posição correta.
- Dentro do loop externo, há um loop interno que percorre a parte não ordenada do array. Este loop compara cada elemento com o próximo elemento na lista.
- Se o elemento atual for maior que o próximo elemento, os dois elementos são trocados de posição.
- Esse processo continua até que todos os elementos estejam ordenados, o que é feito após repetir o loop externo até o penúltimo elemento, garantindo que o último elemento já esteja na posição correta.

02

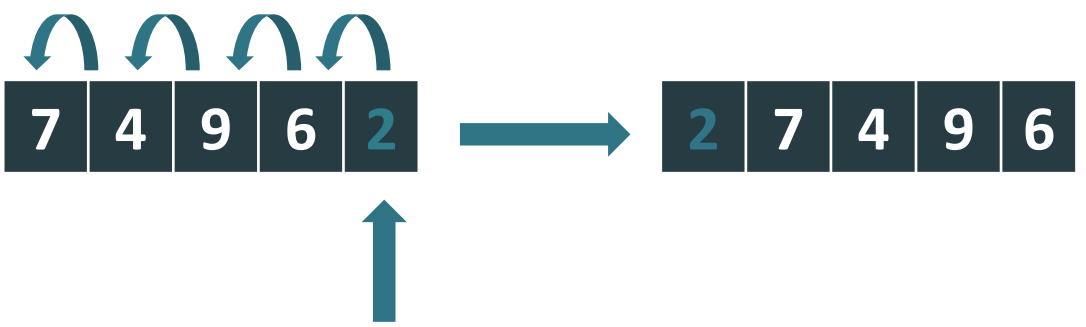
Insertion Sort

Construindo a Ordem

Como funciona?

O Insertion Sort é eficiente para listas pequenas. Ele constrói a lista ordenada um item de cada vez, movendo cada elemento para sua posição correta.

Vejamos como implementá-lo:



Verifica se o valor a esquerda é maior e movimenta uma posição. Este processo se repete até que o número da esquerda seja menor ou não exista posição a esquerda.

Exemplo de código

```
public void InsertionSort(int[] array)
{
    int n = array.Length;
    for (int i = 1; i < n; ++i)
    {
        int key = array[i];
        int j = i - 1;

        while (j >= 0 && array[j] > key)
        {
            array[j + 1] = array[j];
            j = j - 1;
        }
        array[j + 1] = key;
    }
}
```

- A função InsertionSort recebe como entrada um array de inteiros.
- Um loop externo percorre o array, começando do segundo elemento (índice 1) até o último.
- Para cada elemento do loop externo, o algoritmo salva o valor do elemento atual em uma variável chamada key.
- Em seguida, um loop interno é iniciado para comparar o elemento atual com os elementos à sua esquerda (elementos já ordenados).
- Enquanto o elemento à esquerda for maior que o valor de key e não atingirmos o início do array, os elementos são deslocados uma posição para a direita.
- Após encontrar a posição correta para o elemento key, ele é inserido nessa posição no array ordenado.
- Este processo é repetido para cada elemento não ordenado, resultando em um array totalmente ordenado ao final da execução do algoritmo.

03

Merge Sort

Dividir para Conquistar

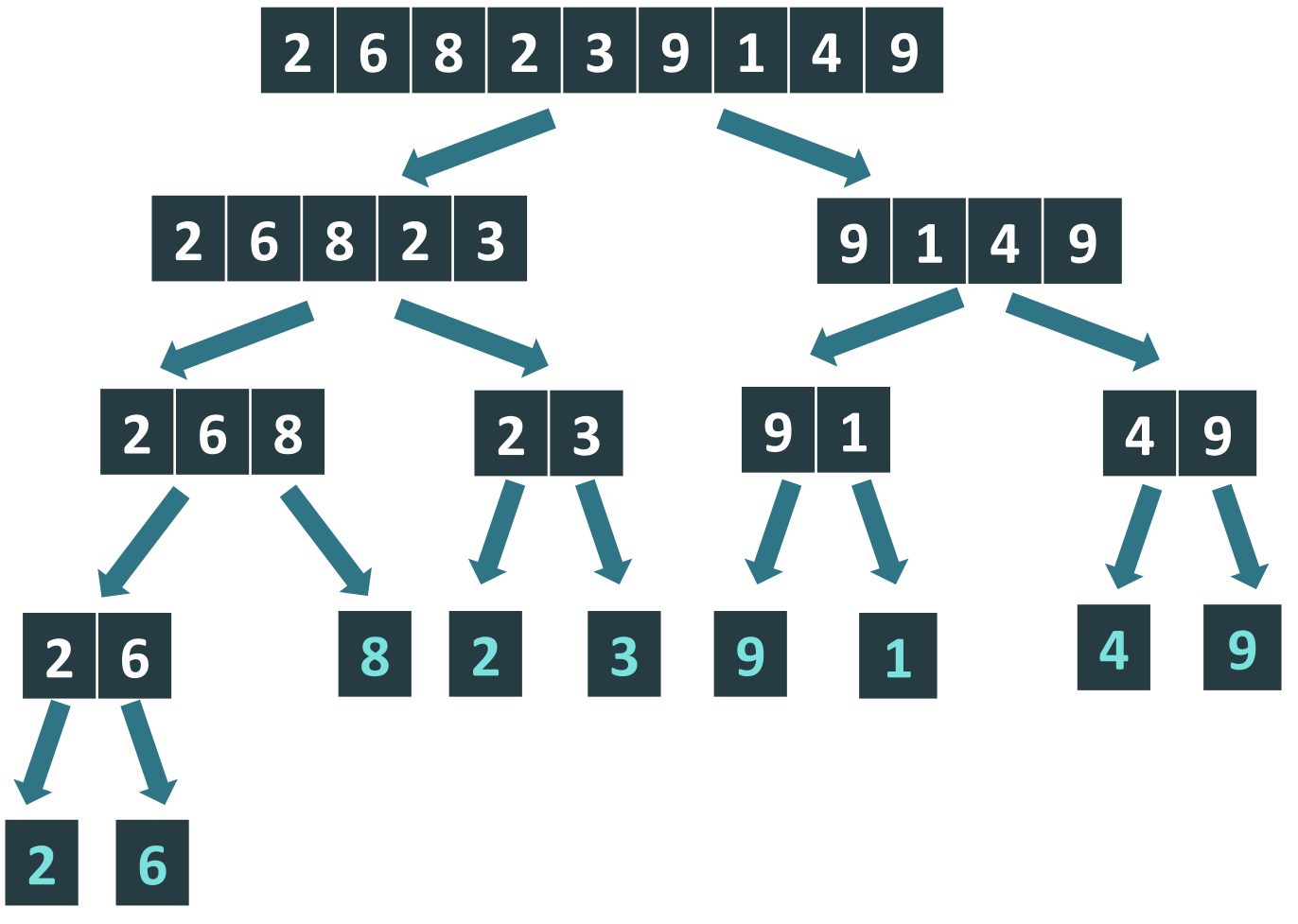
Como funciona?

O Merge Sort é um algoritmo de ordenação que utiliza a técnica de "dividir para conquistar". Aqui está uma explicação simples e objetiva:

- **Dividir**: O array é dividido continuamente pela metade até que cada subarray contenha apenas um elemento. Isso é feito recursivamente até que não seja mais possível dividir.

- **Conquistar**: Em seguida, os subarrays são mesclados em pares, de forma ordenada. Durante a mesclagem, os elementos são comparados e colocados em ordem crescente (ou decrescente).

- **Combinar**: Essa etapa envolve a combinação dos subarrays mesclados para formar um único array ordenado. Isso é feito até que todos os subarrays sejam mesclados em um único array ordenado, resultando na lista final ordenada.



Exemplo de código

```
public void MergeSort(int[] array, int left, int right)
{
    if (left < right)
    {
        int middle = (left + right) / 2;
        MergeSort(array, left, middle);
        MergeSort(array, middle + 1, right);
        Merge(array, left, middle, right);
    }
}

public void Merge(int[] array, int left, int middle, int right)
{
    // Implementação do merge...
}
```

Este código implementa o algoritmo de ordenação Merge Sort em C#. Vou explicar de maneira clara e objetiva o que cada parte do código faz:

MergeSort(int[] array, int left, int right):

- Esta é a função principal do algoritmo Merge Sort.
- Ela recebe como entrada um array de inteiros (array) e dois índices, left e right, que indicam os limites esquerdo e direito do subarray a ser ordenado.
- A função começa verificando se o índice left é menor que o índice right, garantindo que ainda há elementos para serem ordenados no subarray.
- Se essa condição for verdadeira, a função calcula o índice do meio (middle) do subarray, dividindo a soma de left e right por 2.
- Em seguida, a função chama recursivamente MergeSort duas vezes: uma para ordenar a metade esquerda do subarray (left até middle) e outra para ordenar a metade direita do subarray (middle + 1 até right).
- Após ordenar as duas metades, a função chama a função Merge para mesclar as duas metades ordenadas em um único subarray ordenado.

AGRADECIMENTOS

OBRIGADO POR LER ATÉ AQUI

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github

.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.

https://github.com/jsmesquita/prompts-recipe-to-create-a-ebook

Jefferson Mesquita

GitHub | LinkedIn