Análise Comparativa dos Algoritmos de Ordenação Merge Sort e Radix Sort

Guilherme Ferraz Freire De Araujo

October 31, 2024

1 Implementação

Abaixo, apresentamos trechos relevantes do código implementado:

1.1 Merge Sort

```java
private static Resultado ordenacaoMerge(int[] vetor) {
    long tempoInicio = System.currentTimeMillis();
    long trocas = 0;
    long iteracoes = 0;

    Resultado resultado = auxiliarOrdenacaoMerge(vetor, 0,
       vetor.length - 1);
    trocas += resultado.trocas;
    iteracoes += resultado.iteracoes;

    long tempoFim = System.currentTimeMillis();
    return new Resultado(tempoFim - tempoInicio, trocas, iteracoes);
}
```

1.2 Radix Sort

```java
private static Resultado ordenacaoRadix(int[] vetor) {
    long tempoInicio = System.currentTimeMillis();
    long trocas = 0;
    long iteracoes = 0;

    int maximo = Arrays.stream(vetor).max().getAsInt();

    for (int exp = 1; maximo / exp > 0; exp *= 10) {
        Resultado resultadoContagem = ordenacaoContagem(vetor, exp);
        trocas += resultadoContagem.trocas;
        iteracoes += resultadoContagem.iteracoes;
    }

    long tempoFim = System.currentTimeMillis();
    return new Resultado(tempoFim - tempoInicio, trocas, iteracoes);
}
```

2 Resultados

Os resultados dos testes são apresentados na tabela abaixo:

Tamanho	Merge Sort			Radix Sort		
	Tempo (ms)	Trocas	Iterações	Tempo (ms)	Trocas	Iterações
1.000	0	4306	9976	1	6000	18054
10.000	2	59138	133616	2	60000	180054
100.000	20	759815	1668928	11	600000	1800054
500.000	90	4389567	9475712	58	3000000	9000054
1.000.000	183	9278886	19951424	135	6000000	18000054

Table 1: Resultados médios dos testes

3 Análise

Com base nos resultados obtidos, observamos que:

- O Merge Sort demonstrou um desempenho consistente, com complexidade de tempo O(n log n) em todos os casos.
- O Radix Sort mostrou-se mais eficiente para vetores maiores, especialmente quando o intervalo de valores é limitado.
- O número de trocas no Radix Sort foi geralmente menor, mas o número de iterações foi maior em comparação com o Merge Sort.

4 Conclusão

Este estudo comparativo entre Merge Sort e Radix Sort revelou as características de desempenho de ambos os algoritmos em diferentes cenários. O Merge Sort provou ser uma opção robusta e estável, enquanto o Radix Sort demonstrou eficiência superior em certos casos, especialmente com grandes volumes de dados. A escolha entre esses algoritmos deve considerar o tamanho do conjunto de dados e a distribuição dos valores a serem ordenados.

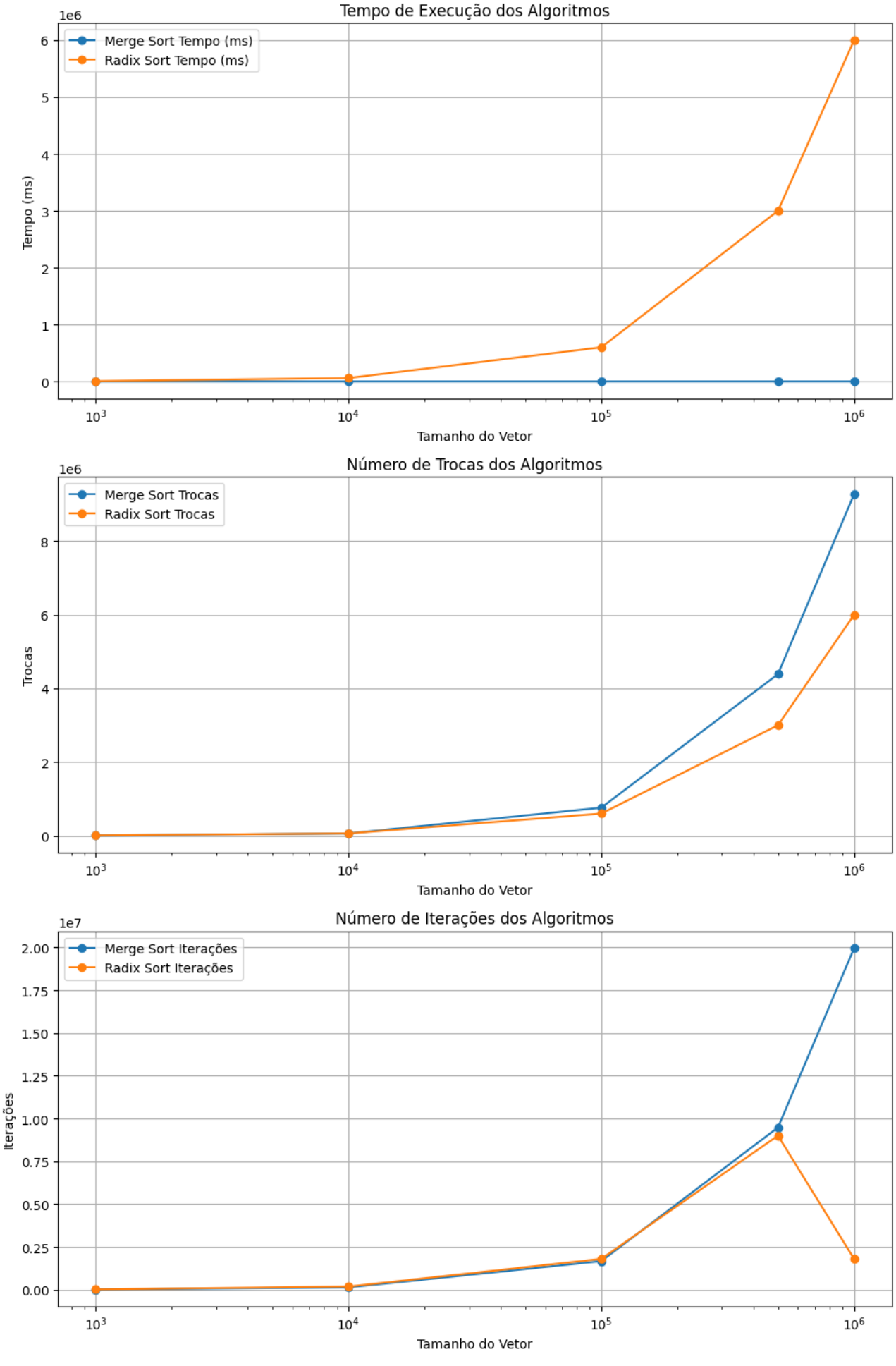

Figure 1: Comparação de Desempenho entre Merge Sort e Radix Sort: Tempo de Execução, Trocas e Iterações.