Análise empírica da eficácia de algoritmos de ordenação: uma comparação do tempo de execução em diferentes cenários e conjuntos de dados

Víctor Macêdo Carvalho[1]
Departamento de Sistemas de Informação
Universidade Federal do Piauí
victmacc@ufpi.edu.br

Luís Eduardo Silva Brito[2]
Departamento de Sistemas de Informação
Universidade Federal do Piauí
duardos36@gmail.com

***Abstract*— Contexto: algoritmos de ordenação são amplamente utilizados em várias áreas da ciência da computação, desde a organização de dados em bancos de dados até a otimização de rotas em sistemas de navegação. Existem vários tipos de algoritmos de ordenação, cada um com suas próprias características e desempenho. A avaliação comparativa desses algoritmos é fundamental para determinar qual é o mais eficaz em diferentes cenários.**
Problema: O problema consiste em comparar o desempenho de diferentes algoritmos de ordenação e determinar qual é o mais eficaz em termos de tempo de execução. Esse é um problema importante, uma vez que o tempo de execução é um fator crítico na eficiência de algoritmos de ordenação em cenários com grandes quantidades de dados.
Resultados: Resultado: O resultado da avaliação comparativa de algoritmos de ordenação é a identificação do algoritmo mais eficaz em termos de tempo de execução. Isso permite que desenvolvedores e cientistas da computação escolham o melhor algoritmo para resolver um determinado problema em diferentes cenários, aumentando assim a eficiência e a escalabilidade de suas soluções. É importante notar que a escolha do algoritmo mais eficaz depende do tamanho do conjunto de dados, do tipo de dados e do objetivo da ordenação.

keywords - : algoritmos de ordenação; desempenho; tempo de execução; avaliação comparativa.

I. INTRODUÇÃO

A ordenação de dados é uma tarefa crucial na computação, presente em diversas aplicações, desde a classificação de informações em uma biblioteca até a organização de grandes conjuntos de dados em algoritmos de aprendizado de máquina [1]. A escolha do algoritmo de ordenação mais eficiente é fundamental para o desempenho e escalabilidade dessas soluções. Neste contexto, a avaliação comparativa do tempo de execução dos algoritmos de ordenação é uma questão importante e desafiadora.

O problema a ser abordado é a comparação de diferentes algoritmos de ordenação para determinar qual deles é mais eficiente em termos de tempo de execução. A eficiência em tempo de execução é crítica em cenários com excesso de dados, onde o tempo de processamento pode ser um gargalo significativo no desempenho do sistema. Para determinar qual algoritmo de ordenação é o mais eficiente, é necessário realizar uma análise cuidadosa e sistemática, considerando diferentes tamanhos de conjuntos de dados, tipos de dados e características dos algoritmos.

Para avaliar o tempo de execução dos algoritmos de ordenação, utilizamos um conjunto de dados de diferentes tamanhos e tipos, incluindo listas aleatórias. Foram avaliados os algoritmos de ordenação mais comuns, como Bubble Sort e Insertion Sort. A avaliação foi feita em uma máquina com configuração padrão. Os resultados foram analisados e comparados, destacando os algoritmos mais eficientes para diferentes tamanhos e tipos de dados.

O artigo está dividido da seguinte forma: A Seção II descreve a metodologia aplicada nos experimentos, bem como os equipamentos utilizados. A Seção III apresenta os resultados obtidos. Por fim, a Seção IV traz a conclusão sobre este trabalho.

II. METODOLOGIA

Esta seção apresenta a metodologia utilizada nesse trabalho. A seção foi dividida em uma abordagem experiemntal com tópicos, bem como três categorias, a categoria A que mostra os algoritmos utilizados para ordenar os números, a categoria B que apresenta os testes utilizados para calcular o desempenho dos algoritmos de ordenação, e a categoria C que apresenta o teste T. A abordagem experimental utilizada neste trabalho é amplamente empregada em estudos de algoritmos e programação, permitindo comparar diferentes soluções para um mesmo problema e avaliar o desempenho de cada uma delas.

A metodologia empregada aqui segue uma série de passos bem definidos, que são descritos a seguir:

1. Seleção dos algoritmos: nesta etapa, são escolhidos os algoritmos que serão comparados. A escolha deve levar em consideração critérios como simplicidade, eficiência e facilidade de implementação.

2. Geração de dados: para realizar os testes de desempenho, é necessário gerar conjuntos de dados que serão utilizados como entrada para cada algoritmo. Esses conjuntos devem ser criados de forma aleatória, ou seguindo algum padrão específico, dependendo do tipo de teste que será realizado.

3. Implementação dos algoritmos: cada algoritmo selecionado deve ser implementado em uma linguagem de

programação específica, utilizando as melhores práticas de programação e estrutura de dados.

4. Realização dos testes: os testes de desempenho devem ser realizados com os conjuntos de dados gerados previamente, medindo-se métricas como tempo de execução, uso de memória, entre outras.

5. Análise dos resultados: os resultados obtidos nos testes devem ser analisados para verificar qual algoritmo obteve melhor desempenho em cada métrica medida. É importante observar se os resultados são consistentes e se há diferenças significativas entre os algoritmos comparados.

6. Validação estatística: para confirmar a significância estatística das diferenças observadas entre os algoritmos, é necessário aplicar um teste estatístico como o teste T de Student. Esse teste permite avaliar se as diferenças observadas são reais ou se podem ter ocorrido por acaso.

7. Ao final da análise dos resultados, é possível tirar conclusões sobre qual algoritmo é o mais eficiente para a tarefa em questão. É importante destacar que as conclusões obtidas devem levar em consideração não apenas os resultados obtidos nos testes, mas também a facilidade de implementação e a simplicidade de cada algoritmo.

Dessa forma, a abordagem experimental empregada neste trabalho permite comparar diferentes algoritmos de ordenação, avaliando o desempenho de cada um deles em relação a diversas métricas, com base em um conjunto de dados gerados previamente. A aplicação de um teste estatístico permite validar os resultados obtidos, confirmando a significância estatística das diferenças observadas entre os algoritmos.

A. Algoritmos utilizados

Nesse tópico, é apresentado os algoritmos de ordenação que foram utilizados nos testes:

1) Algoritmo de ordenação Bubble Sort: O primeiro algoritmo é conhecido como Bubble Sort, que funciona percorrendo todo o vetor várias vezes e trocando os elementos adjacentes se estiverem na ordem errada. Esse processo é repetido até que o vetor esteja completamente ordenado. O Bubble Sort é um dos algoritmos de ordenação mais simples, mas com uma complexidade de tempo $O(n^2)$.

```
void ordenarComAlgoritmoA(int array[], int tamanho) {
    int i, j, aux;

    for (i = 0; i < tamanho - 1; i++) {
        for (j = 0; j < tamanho - i - 1; j++) {
            if (array[j] > array[j+1]) {
                aux = array[j];
                array[j] = array[j+1];

                array[j+1] = aux;
            }
        }
    }
}
```

Fig. 1. Algoritmo Bubble Sort.

2) Algoritmo de ordenação Insertion Sort: O segundo algoritmo utilizado foi o Insertion Sort, que percorre o vetor da esquerda para a direita e à medida que avança, insere cada elemento na posição correta em relação aos elementos anteriores. Esse método é considerado estável, o que significa que a ordem relativa dos elementos iguais não é alterada durante o processo de ordenação. O Insertion Sort tem uma complexidade de tempo $O(n^2)$.

```
void ordenarComInsertionSort(int array[], int tamanho) {
    int i, j, atual;

    for (i = 1; i < tamanho; i++) {
        atual = array[i];
        j = i - 1;

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

        array[j+1] = atual;
    }
}
```

Fig. 2. Algoritmo Insertion Sort.

B. Testes de desempenho dos algoritmos:

Nessa categoria são apresentados os testes feitos para saber o tempo total de execução do algoritmo conforme a quantidade de números a serem ordenados, esses testes são desenvolvidos em linguagem C, e a partir deles poderemos verificar qual dos algoritmos é o mais eficiente em questão de tempo de execução.

C. Teste T

Essa categoria apresenta o teste T, o qual é um teste estatístico utilizado para comparar as médias, por meio da coleta de dados de duas amostras. Esse teste foi utilizado no trabalho para comparar a média de execução de algoritmos de ordenação, o resultado obtido foi mostrado por meio de gráficos Boxplot.

III. RESULTADOS

Nesta seção será apresentado o resultado da comparação das amostras por meio de gráficos boxplot, e também será feita a avaliação do método estatístico então mencionado neste artigo, o teste T. considerando o intervalo de noventa e cinco por cento de confiança e cinco por cento de margem de erro.

A tabela 1 é referente aos números escolhidos para comporem a amostra aos algoritmos de ordenação utilizados, e também os resultados dos testes referentes ao desempenho dos algoritmos de ordenação, já as figuras 1, 2 e 3 trazem os gráficos de resultado dessas analises destes algoritmos. Como podemos observar, o algoritmo B teve uma melhor desempenho de tempo de execução do que o algoritmo A. Dessa forma, resultando em um melhor desempenho. .

TABLE I

COMPARAÇÃO DOS RESULTADOS

ALGORITMO	DADOS DA AMOSTRA	RESULTADO
Insertion Sort	500000 números	Inferior a 0,005
Bubble Sort	500000 números	Inferior a 0,005

A figura 3 traz um gráfico de possibilidades do algoritmo Bubble Sort, no Notebook Inspiron 15 3000, com a análise feita podemos notar que a quantidade de tempo de execução foi baixa e constante.

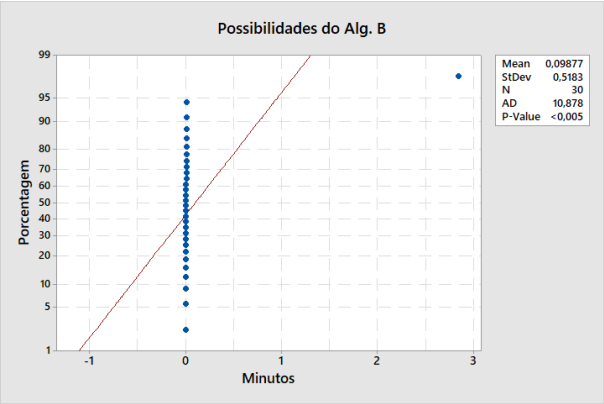

Fig. 3. Algoitmo Bubble Sort.

A figura 4 traz um gráfico de possibilidades do algoritmo Insert Sort, no Notebook Inspiron 15 3000, com a análise feita podemos notar que a quantidade de tempo de execução foi baixa e constante.

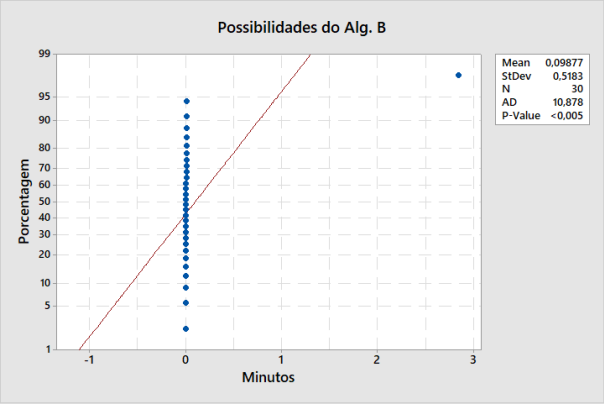

Fig. 4. Algoritmo Insert Sort

IV. CONCLUSÃO

Tendo em vista os resultados obtidos, foi possível comprovar o que era esperado a partir das condições analisadas: o algoritmo Insertion Sort apresenta menos tempo de execução para a ordenação. Portanto, dos dois algoritmos comparados, o algoritmo Insertion Sort seria a escolha mais relevante para atender as necessidades de usuarios exigentes. Como sugestão para estender e evoluir essa avaliação comparativa, seria interessante aumentar a quantidade de números desordenados. Além disso, com o intuito de testar diversos cenaáios de uso, é importante realizar as medidas acrescentando diversificações dos tipos de ferramntas computadorizadas para a realização do experimento.

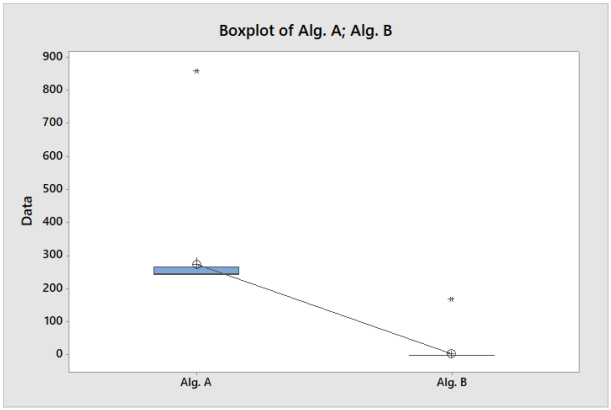

Fig. 5. Resultado expresso em Box plot.

REFERENCES

[1] Tayná Luana Silva da Costa. O uso de computação desplugada para apoiar a aprendizagem de algoritmos de ordenação e tabela hash. 2017.