Universidade Federal do Piauí

Sistemas de Informação

Projeto e Análise de Algoritmos

5º Período

Relatório referente ao primeiro trabalho

Alunos: João Marcos Rufino e Raildom da Rocha

Professor: José Denes Lima Araújo

Setembro de 2025

Conteúdo

1 Introdução

A ordenação de dados é uma operação fundamental na ciência da computação, essencial para a eficiência no processamento e na recuperação de informações em uma vasta gama de aplicações, desde sistemas operacionais a bancos de dados. A existência de múltiplos algoritmos de ordenação, cada um com suas particularidades, torna o estudo comparativo entre eles uma prática de grande valor.

Este trabalho tem como foco principal aprofundar o conhecimento sobre algoritmos de ordenação por meio da implementação e análise de dois métodos específicos: o Insertion Sort e o Binary Insertion Sort. O Insertion Sort é um algoritmo conhecido por sua simplicidade, que ordena os elementos um a um. O Binary Insertion Sort, por sua vez, é uma otimização que utiliza a busca binária para acelerar a localização da posição correta de cada elemento, visando reduzir o número de comparações.

Para realizar uma análise de desempenho completa, os dois algoritmos serão submetidos a testes práticos. Serão utilizados seis conjuntos de dados com volumes distintos (1.250, 2.500, 5.000, 10.000, 20.000 e 40.000 elementos) e em três cenários diferentes: vetores ordenados em ordem crescente, em ordem decrescente e com números aleatórios. Além disso, cada conjunto de dados foi executado 30 vezes. O tempo de execução de cada algoritmo será medido e os resultados serão compilados em gráficos comparativos.

O objetivo final é, através dessa análise empírica, visualizar e compreender as diferenças de eficiência entre o Insertion Sort e o Binary Insertion Sort, identificando como o tamanho e a ordem inicial dos dados influenciam seu desempenho. Isso permitirá uma conclusão informada sobre qual dos dois algoritmos é mais adequado para diferentes tipos de problemas práticos.

2 Definição

Nesta seção serão apresentadas as definições formais dos algoritmos Insertion Sort e Binary Insertion Sort.

2.1 Insertion Sort

O Insertion Sort (Ordenação por Inserção) é um algoritmo de ordenação simples que constrói a sequência final ordenada um elemento por vez. É considerado um algoritmo de ordenação in-place (não requer espaço adicional significativo além do array de entrada) e estável (mantém a ordem relativa de elementos com chaves iguais). A lógica fundamental do Insertion Sort assemelha-se à forma como uma pessoa organiza um baralho de cartas na mão. A cada nova carta retirada do monte, ela é inserida na posição correta entre as cartas já ordenadas na mão.

2.2 Binary Insertion Sort

O Binary Insertion Sort (Ordenação por Inserção Binária) é uma variação do algoritmo Insertion Sort tradicional que busca otimizar o processo de encontrar a posição correta para cada elemento na sub-array já ordenada. Enquanto o Insertion Sort padrão utiliza uma busca linear para encontrar essa posição (comparando o elemento atual com cada elemento da sub-array ordenada, um por um), o Binary Insertion Sort emprega a busca binária.

3 Aspectos Históricos

Nesta seção serão apresentados os principais acontecimentos históricos relacionados ao desenvolvimento e à evolução dos algoritmos de ordenação, com ênfase no Insertion Sort e no Binary Insertion Sort.

3.1 Insertion Sort

O Insertion Sort não tem uma data ou autor específicos para sua criação porque ele surgiu de forma muito natural, a partir da própria necessidade de organizar elementos de maneira simples e intuitiva. A lógica do algoritmo reflete o modo tradicional que as pessoas usam há séculos para ordenar cartas, papéis ou objetos: você mantém os itens já organizados e, ao pegar um novo, insere-o na posição correta, deslocando os que forem maiores. Quando os primeiros computadores começaram a ser utilizados, nas décadas de 1940 e 1950, essa técnica manual acabou sendo adaptada para procedimentos computacionais justamente porque era fácil de implementar e não exigia muitos recursos de processamento ou memória, que eram extremamente limitados na época. Assim, o Insertion Sort foi sendo formalizado e documentado em livros e materiais de ciência da computação como um dos métodos mais básicos e fundamentais de ordenação, tanto para uso prático em pequenos conjuntos de dados quanto para fins didáticos, ajudando no entendimento de conceitos mais complexos de algoritmos.

3.2 Binary Insertion Sort

O Binary Insertion Sort surgiu como uma evolução natural do Insertion Sort tradicional, quando pesquisadores e programadores começaram a buscar formas de reduzir o número de comparações realizadas pelo algoritmo. A ideia central permaneceu a mesma, inserir cada elemento na posição correta na parte já ordenada, mas a diferença foi o uso da busca binária para encontrar esse ponto de inserção, em vez da busca linear usada no método original.

Não existe um registro histórico preciso com nome de autor ou ano exato para o surgimento do Binary Insertion Sort, pois ele aparece como uma variação prática do Insertion Sort em materiais acadêmicos e livros de algoritmos já nas décadas seguintes ao desenvolvimento dos primeiros métodos de ordenação.

Assim como o Insertion Sort, ele também passou a ser utilizado principalmente em contextos educacionais e em implementações híbridas, onde a simplicidade e a eficiência em listas pequenas eram desejáveis.

4 Comparação de Funcionamento dos Algoritmos

O Insertion Sort e o Binary Insertion Sort são algoritmos de ordenação por inserção que constroem uma lista ordenada elemento por elemento, ambos sendo in-place e estáveis. A principal diferença reside na forma como encontram a posição de inserção: o Insertion Sort usa busca linear (O(n) comparações por inserção), enquanto o Binary Insertion Sort utiliza busca binária (O(log n) comparações por inserção), reduzindo o número total de comparações. Contudo, ambos mantêm a complexidade de tempo de $O(n^2)$ no pior e caso médio devido ao custo das movimentações de elementos, que não é otimizado pela busca binária. O Binary Insertion Sort é vantajoso quando as comparações são mais custosas que as movimentações, mas para grandes volumes de dados, ambos são superados por algoritmos com complexidade O(n log n).

5 Testes e Resultados

Nesta seção será apresentado como foram realizados os testes mencionados na introdução e os resultados obtidos a partir deles.

5.1 Hardware Utilizado

Todos os testes foram realizados utilizando o seguinte hardware:

Tabela 1: Especificações de Hardware

Componente	**Especificação**
Marca/Modelo	Samsung Galaxy Book 2
Processador	Intel Core i5-1235U (12ª geração)
Núcleos / Threads	10 núcleos (2 P-cores + 8 E-cores) / 12 threads
Memória RAM	16 GB LPDDR4x 3200 MHz
Armazenamento	256 GB SSD NVMe
Sistema Operacional	Windows 11 Home

5.2 Tempos de Execução

Os dois gráficos a seguir apresentam o desempenho temporal de cada algoritmo de ordenação em relação a cada conjunto de dados.

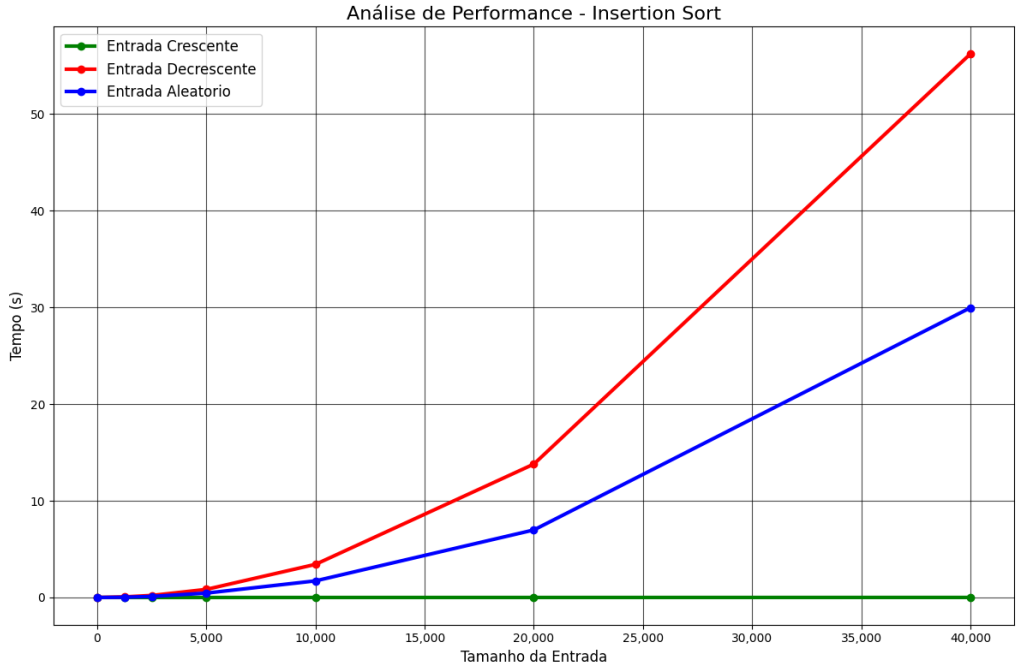

Figura 1: Tempo Insertion Sort

A figura 1 demonstra o tempo de execução do Insertion Sort para diferentes tamanhos de entrada, considerando três cenários: crescente, decrescente e aleatório. O caso crescente representa o melhor caso, pois o algoritmo praticamente não realiza trocas ele apenas verifica que os elementos já estão no lugar certo. O caso decrescente é o pior caso, pois cada novo elemento precisa ser comparado e movido para o início do vetor, gerando muito mais operações. A entrada aleatória fica entre os dois casos, pois nem sempre será necessário mover todos os elementos, mas ainda assim haverá várias comparações e inserções.

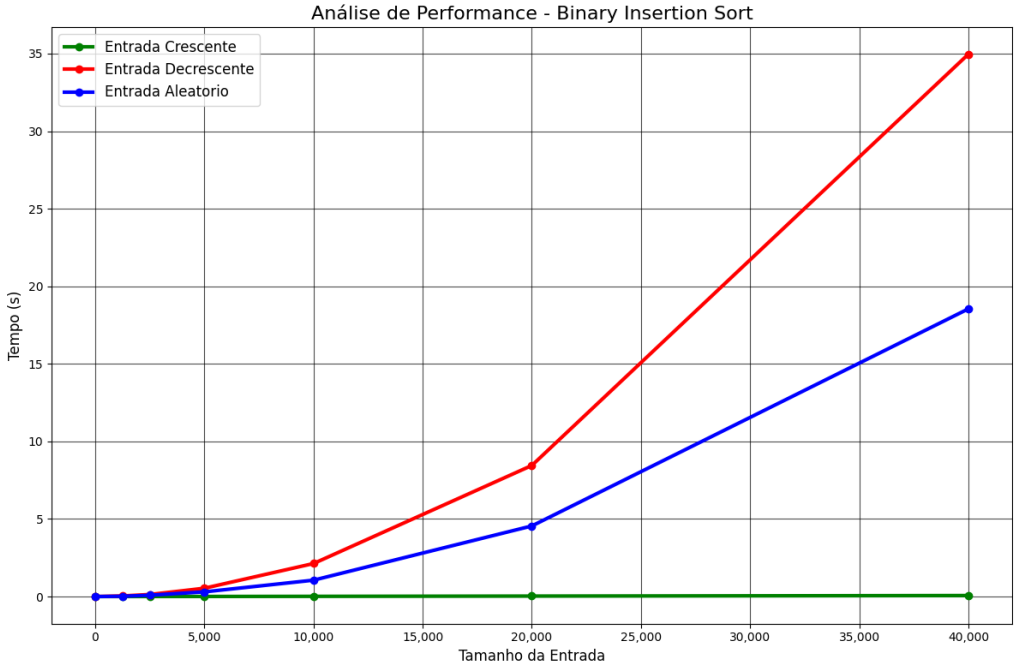

Figura 2: Tempo Binary Insertion Sort

A figura 2 segue a mesma lógica da primeira, mas para o Binary Insertion Sort. Podemos ver que o tempo de execução no pior caso é menor que no Insertion Sort tradicional, pois a busca binária reduz significativamente o tempo para encontrar a posição correta. Para entradas aleatórias, o desempenho também melhora em relação ao Insertion Sort simples, mas como o custo de mover os elementos permanece, o ganho não é tão expressivo. O melhor caso continua com tempo quase nulo, já que não há deslocamentos a serem feitos.

Como se pode observar, ambas as linhas apresentam comportamento de crescimento semelhante, entretanto a velocidade desse crescimento difere devido às escalas dos gráficos. Enquanto o Insertion Sort tradicional atinge valores próximos a 60 segundos de execução no pior caso, o Binary Insertion Sort permanece na faixa de 35 segundos. Esse resultado evidencia um ganho significativo de desempenho no tempo de ordenação, o qual pode ser atribuído ao processo de busca de posições mais eficiente implementado no Binary Insertion Sort.

5.3 Quantidades de Operações

O gráfico a seguir ilustra a tendência de crescimento do número de operações à medida que o tamanho da entrada a ser ordenada aumenta.

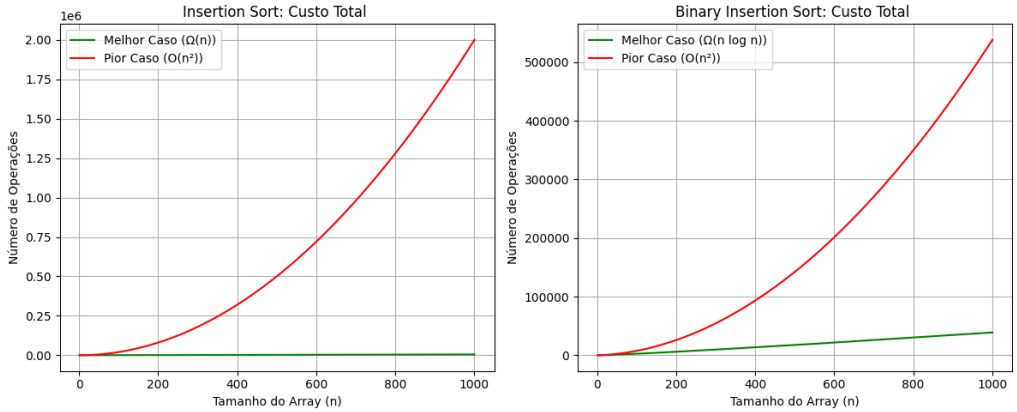

Figura 3: Quantidade de Operações

Os gráficos presentes na figura 3 mostram a quantidade de operações executadas conforme o tamanho do vetor aumenta.

- Insertion Sort: No melhor caso $\Omega(n)$, quando os dados já estão ordenados, o algoritmo só faz uma varredura simples sem muitos deslocamentos, resultando em comportamento linear. Já no pior caso $O(n^2)$, cada inserção exige mover quase todos os elementos, fazendo com que o custo cresça quadraticamente à medida que o tamanho da entrada aumenta.
- Binary Insertion Sort: Aqui a busca pela posição correta de inserção é feita usando busca binária, o que reduz as comparações para O(log n) por inserção, tornando o melhor caso $\Omega(n)$ log n). No entanto, os deslocamentos ainda permanecem, por isso o pior caso continua sendo $O(n^2)$. Isso significa que o algoritmo melhora a etapa de busca, mas não resolve o custo de movimentação dos elementos.

Observa-se que a tendência de crescimento mantém-se semelhante para ambos os algoritmos, porém com escalas distintas. Nesse experimento, o Binary Insertion Sort alcança aproximadamente 500.000 operações, enquanto, para a mesma entrada, o Insertion Sort tradicional ultrapassa em muito essa marca, chegando a mais de 2.000.000 de operações para realizar a ordenação. Essa diferença deve-se ao uso da busca binária no Binary Insertion Sort, que torna o processo de localização da posição correta de inserção significativamente mais eficiente, reduzindo drasticamente o número de comparações necessárias e, consequentemente, melhorando o desempenho e a velocidade de execução do algoritmo.

6 Conclusão

Os resultados indicam que a escolha entre Insertion Sort e Binary Insertion Sort depende do tipo de entrada. O Insertion Sort é preferível para listas já ordenadas (crescentes), especialmente em tamanhos menores, onde sua simplicidade resulta em execução mais rápida. Por outro lado, o Binary Insertion Sort é mais eficiente para listas decrescentes ou aleatórias, onde a busca binária compensa seu custo computacional ao reduzir comparações, tornando-o mais vantajoso em cenários desordenados ou maiores. Portanto, a decisão sobre qual algoritmo usar deve considerar o padrão esperado dos dados e o tamanho da entrada, com o Insertion Sort sendo ideal para casos ordenados e o Binary Insertion Sort para situações mais complexas ou desordenadas.

Referências

CORMEN, Thomas H. *et al.* **Algoritmos: teoria e prática**. 3. ed. Rio de Janeiro: Elsevier, 2012.

SIMIC, Milos. **Binary Insertion Sort**. Acesso em: 16 set. 2025. Baeldung: Computer Science. 2024. Disponível em: https://www.baeldung.com/cs/binary-insertion-sort.

SZWARCFITER, Jayme Luiz; LOPES, Liliane Maria da Silva. **Estruturas de dados e seus algoritmos**. 2. ed. Rio de Janeiro: LTC, 2010.