Universidade Federal de Viçosa
Campus **Rio Paranaíba**

Filipe Brener - 5952

ANÁLISE DO ALGORITMO INSERTION SORT

Rio Paranaíba - MG
2022

Universidade Federal de Viçosa
Campus **Rio Paranaíba**

Filipe Brener Ferreira Santos - 5952

ANÁLISE DO ALGORITMO INSERTION SORT

Trabalho apresentado para obtenção de créditos na disciplina SIN 213 - Projeto de Algoritmo da Universidade Federal de Viçosa - Campus de Rio Paranaíba, ministrada pelo Professor Pedro Moisés de Souza.

Rio Paranaíba - MG
2022

RESUMO

O acúmulo de dados tem se tornado cada vez mais um problema urgente a ser resolvido. Então é necessário usar um algoritmo de ordenação, que nada mais é que um processo lógico de organizar algum tipo de estrutura linear. Este estudo tem como objetivo avaliar experimentalmente os dados gerados usando o algoritmo de ordenação por inserção. Seis vetores de diferentes tamanhos foram testados em três cenários e os dados coletados foram comparados.

Sumário

1 INTRODUÇÃO

Ao iniciar com um vetor de capacidade N, são avaliados os tempos de execução de:

1. Uma lista ordenada em ordem crescente
2. Uma lista ordenada em ordem decrescente
3. Uma lista desordenada com números aleatórios entre 1 e N

Esse procedimento é realizado com vetores de tamanhos (valores de N) 10, 100, 1.000, 10.000, 100.000 e 1.000.000. Com os resultados espera - se apontar as vantagens e desvantagens ao se fazer uso de um algoritmo.

O estudo se dispõe da seguinte forma: na próxima seção (2 Algoritmos de Ordenação) será apresentado de forma sucinta o algoritmo de ordenação cujo se faz presente como objeto de estudo deste projeto Insertion Sort e seu respectivo código implementado na linguagem C; na terceira seção (3 Resultados) os dados coletados nos testes realizados serão apresentados; e finalmente na quarta seção (4 Conclusão) serão apresentadas as conclusões sobre à análise dos resultados obtidos na seção anterior.

2 ALGORITMOS DE ORDENAÇÃO

Os algoritmos de ordenação são algoritmos que direcionam a ordenação e rearranjo dos valores apresentados em uma determinada sequência, para que os dados possam então ser acessados com mais eficiência. Um dos principais objetivos desse tipo de algoritmo é a ordenação de vetores, pois uma mesma variável pode ter várias colocações, dependendo do tamanho do vetor declarado. Por exemplo, organizar uma lista de frequência escolar para que a lista seja organizada em ordem alfabética.

2.1 INSERTION SORT

O Insertion Sort é de fácil implementação e seu funcionamento se dá por comparação e inserção direta. A medida que o algoritmo percorre a lista, o mesmo os organiza, um a um, em sua posição mais correta. De forma resumida é como se basicamente o algoritmo procurasse o elemento de menor valor e o inserisse em uma nova lista, ao final de sua execução a nova lista estará ordenada, por isso o nome Insertion Sort, ordenação por inserção.

Figura 1. implementação do algoritmo insertion sort

```c
void insertion_sort(structure *s){
    int current_number;
    int current_index;
    s->execution_time = 0.0;
    clock_t begin = clock();
    for(int i = 1; i < s->input_size; i++){
        current_number = s->input_list[i];
        current_index = i - 1;
        while(current_index >= 0 && s->input_list[current_index] > current_number){
            s->input_list[current_index + 1] = s->input_list[current_index];
            current_index--;
        }
        s->input_list[current_index + 1] = current_number;
    }
    clock_t end = clock();
    s->execution_time += (double)(end - begin) / CLOCKS_PER_SEC;
}
```

Pode ser observado na Figura 1 uma função implementada em Linguagem C que, recebe uma estrutura por parâmetro e ordena sua lista de entrada (s->input_list) e calcula seu tempo de execução.

3 RESULTADOS

Foi definido como meio de codificação a linguagem C juntamente com a IDE Dev-C++ como ambiente de desenvolvimento, para a realização dos testes que serão apresentados a seguir. Em relação ao hardware utilizado, a máquina em que foram executados as simulações possui as seguintes especificações:

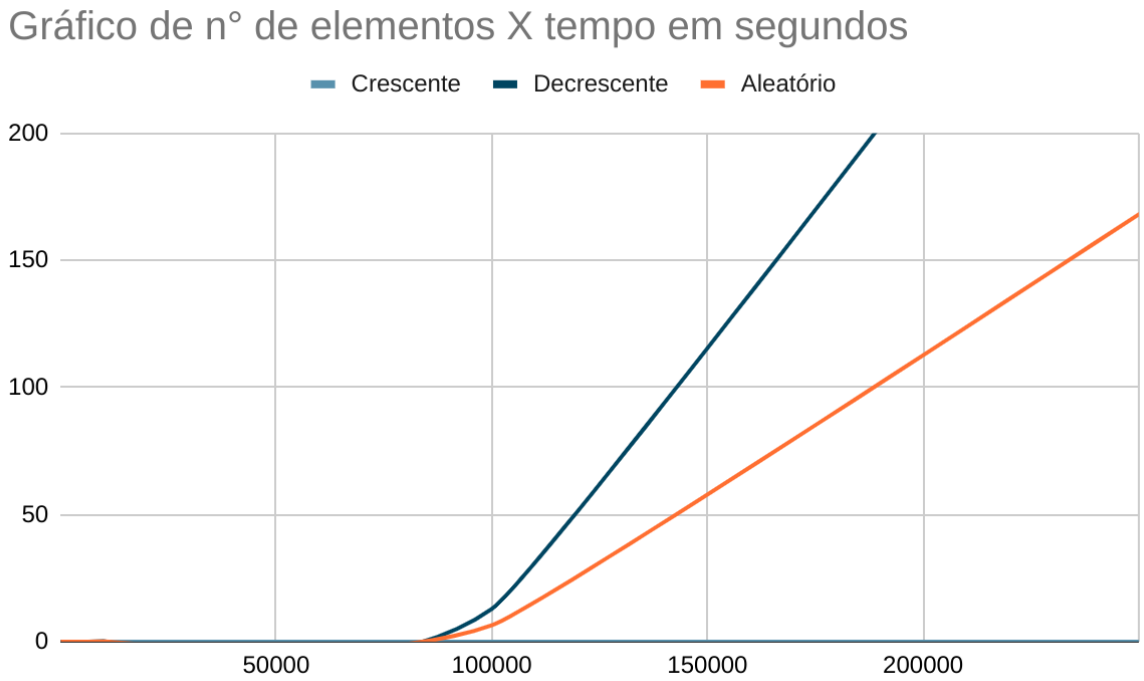
- Processador: AMD Ryzen 5 3600;
- Frequência: 4000 GHz;
- Memória RAM: 16GB (DDR4 3.200MHz);
- Sistema Operacional: Ubuntu (Linux)

3.1 INSERTION SORT

Tabela 1. Tempo de execução por tamanho do vetor

	10	100	1.000	10.000	100.000	1.000.000
Crescente	0.000001	0.000001	0.000004	0.000041	0.000300	0.002898
Decrescente	0.000000	0.000014	0.001278	0.128645	12.834253	2028.224182
Aleatório	0.000001	0.000008	0.000771	0.066064	6.417593	1016.672662

Gráfico 1



4 CONCLUSÃO

4.1 INSERTION SORT

O Insertion Sort, por sua vez, é útil para estruturas lineares pequenas, pode ser observado no Gráfico 1 que, até perto da ordem de 100.000 objetos a diferença de tempo de execução entre as ordens é praticamente nula, e após esse intervalo, a diferença entre os cenários(crescente, decrescente e aleatório), já cresce exponencialmente. Em seu melhor caso o algoritmo tem como, em notação BIG O, O(n) que é linear em relação ao número de objetos da lista. Já no seu pior caso tem como notação $O(n^2)$, quadrática, crescendo exponencialmente em relação à quantidade de ítens da lista.

5 REFERÊNCIAS

Szwarcfiter, J. L. and Markezon, L. (2015). “Estruturas de Dados e Seus Algoritmos.” 3ª edição. Rio de Janeiro. LTC.

SOUZA, Jackson EG; RICARTE, João Victor G.; DE ALMEIDA LIMA, Náthalee Cavalcanti. Algoritmos de Ordenação: Um estudo comparativo. **Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)**, n. 1, 2017.