PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Relatório de Análise Algoritmos de Ordenação Quadráticos*

Analysis Report Quadratic Sort Algorithms

Samuel Clementino de Oliveira Rocha[1]

Resumo

O resumo deverá conter pelo menos cento e cinquenta palavras de acordo com o padrão de normalização da ABNT. Este artigo irá abordar as principais linguagens de programação voltadas a ambiente WEB usadas atualmente, comparando suas características de maneira a indicar o melhor uso para determinada linguagem. As linguagens serão divididas de acordo com 4 principais características: Interpretadas, compiladas, server-side e client-side. O resumo deverá conter pelo menos cento e cinquenta palavras de acordo com o padrão de normalização da ABNT. as linguagens serão divididas de acordo com 4 principais características: Interpretadas, compiladas, server-side e client-side. O resumo deverá conter pelo menos cento e cinquenta palavras de acordo com o padrão de normalização da ABNT.

Palavras-chave: Template. LaTeX. Abakos. Periódicos.

*Relatório apresentado para a matéria de Algoritmos e Estruturas de Dados II, do curso de Engenharia de Computação, ministrada pelo professor Kleber Jacques Ferreira de Souza.

[1]Aluno do Programa de Graduação em Engenharia de Computação, Brasil– samuel.clementino@sga.pucminas.br .

Abstract

The abstract should contain at least one hundred and fifty words in accordance with the standards of ABNT standard. This present article will address the main features of the web programming languages, that are used currently, comparing its features as to indicate to indicate the better use of determined language. The linguage will be divided according with four major caracteristics: Interpreted, compiled, server-side and client-side. This present article will address the main features of the web programming languages. The abstract should contain at least one hundred and fifty words in accordance with the standards of ABNT standard. The linguage will be divided according with four major caracteristics: Interpreted, compiled, server-side and client-side. This present article will address the main features of the web programming languages. The abstract should contain at least one hundred and fifty words in accordance with the standards of ABNT standard.

Keywords: Template. LaTeX. Abakos. Periodics.

1 INTRODUÇÃO

Este relatório tem como objetivo expor os resultados obtidos durante a implementação de algoritmos de ordenação quadráticos, bem como a medição do desempenho (tempo de execução) de cada algoritmo em uma determinada máquina; a análise de desempenho dos algoritmos baseada nas medições realizadas; e comparação entre a análise de complexidade e a de desempenho.

2 DESENVOLVIMENTO

2.1 Implementação

A primeira parte do trabalho consiste na implementação dos algoritmos de ordenação estudados em sala: Bolha, Seleção e Inserção. Esta implementação será usada nas fases de medida e análise de desempenho tanto em termos de análise de complexidade quanto tempo de execução.

2.1.1 Bubble sort

Algoritmo 1 - Bubble sort

Algorithm 1: Bubble sort

```
1:  void bolha (int * array, intn)
2:  {
3:  int trocas = 0, comp = 0;
4:  for (int i = (n − 1); i > 0; i − −)
5:  {
6:  for (int j = 0; j < i; j + +)
7:  {
8:  comp + +;
9:  if (array[j] > array[j + 1])
10: {
11: trocas + +;
12: swap (array[j], array[j + 1])
13: }
14: }
15: }
16: }
```

Fonte: Souza(2022).

2.1.2 Insertion sort

Algoritmo 1 - Insertion sort

Algorithm 2: Insertion sort

```
1: void insercao (int * array, intn)
2: {
3: for (int i = 1; i > n; i++)
4: {
5: int tmp = array[i];
6: int j = i - 1;
7: while ((j >= 0) && (array[j] > tmp))
8: {
9: array[j + 1] = array[j];
10: j--;
11: }
12: array[j + 1] = tmp
13: }
14: }
```

Fonte: Souza(2022).

2.1.3 Selection sort

Algoritmo 1 - Selection sort

Algorithm 3: Selection sort

```
1: void selecao (int * array, intn)
2: {
3: for (int i = 0; i > (n - 1); i++)
4: {
5: int menor = i;
6: int j = i - 1;
7: for (intj = (i + 1); j < n; j++)
8: {
9: if (array[menor] > array[j])
10: {
11: menor = j;
12: }
13: }
14: swap (&array[menor], array[i])
15: }
16: }
```

Fonte: Souza(2022).

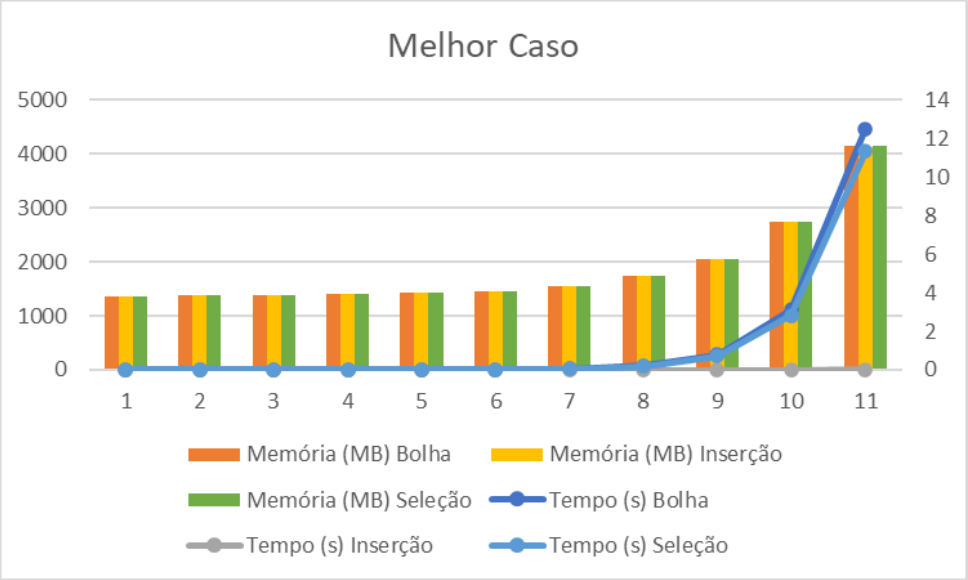
2.2 Análise de Desempenho Experimental

Após a obtenção de dados de desempenho gerado pelos algoritmos, colocamos os dados de tempo e memória consumida em cada caso (melhor, pior e intermediário caso).

2.2.1 Melhor caso

O melhor caso se dá quando os valores dentro do vetor já estão ordenados.

Gráfico 1 - Melhor Caso



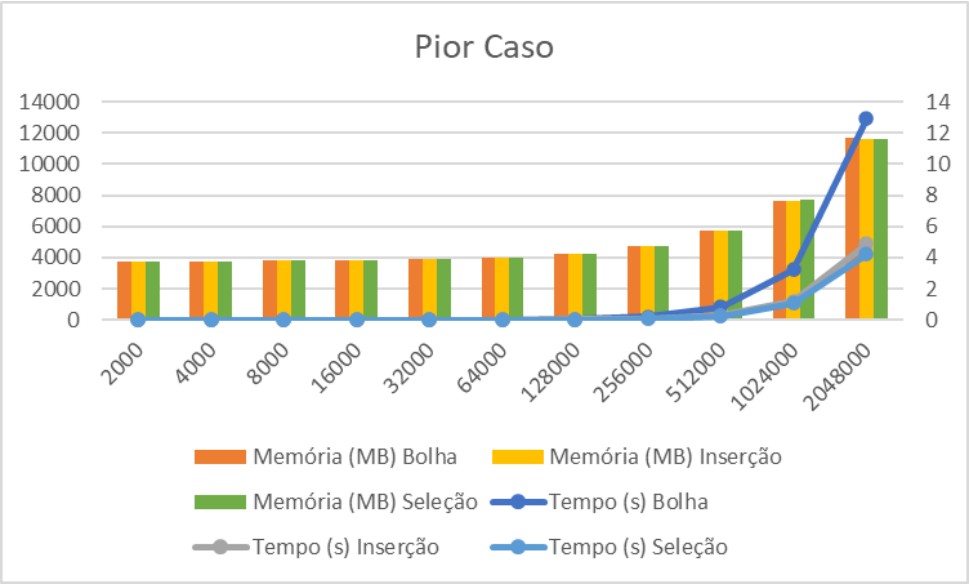
Fonte: Própria (2022)

Como podemos observar o algoritmo que mais se destacou foi o de inserção, já que levou aproxidamente 0 segundos para terminar a ordenação, sendo que os outros dois levaram mais de 4000 segundos. Já o consumo de memória é muito próximo nos três casos.

2.2.2 Pior caso

O pior caso se dá quando os valores dentro do vetor já estão ordenados ao contrário.

Gráfico 2 - Pior Caso



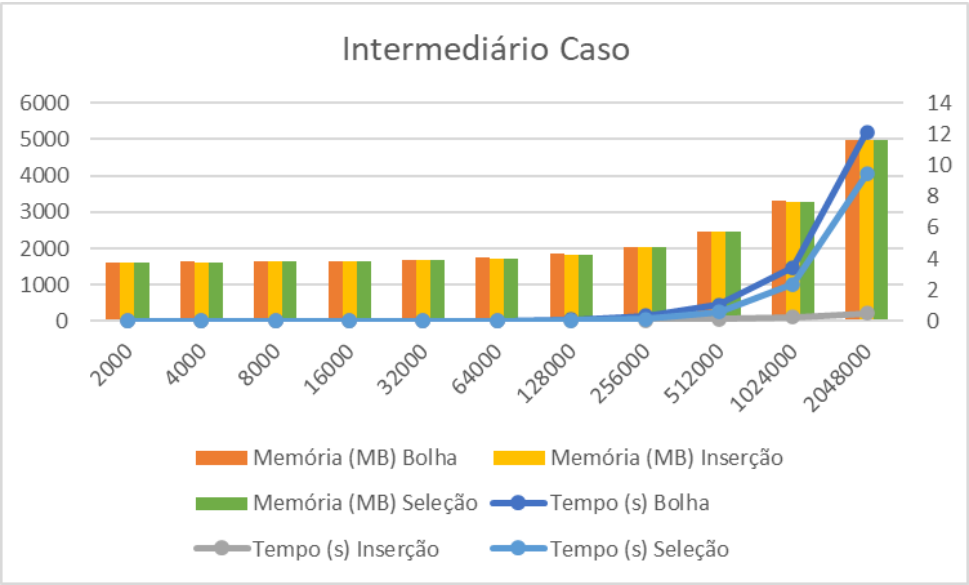
Fonte: Própria (2022)

Como podemos observar o algoritmo que mais se destacou foi o bolha, já que levou quase 3 vezes mais que os outros dois algoritmos, totalizando 12903 segundos. Já o consumo de memória, novamente é muito próximo nos três casos.

2.2.3 Intermediário caso

O intermediário caso se dá quando os valores dentro do vetor são totalmente aleatórios, sem nenhuma pré-ordenação já estabelecida.

Gráfico 3 - Intermediário Caso



Fonte: Própria (2022)

Como podemos observar novamente o algoritmo que mais se destacou foi o inserção, já que levou um pouco mais 200 segundos para terminar a ordenação, sendo que os outros dois levaram mais de 4000 e 5000 segundos. Já o consumo de memória, novamente é muito próximo nos três casos.

2.3 Análise dos Resultados Analíticos e Experimentais

A notação Big O é uma maneira de classificar a escalabilidade de sua função ou algoritmo. Isso nos ajuda a determinar como nosso algoritmo se comportará em caso de PIOR ou quanto tempo levará para ser concluído com base em seus valores de entrada.

A complexidade dos algoritmos para o melhor caso:

- Bubble sort é O(n);
- Insertion sort é O(n);
- Selection sort é O(n^2).

A complexidade dos algoritmos para o pior caso:

- Bubble sort é O(n^2);
- Insertion sort é O(n^2);
- Selection sort é O(n^2).

A complexidade dos algoritmos para o intermediário caso:

- Bubble sort é O(n^2);
- Insertion sort é O(n^2);
- Selection sort é O(n^2).

2.4 Informações Adicionais

2.4.1 Especificações do computador

- Processador: Intel Core i5 4590 3,30 GHz;
- Memória: 2X4GB Corsair Vengeance Pro DDR3 1866MHz.

2.4.2 GitHub

https://github.com/samuelcorocha/C-Quadratic-Sort-Algorithms

3 CONCLUSÃO

Considerando as medidas e análises realizadas neste atividade, concluímos que a complexiade dos algoritmos tem uma média igual, $O(n^2)$ e que o uso de memória também muda pouco. Todavia, percebemos difereças abruptas quando analisamos o tempo necessário para a execução de cada algoritmo.

Analisando isso, podemos perceber que dentre os três algoritmos de ordenação quadrática, o que tem melhor resultado é o Insertion sort.

REFERÊNCIAS