UFC

UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS-RUSSAS

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PAULO HENRIQUE DINIZ DE LIMA ALENCAR

ESTRUTURA DE DADOS AVANÇADAS – TRABALHO 1 - PARTE 1

RUSSAS

2021

1 VISÃO GERAL

Primeiro trabalho na disciplina de estrutura de dados Avançadas. O trabalho consiste na implementação dos algoritmos: MaxHeap, Heap-Sort, Insertion-Sort e Tabela-Hash. Após a implementação, os algoritmos deveriam ser testados com diferentes instâncias, contextos e métodos.

Outro requisito do trabalho consiste em implementar as funções de dispersão (função hash) considerando os métodos da divisão, dobra , multiplicação e análise de dígitos. Em seguida realizar a comparação dos métodos com determinados tamanhos para ‘n’.

2 ESPECIFICAÇÕES GERAIS

- Linguagem de implementação: C;
- Máquina utilizada na análise dos algoritmos: notebook ACER-Aspire ES15 ES1-572-3562, SSD de 100 Gb, 12 Gb de memória RAM (DDR4) e processador Intel Core i3-6006U (2.0 GHz L3 Cache);
- Sistema Operacional: Linux – Zorin-OS – 64 bits;
- Compilador: gcc versão 9.3.0.

3 ALGORITMOS DE ORDENAÇÃO: HEAP-SORT E INSERTION-SORT

3.1 Análise Empírica

Seguindo os critérios estabelecidos pelo professor da disciplina, foram gerados 4 arquivos com valores pseudoaleatórios (inteiros) com tamanhos: 1.000, 100.000, 10.000.000 e 1.000.000.000. Cada arquivo foi lido e seus valores foram transferidos para um determinado vetor. Em seguida, o vetor foi ordenado pelo Heap-Sort e Insertion-Sort para cada um dos tamanhos citados acima. O tempo gasto por cada algoritmo na ordenação, baseado no comprimento do vetor serão apresentados graficamente nos tópicos seguintes.

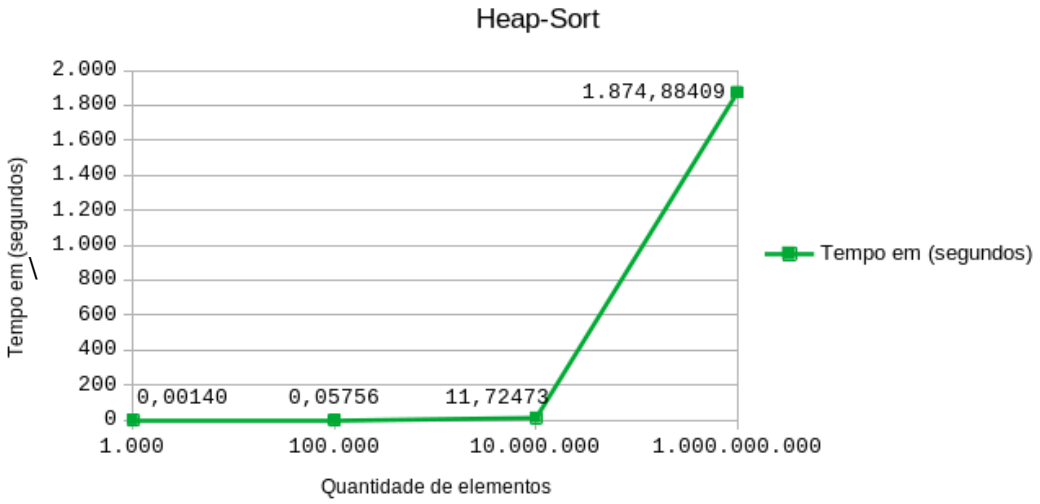
> **Observação:** os arquivos foram gerados utilizando um *script* python através da biblioteca *numpy*. Além disso, os números pseudoaleatórios inseridos nos arquivos possuem um range (0, 2.000.000) em média.

3.1 Complexidade

Algoritmo	Complexidade de pior caso
Heap Sort	$\Theta(n \cdot logn)$
Insertion Sort	$\Theta(n^{2})$

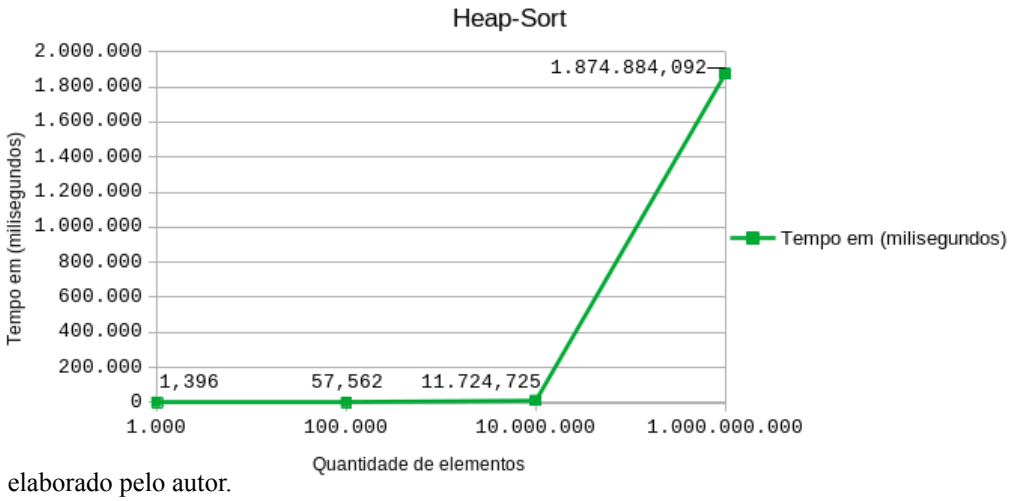
3.2 Resultados - Heap-Sort

Gráfico 1 - Tempo gasto em segundos pelo heapsort para realizar a ordenação de *mil*, *cem mil*, *dez milhões* e *um bilhão de elementos* aleatórios presentes em um vetor.



Fonte: elaborado pelo autor.

Gráfico 2 - Tempo gasto em milissegundos pelo heapsort para realizar a ordenação de *mil*, *cem mil, dez milhões* e *um bilhão* de elementos aleatórios presentes em um vetor.



Fonte: elaborado pelo autor.

3.2.2 Insertion-Sort

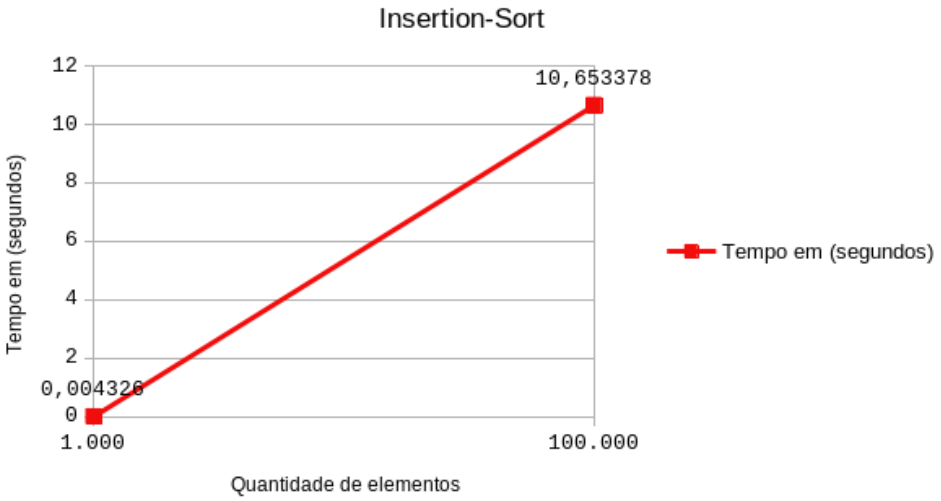
Em relação a análise empírica dos dados solicitadas pelo professor fazendo uso do Insertion-Sort, foram obtidos os seguintes resultados:

1. Foram obtidos apenas o tempo de ordenação para 1.000 elementos e 100.000 elementos;

2. O algoritmo Insertion-Sort tentou ordenar durante 24 horas o arquivo com entrada 10.000.000. Após esse período, foi decidido encerrar o processo e desligar a máquina;

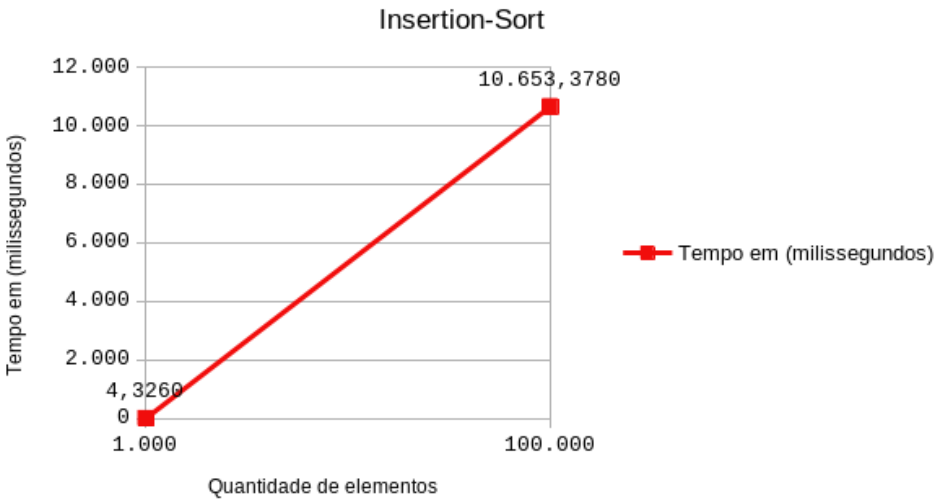
3. Ainda foi pensado na possibilidade de obter um tempo máximo no pior caso (vetor ordenado decrescentemente). O tempo máximo poderia ser obtido fazendo alguns experimentos para encontrar o tempo médio de execução de uma operação no computador e depois multiplicando esse tempo por n^2(complexidade do Insertion-Sort).

Gráfico 3 - Tempo gasto em segundos pelo Insertion-Sort para realizar a ordenação de *mil* e *cem mil* elementos aleatórios.



Fonte: elaborado pelo autor.

Gráfico 4 - Tempo gasto em milissegundos pelo Insertion-Sort para realizar a ordenação de *mil* e *cem mil* elementos aleatórios.



Fonte: elaborado pelo autor

4. COMPARAÇÃO DOS MÉTODOS DE DISPERSÃO DA TABELA HASH

Como citado nas especificações deste trabalho foram necessário criar tabelas com tamanhos de m = 100.000.

- 1 tabela para o Método da divisão;
- 1 tabela para o Método da multiplicação;
- 1 tabela para o Método da dobra.

Em seguida realizei os testes de colisões com vetores de chaves com o n's:

n = 200.000;

n = 400.00;

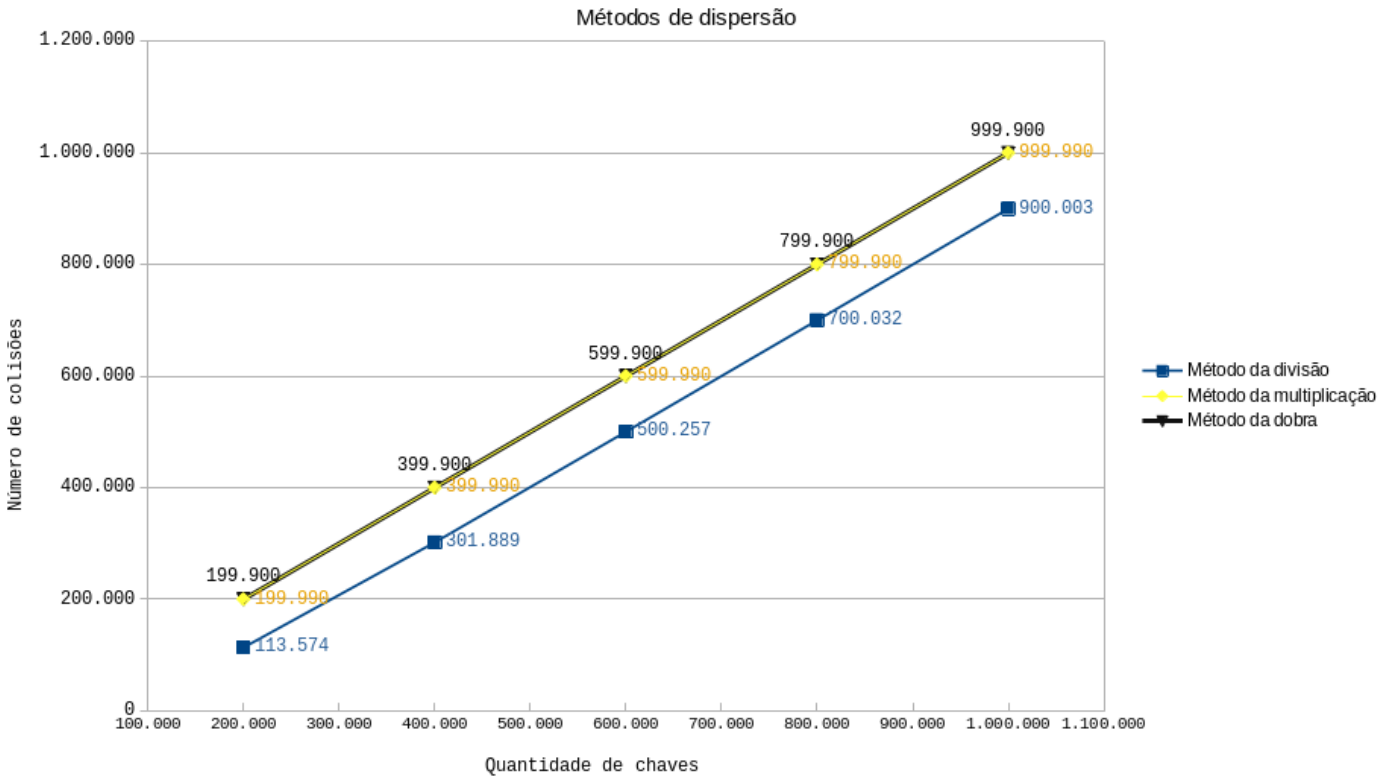
n = 600.000;

n = 800.000;

n = 1.000.000.

Os números aleatórios colocados nos vetores de chaves, foram inseridos em arquivos .txt com um range de [0, 2.000.000.000].

Gráfico 5: representa (quantidade de chaves n) × (nº de colisões) em que é mostrado as 3 curvas (uma curva para método da divisão, multiplicação e dobra) .



Fonte: elaborado pelo autor

Gráfico 6: análise empírica com o método da divisão.

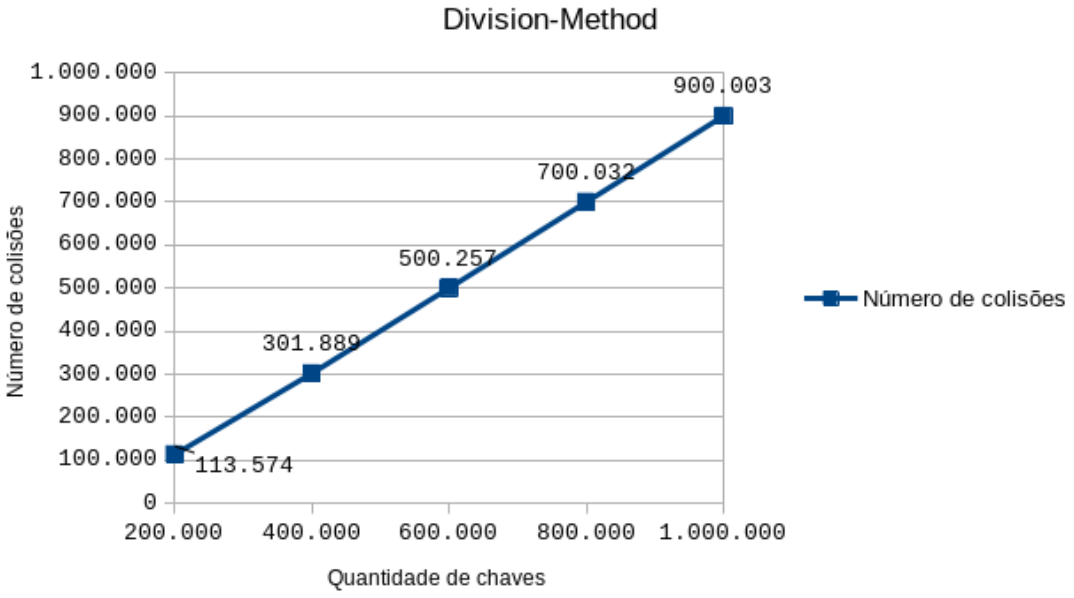
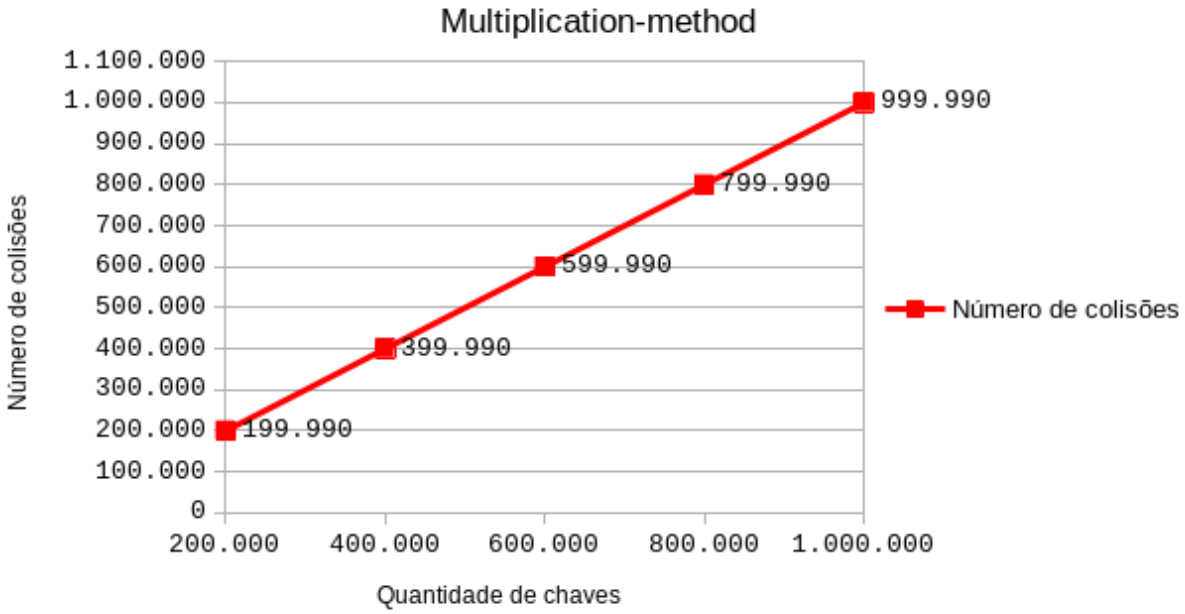
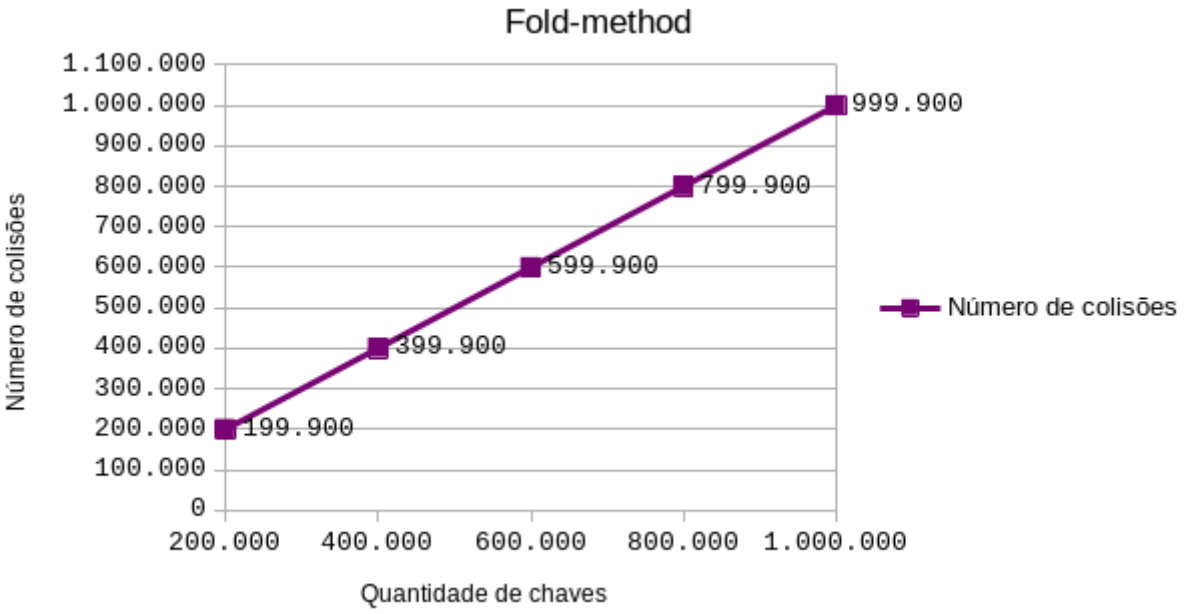


Gráfico 7: análise empírica com o método da multiplicação.



Fonte: elaborado pelo autor

Gráfico 9: análise empírica com o método da dobra.



Fonte: elaborado pelo autor

Assim, com base nessas análises e entradas já citadas (m = 100.000, n = 200.000, n = 400.000…) podemos concluir que o método da divisão foi o que gerou menor número de colisões.