PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
UNIDADE CONTAGEM
ICEI – INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA

Arthur Andrade Gonçalves
Assuério Batista dos Santos
Lucas Braga Ferreira
Marcos Pablo Souza de Almeida
Samuel Lucas Oliveira Martins

Algoritmos de Pesquisa

CONTAGEM
2019

1. Introdução:

O trabalho consiste na implementação e análise de diferentes algoritmos de pesquisa estudados em sala, sendo eles: Pesquisa Sequencial, Pesquisa Binária, Árvore Binária e Tabela Hash.

A base de dados utilizada para os testes encontram disponibilizados no link https://1drv.ms/f/s!Ai1mr_X9-Sz7lLYXlG8rF6RpaWD0nQ pertencentes à empresa AirBnB, a qual efetua a locação de casas, apartamentos e quartos.

2. Tomada de Decisão

Para o início do trabalho, foi necessário escolha entre quais chaves seriam utilizadas. Dentre elas foram selecionadas chaves a partir da ordem original dos dados na base de Arquivos da AirBnB onde foram selecionado em um Array a posição 0, n/2 e n

- 10047973
- 913589
- 16389974

3. Configurações do cenário

Inicialmente foi configurado um repositório no *GitHub* a fim de que todos desenvolvedores tivessem acesso ao código. Posteriormente um scopo base foi implementado com algumas funções de leitura e gravação dos dados da base utilizada para os testes.

A priori, havíamos decidido utilizar o tempo gasto em cada algoritmo como parâmetro da análise. Entretanto, tendo em vista que as estruturas de complexidade logarítmicas só atingiram a casa dos nanossegundos (ns), e que o método Timer (temporizador) da linguagem C# contabiliza o tempo a partir dos milissegundos (ms), optamos por aferir o número de comparações realizadas até obter-se o resultado desejado.

A máquina utilizada para os testes apresenta as seguintes especificações:

- DELL Inspiron
- Intel Core i5 5210 - U
- Frequência de processamento: 1.7Ghz a 2.4GHz
- 2 pentes de memória de 4GB 1600mhz (dual Channel)

4. Início do desenvolvimento

4.1- Pesquisa Sequencial

O algoritmo de pesquisa sequencial consiste em, verificar cada uma das posições do vetor de dados sequencialmente, iniciando na primeira posição da coleção e finalizando ao achar o dado procurado ou ao chegar ao fim do vetor. Apresenta seu melhor caso de pesquisa com custo **O(1)** e os casos médio e pior caso **O(N).**

Para o algoritmo de pesquisa sequencial utilizamos o Array contendo todos os registros da base de dados teste, comparando um a um elemento até encontrar o registro desejado.

4.2 - Pesquisa Binária

O algoritmo de pesquisa binária consiste em trabalhar com vetor com os dados e ir dividindo a pesquisa pela metade, “dividir para conquistar”, onde é sempre verificado se o elemento do meio do vetor ou sub vetor é o elemento procurado, caso não seja é verificado se a chave procurada é maior ou menor que a do meio. Caso seja maior o sub vetor será o da direita , caso seja menor o sub vetor será o da esquerda. O vetor para este tipo de pesquisa deve estar ordenado.

Para o algoritmo de pesquisa Binária implementamos o Algoritmo de ordenação do *Quicksort* para ordenar o *Array* e utilizar o algoritmo de Pesquisa Binária.

4.3 - Árvore Binária

O algoritmo de árvore binária consiste em trabalhar com encadeamento de células sendo elas denominadas de “nó” contendo os dados, uma chave para comparação de inserção e dois ponteiros (esquerda e direita). É verificado a cada inserção se a chave do novo nó é maior ou menor que o último nó inserido, caso seja menor é feito seu apontamento à direita do último nó inserido, caso seja maior é feito seu apontamento à esquerda do último nó inserido.

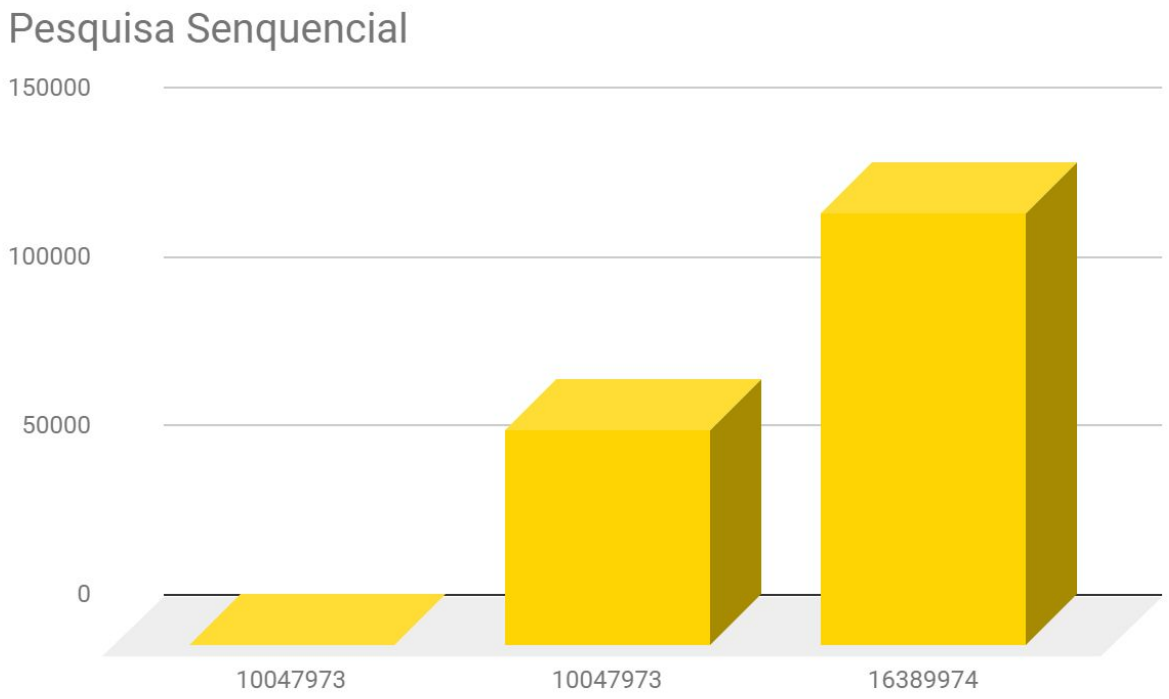
4.4 - Tabela Hash

A tabela Hash foi implementada utilizando o tamanho primo mais próximo da metade sendo esse número **63647** evitando maior quantidade de colisões. sempre inserindo da tabela pelo cálculo da localização da hash.

5. Análises

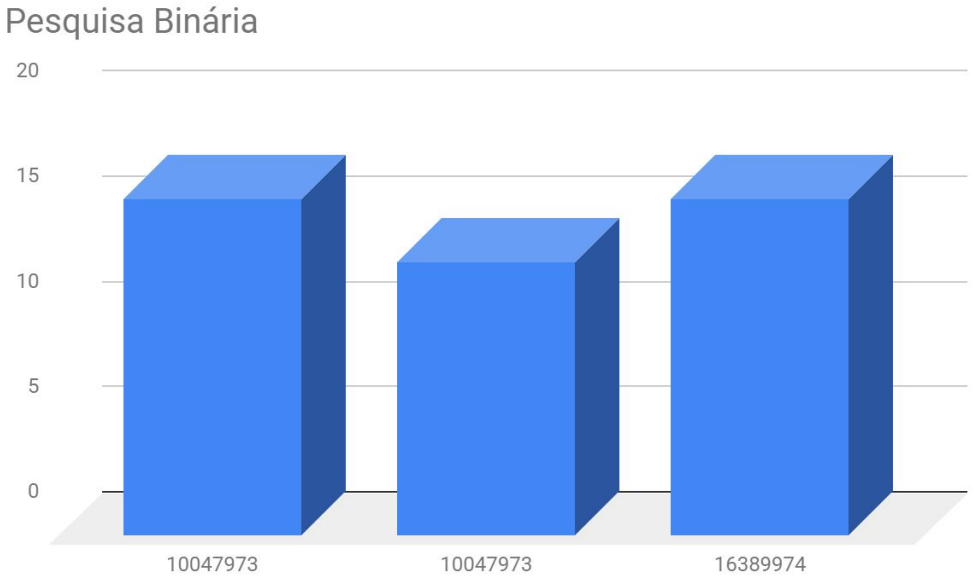
5.1 - Pesquisa Sequencial

Analisando a pesquisa sequencial podemos ver que o primeira pesquisa da chave que está no início do vetor foi de custo O(1) e o do meio O(n/2) e o final sendo O(n).



5.2 - Pesquisa Binária

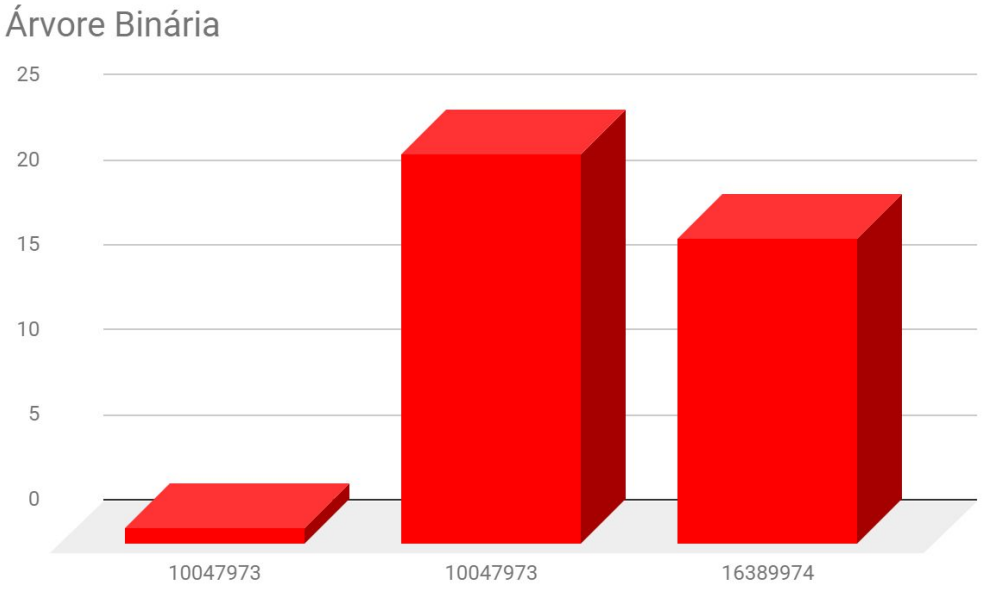
Na Pesquisa Binária, levando em consideração o seu comportamento logarítmico de complexidade O(*log* N) na maioria dos casos, obtivemos os seguintes resultados, ressaltando a que o vetor sempre tem de estar ordenado, não sendo bom em alguns casos:



Os números apresentados no eixo do gráfico referem-se ao número de comparações realizadas. Já o valor base das colunas dos gráficos faz menção ao room_id dos registros analisados.

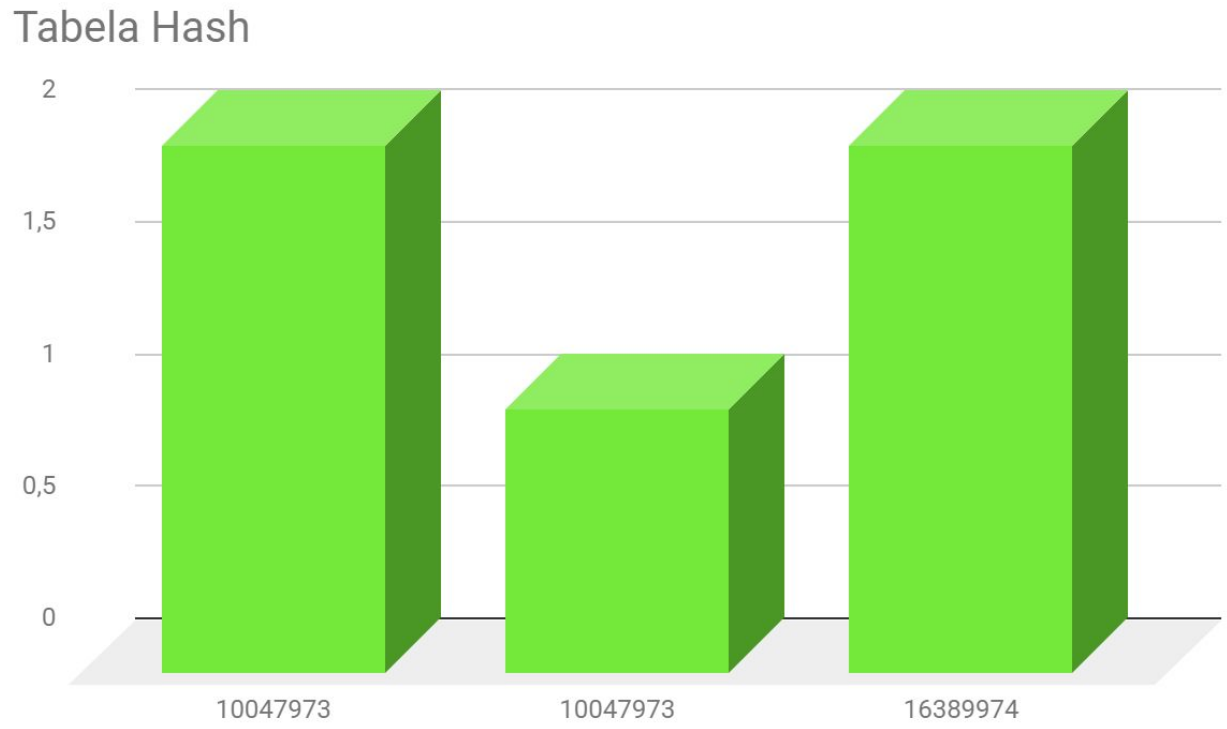
5.3 Árvore Binária

Analisando a pesquisa na Árvore Binária podemos observar que o primeira chave foi inserida primeiramente foi encontrado em O(1) pois ele foi o primeiro a ser inserindo se tornando a raiz da árvore, as outras chaves foram encontradas percorrendo a árvore com a complexidade O(log n)



5.4 Tabela Hash

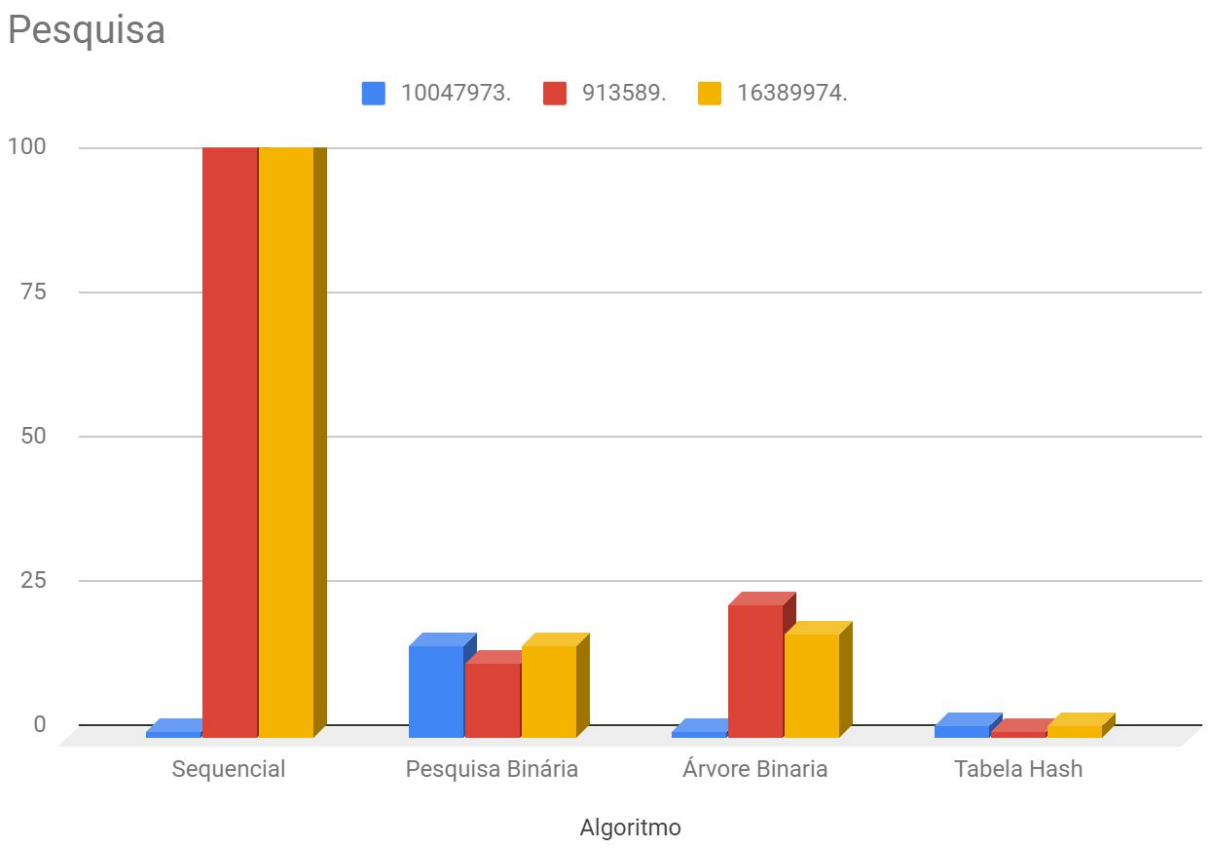
Analisando a pesquisa da Tabela Hash utilizando o número primo foi bem podemos observar as pesquisas utilizando poucas comparações, sendo otimizada e causando poucas colisões.



6. Conclusão

Cada algoritmo possui vantagens e desvantagens, dependendo da situação que se encontra é necessário analisar a melhor implementação, a seguir a tabela dos resultados obtidos:

Chave	Sequencial	Pesquisa Binária	Árvore Binária	Tabela Hash
10047973	1	16	1	2
913589	64001	13	23	1
16389974	128000	16	18	2



Sendo assim, dentre os algoritmos analisados, a Tabela Hash obteve os melhores resultados. Reiteramos que tal comportamento era esperado pelo fato de possuir a complexidade de O(1) para todos os casos.