

# Projeto de Hash com base de dados do Bolsa Família

## CEFET- Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Disciplina de Organização e Estrutura de Arquivos

Gabriel Eduardo Feitosa Lima

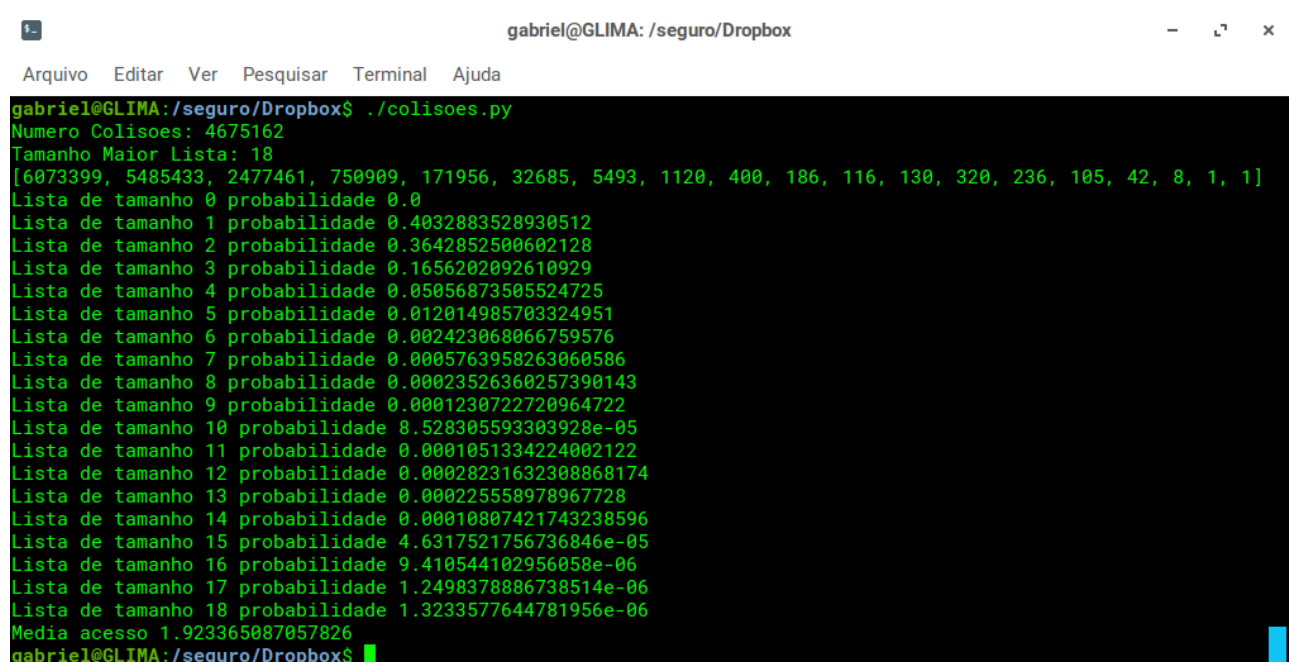
Resumo: O intuito deste estudo foi praticar a implementação da estrutura de indexação Hash, trabalhar com estruturas de arquivos com uma massa de dados bem elevada e efetuar uma breve comparação entre métodos de busca, utilizando estrutura de arquivos já indexada, que utilizam como parâmetro de comparação o tempo.

### Introdução

A disciplina de *Organização e Estruturas de Arquivos*, demonstrou com muita maestria, as diversas formas de estruturas de arquivos que podemos utilizar, seja ela uma Árvore B, Árvore B+, Árvore Tree, Estruturas Espaciais, Estruturas de Hash ou até mesmo uma simples estrutura onde utiliza-se apenas um arquivo como objeto de armazenamento sem nenhum tipo indexação, ou qualquer outro modelo de organização. Neste estudo foi utilizada a estrutura de indexação *Hash* usando como comparativo a técnica de busca conhecida como ‘força bruta’, ambas aplicadas sobre todos os arquivos com registro do programa social *Bolsa Família do Brasil*, arquivo este que possui, em média, mais de 13 milhões de registros, sendo assim, considerada uma alta base de dados.

### Estruturas de Hash

Foram elaborados alguns códigos utilizando a linguagem de programação *Python3*. Estes códigos têm como objetivo efetuar o processo de indexação dos arquivos do *Bolsa Família*, e algumas verificações da própria estrutura de arquivos. A *Figura 1* representa um código de colisões.py.



```
gabriel@GLIMA: /seguro/Dropbox
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
gabriel@GLIMA:/seguro/Dropbox$ ./colisoes.py
Numero Colisoes: 4675162
Tamanho Maior Lista: 18
[6073399, 5485433, 2477461, 750909, 171956, 32685, 5493, 1120, 400, 186, 116, 130, 320, 236, 105, 42, 8, 1, 1]
Lista de tamanho 0 probabilidade 0.0
Lista de tamanho 1 probabilidade 0.4032883528930512
Lista de tamanho 2 probabilidade 0.3642852500602128
Lista de tamanho 3 probabilidade 0.1656202092610929
Lista de tamanho 4 probabilidade 0.05056873505524725
Lista de tamanho 5 probabilidade 0.012014985703324951
Lista de tamanho 6 probabilidade 0.002423068066759576
Lista de tamanho 7 probabilidade 0.0005763958263060586
Lista de tamanho 8 probabilidade 0.00023526360257390143
Lista de tamanho 9 probabilidade 0.0001230722720964722
Lista de tamanho 10 probabilidade 8.528305593303928e-05
Lista de tamanho 11 probabilidade 0.0001051334224002122
Lista de tamanho 12 probabilidade 0.00028231632308868174
Lista de tamanho 13 probabilidade 0.000225558978967728
Lista de tamanho 14 probabilidade 0.00010807421743238596
Lista de tamanho 15 probabilidade 4.6317521756736846e-05
Lista de tamanho 16 probabilidade 9.410544102956058e-06
Lista de tamanho 17 probabilidade 1.2498378886738514e-06
Lista de tamanho 18 probabilidade 1.3233577644781956e-06
Media acesso 1.923365087057826
gabriel@GLIMA: /seguro/Dropbox$
```

Figura 1 – saída do código de colisões.py

Na *figura 1* podemos perceber que um arquivo indexado possui o tempo médio de busca de 1.9 segundos, ou seja, quando há necessidade de identificar uma pessoa no arquivo indexado, o tempo de resposta é de apenas 2 segundos, aproximadamente.

Foi estabelecida uma estrutura para que possa ser realizada a indexação das chaves, que seguem o molde retratado na *Figura 2*: “14sLL”, onde podemos identificar as seguintes simbologias: A 14s identifica a quantidade de caracteres da chave que será utilizada para indexar, no caso deste estudo, o NIS do beneficiado. O primeiro L simboliza um Long, tipo de variável que guarda um valor inteiro com uma capacidade elevada de números, que, neste caso, guardará o número do registro (ou o número da linha onde registro está armazenado no arquivo). E, por último, o outro L, também um Long, armazenará o ponteiro (aponta o local onde a leitura deve iniciar, para captar o registro no arquivo) dos registros.

```
dataFormat = 2303143030323 #Formato da estrutura do Bolsa Família
indexFormat = "14sLL" #Formato da estrutura do arquivo de hash
keyColumnIndex = 2 #Posicao do NIS (o que sera procurado e chave pelo
```

Figura 2 – Formato da estrutura

A máquina utilizada para efetuar as indexações e os testes de comparação possui a seguinte configuração: processador core i7, 8GB de memória RAM com um SSD de 120 GB. O tempo necessário para indexar o arquivo do *Bolsa Família*, de tamanho 1,4 GB, com aproximadamente 14.000.000 de registros, foi de 725,9 segundos, ou seja, convertendo os segundos para minutos, temos a quantidade de, aproximadamente, 12 minutos.

## Testes de ambiente

Uma vez que o arquivo do *Bolsa Família* foi indexado, podemos dar início aos testes de ambiente e testes comparativos nos métodos de buscas de registros. Eis que efetuamos um confronto entre o método de busca com um arquivo já indexado e outro arquivo sem indexar, e o método onde nenhum dos dois arquivos foram indexados, conhecido como teste de *Brute Force* (Força Bruta) onde cada registro de um arquivo deve percorrer o outro arquivo por completo.

## Força Bruta

Para o teste de força bruta observa-se na tabela abaixo alguns comparativos de tempo em relação a quantidade de registros.

	1.000 Linhas de Fevereiro	10.000 Linhas de Fevereiro	100.000 Linhas de Fevereiro
1.000 Linhas de Janeiro	0.77 segundos	-	-
10.000 Linhas de Janeiro	-	71 segundos	-
100.000 Linhas de Janeiro	-	-	730 segundos
Arquivo completo Janeiro	0.98 segundos	117.8 segundos	9145 segundos

Figura 3 – Tempo X Quantidade de registros

Concluindo este teste, podemos notar que, se analisarmos a tabela acima que verifica 100.000 linhas de fevereiro no arquivo completo de janeiro, sem utilizar nenhuma estrutura, dura 9145 segundos, que convertidos em horas, são aproximadamente 2 horas e meia, gastas apenas para verificar se 100.000 linhas de fevereiro constam no arquivo de janeiro completo. Sendo assim, afirma-se que o tempo gasto para verificar todos os registros no arquivo completo do *Bolsa família* do mês de

fevereiro constam no arquivo completo do *Bolsa família* do mês de janeiro, levando em conta que ambos os arquivos completos possuem aproximadamente 14.000.000 linhas de registros, seria necessário aproximadamente 1 dia e meio de comparações.

## Tabela Hash

Na tabela abaixo, pode se analisar algumas informações retiradas dos testes utilizando a *estrutura hash*, onde são criados arquivos que armazenam as *chaves de hash*, junto com a linha onde esse registro estava e a posição dele no arquivo do *Bolsa Família*. Dos números que pudemos obter dos testes, utilizamos os seguintes dados: Tamanho da tabela Hash, Tempo de indexação, Maior Lista de colisões e Media de acessos. O que cada um representa está exemplificado abaixo da tabela.

Tamanho da tabela hash (linha)	Tempo de indexação (s)	Maior lista de colisões	Media de acesso (s)
15.000.001	725.9	18	1.9
20.000.001	725.9	20	1.6
200.000.001	2335.6	13	1.0

Figura 4 – Tabela Hash

Com esses valores, deve-se entender primeiro o que é cada atributo da tabela: *Tamanho da tabela de hash* é a quantidade de linhas que o arquivo de *hash* vai ter para poder guardar os indexes, esse valor define o tamanho do arquivo. No primeiro caso, onde o tamanho era 15.000.001 linhas, o tamanho do arquivo de hash era de 645MB, já com o aumento de mais 5.000.000 linhas totalizando 20.000.001 passou a ser de 731MB, com o aumento brusco para 200.000.001 linhas foi para o valor de 6,4GB, sendo este tamanho considerado grande.

Já o *Tempo de indexação*, dá-se ao tempo levado para coleta de todos os registros do arquivo do *Bolsa Família* e indexá-los no arquivo de *hash*. Pode-se analisar que, com um aumento pequeno não houve mudanças no tempo, porém, uma vez que o tamanho da tabela *hash* teve um aumento bruto, o tempo também aumentou consideravelmente.

Em todos os arquivos de *hash*, ocorrem as chamadas colisões, que são valores de *hash* gerados em cima de uma chave, e acabam se repetindo, por serem valores de outras chaves, ou seja, se os valores X e Y fossem guardados utilizando o calculo de *hash*, e ambos fossem colocados na posição 001, ocorreria uma colisão. Para esse estudo, foi realizado um método de tratamento de colisões, onde cria-se uma lista encadeada entre as chaves que caem na mesma posição. A partir daí, de acordo com a tabela, para se ter o menor tamanho de lista de colisões é necessário utilizar o maior tamanho possível da tabela *hash*.

O ultimo fator é a *Medida de acessos*, utilizada ao procurar um registro que já tenha sido indexado, como por exemplo, se quisermos realizar a seguinte busca NIS: 00000000010234, que sabemos que está no arquivo do *bolsa família* do mês de janeiro, considerando que este arquivo já foi indexado, se ele pertence à uma tabela que possui tamanho igual a 15.000.001 linhas, seria necessário aproximadamente 1.9 segundos para que o NIS fosse encontrado, já se o tamanho escolhido fosse o de 20.000.001 levaria cerca de 1.6 segundos para que o arquivo fosse localizado, e, no ultimo caso, se a tabela escolhida fosse 200.000.001, o arquivo seria encontrado em cerca de 1 segundo. Se esse teste fosse realizado sem indexação por hash levaria alguns minutos para que a tabela fosse varrida e as comparações realizadas no arquivo puro, uma vez em que ele não é ordenado, seria necessário,

no pior caso, varrer todas as 14.000.000 de linhas do arquivo até identificarmos o resultado da busca.

## **Comparações**

Foram elaborados dois códigos que possuem a mesma função: analisar quantos registros possuem a mais no mês de Fevereiro no *Bolsa Família* em relação ao mês de Janeiro. Um dos códigos realiza esta tarefa utilizando o mês de Janeiro já indexado, enquanto o outro faz as comparações sem que nenhum deles tenham sido indexados. O objetivo dos códigos era analisar questões de tempo, e de desempenho.

E, como já era esperado, o código que utiliza um dos meses já indexado é, indiscutivelmente, superior, em todos os quesitos, uma vez em que levou-se 211 segundos para verificar quantos registros haviam em Fevereiro que não constavam em Janeiro, enquanto o código que não utilizou o hash não foi capaz de completar o teste com o arquivo completo considerando que a comparação com 100.000 linhas de fevereiro com arquivo de Janeiro completo levou 2,5 horas e o arquivo de Fevereiro possui um total de 13.500.000 linhas. Seria necessário cerca de dois dias rodando o código para chegarmos a um resultado.

## **Conclusão**

Esse estudo auxiliou no esclarecimento do assunto de *hash* e proporcionou uma oportunidade de trabalho e implementação do conteúdo, gerando possíveis códigos, que podem ou não ser utilizados posteriormente. Foi notória a grande utilidade de trabalhar com estruturas indexadas. Para projetos futuros, seria interessante trabalhar e analisar as estruturas de indexação entre si, utilizando o *hash* em paralelo com a *árvore b+*. Ambas estruturas são utilizadas para indexação, cada uma com a sua peculiaridade, mas, ainda assim, cumprindo seu papel de eficiência.