Relatório Tabela Hash

João Pedro Igeski Morais

Pontifícia Universidade Católica do Paraná - PUCPR

1. Introdução

O objetivo desse relatório é comparar o desempenho das funções hash com resto da divisão, multiplicação e outra chamada personalizada. Essa função personalizada consiste em uma combinação entre as duas primeiras, sendo que vai calcular primeiro a posição com base na função hash da divisão, caso seja detectada uma colisão, vai recalcular a posição com base na função hash de multiplicação.

Para os testes foram considerados vetores de tamanho 10000, 20000, 30000, 40000 e 50000.

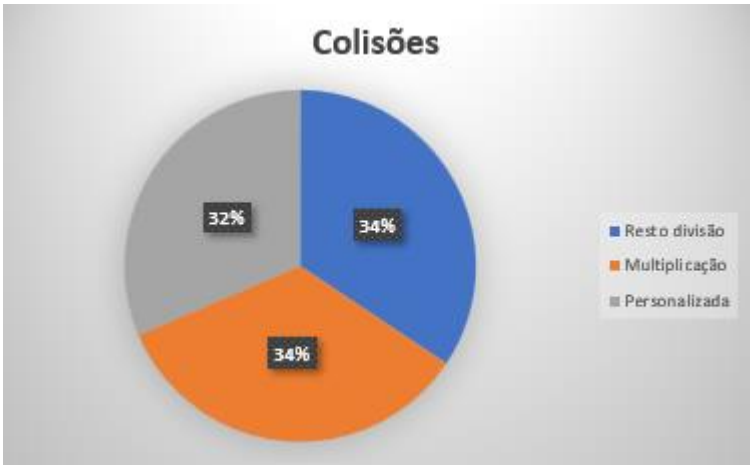
Link do GitHub: https://github.com/jpedro1711/AtividadeAvaliativa-TabelaHash

2. Inserção

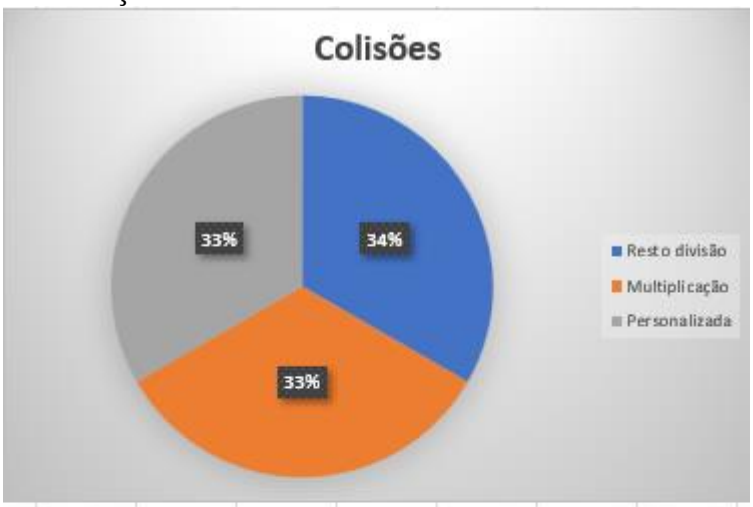
No cenário da inserção as funções hash tiverem desempenho muito parecido quando o tamanho do vetor da tabela hash era menor, sendo que a função personalizada obteve um número um pouco menor de colisões, o que ocorre pelo fator do recálculo da posição em caso de colisão. Além disso, o tempo de execução em milissegundos foi muito parecido.

Resultado para tabela hash com tamanho 10000 e 20000 dados

Função hash	Tamanho tabela hash	Número de dados	Colisões	Tempo de execução
Resto divisão	10000	20000	11352	232
Multiplicação	10000	20000	11298	266
Personalizada	10000	20000	10382	243
Total				3



No entanto, quando o número de dados na tabela hash de 10000 elementos aumenta, o tempo de execução da função hash personalizada é menor quando comparado as outras funções, no entanto, o número de colisões aumenta, se tornando bem parecido entre as diferentes funções. Esse cenário é representado abaixo, com tabela hash de tamanho 10000 e inserção de 5 milhões de elementos.



Quando aumentamos o tamanho do vetor da tabela hash, a função personalizada se mostrou ainda mais eficiente do que as outras duas, tanto em colisões quanto em tempo de execução, no cenário de 20000 dados. Isso ocorre por conta de que com uma capacidade maior, caso seja feito o recálculo da posição do elemento, a probabilidade do seu espaço estar ocupado (colisão) é menor.

Tabela hash de tamanho 20000 (inserção de 20000 dados)

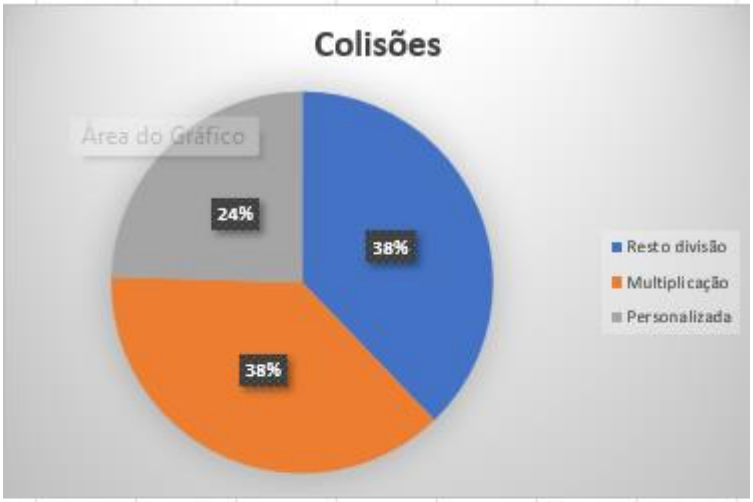
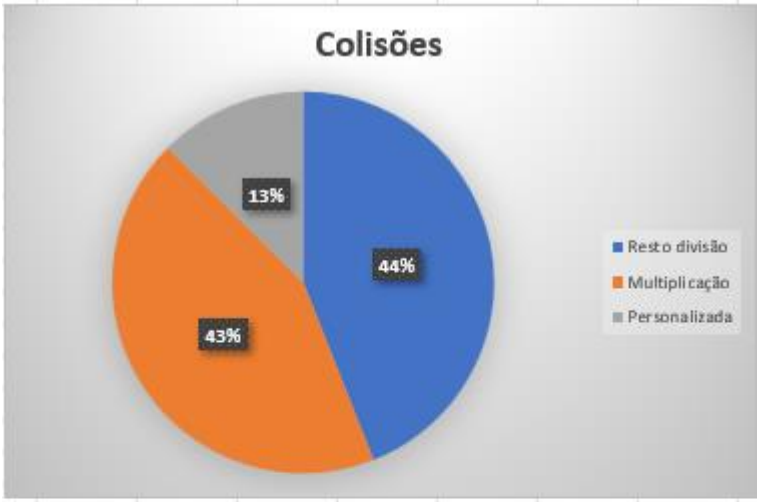
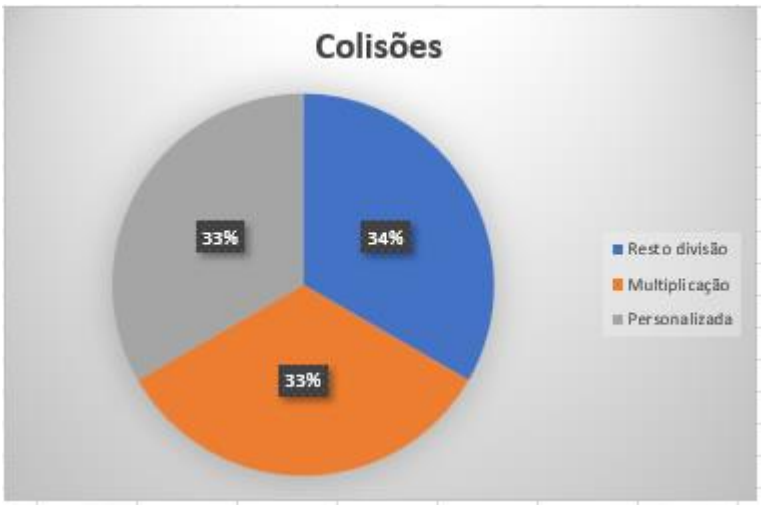


Tabela hash de tamanho 50000, inserção de 20000 dados



No entanto, quanto maior o número de elementos inseridos, menor é o desempenho da função de espalhamento personalizada, independentemente do tamanho da tabela. Isso ocorre por conta de que quanto maior o número de dados, menos espaços livres teremos, ocasionando em número maior de colisões, independentemente da função hash.

Tabela hash de tamanho 20000 (inserção de 5 milhões de dados)


Colisões
34%
33%
33%
Resto divisão
Multiplicação
Personalizada


Tempo de execução
32648
53383
39219
Resto divisão
Multiplicação
Personalizada

Tabela Hash de tamanho 20000 (inserção de 500000 dados)

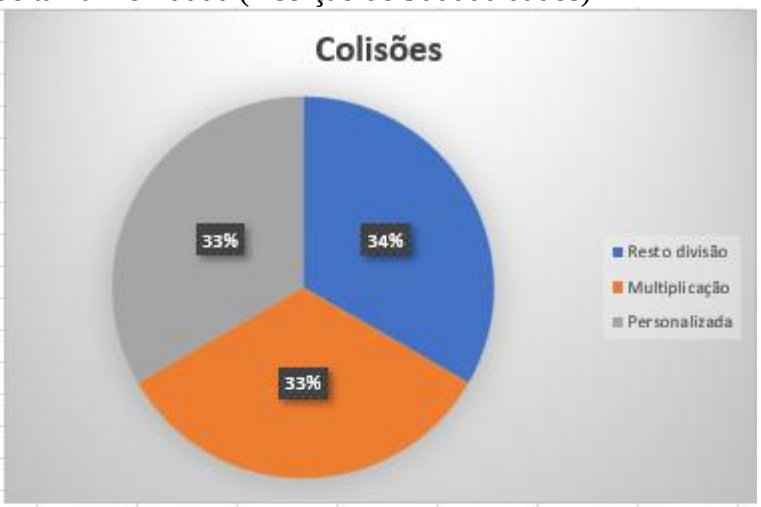


Tabela Hash de tamanho 40000 (inserção de 20000 dados)

Tabela Hash de tamanho 40000 (inserção de 500 mil dados)

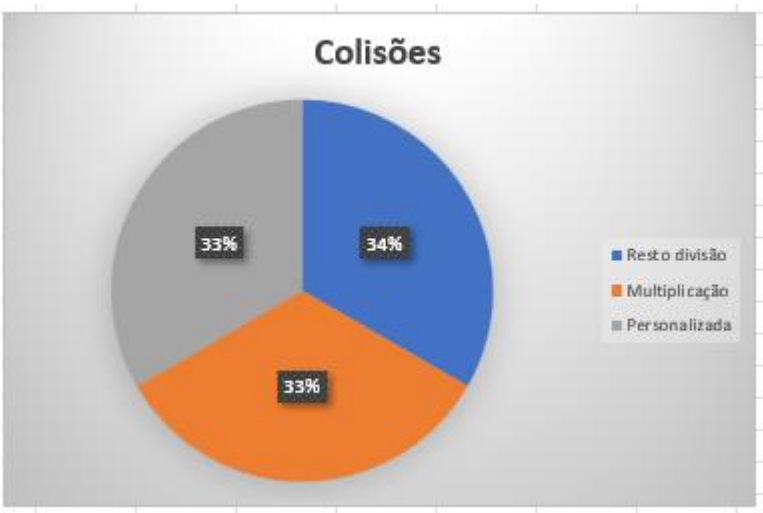
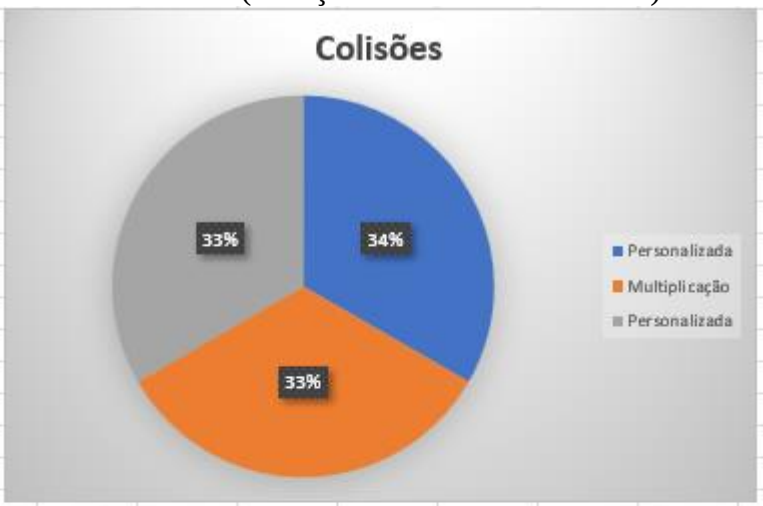


Tabela Hash de tamanho 40000 (inserção de 5 milhões de dados)



Além disso, nos cenários em que a função personalizada tem número menor de colisões, o tempo de execução tende a ser maior por conta da eventual necessidade de execução de duas funções hash, ao invés de uma.

3. Busca

Para a realização de busca na tabela hash, a função hash personalizada não foi eficiente em relação as outras duas, isso se dá por conta de que não se sabe qual função hash foi executada para inserir o elemento. Logo, devemos percorrer a lista encadeada da posição retornada pela função hash do resto da divisão, caso não seja encontrado o valor, devemos realizar a busca na lista encadeada da posição retornada na posição hash multiplicação.

Tabela hash de tamanho 10000 (busca em 100000 dados)

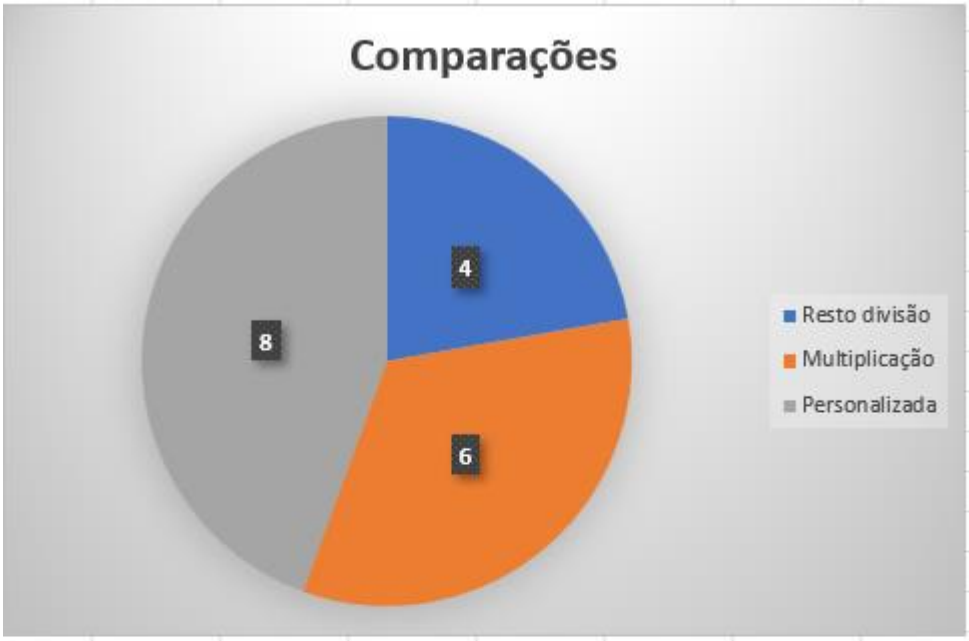


Tabela hash de tamanho 10000 (busca em 5 milhões de dados)

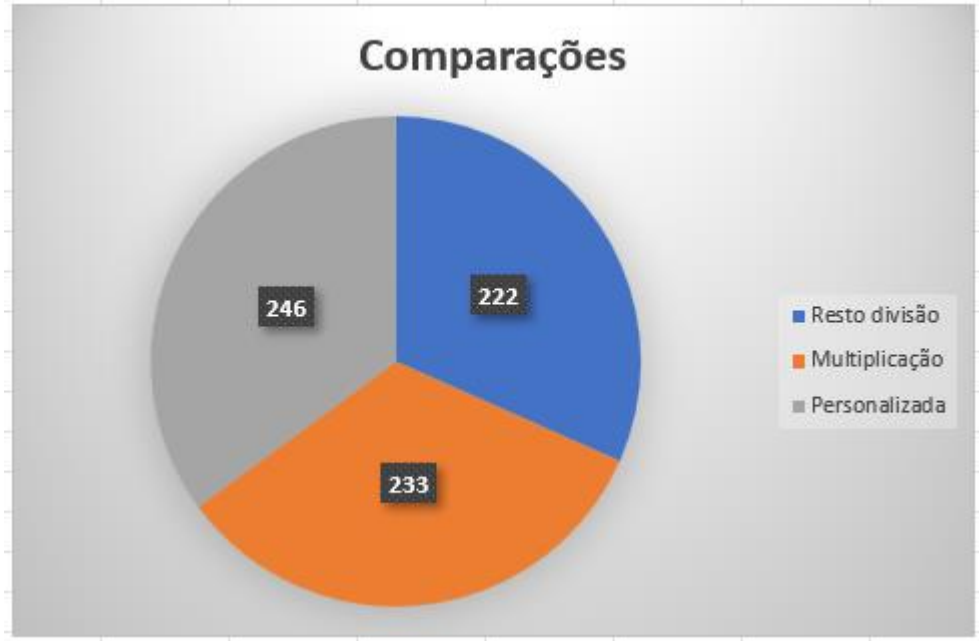
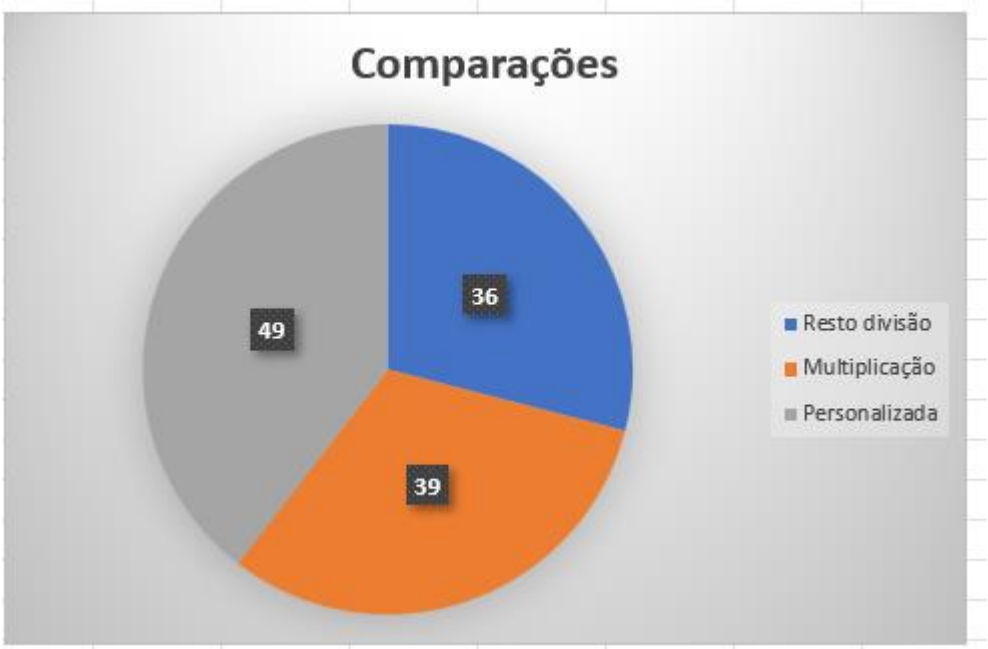


Tabela Hash de tamanho 50000 (busca em 5 milhões de dados)



Com muita sorte, o desempenho da busca usando a função hash personalizada, pode se equiparar as outras, no entanto, isso não é garantido.

Tabela Hash com tamanho 40000 (busca em 5 milhões de dados)

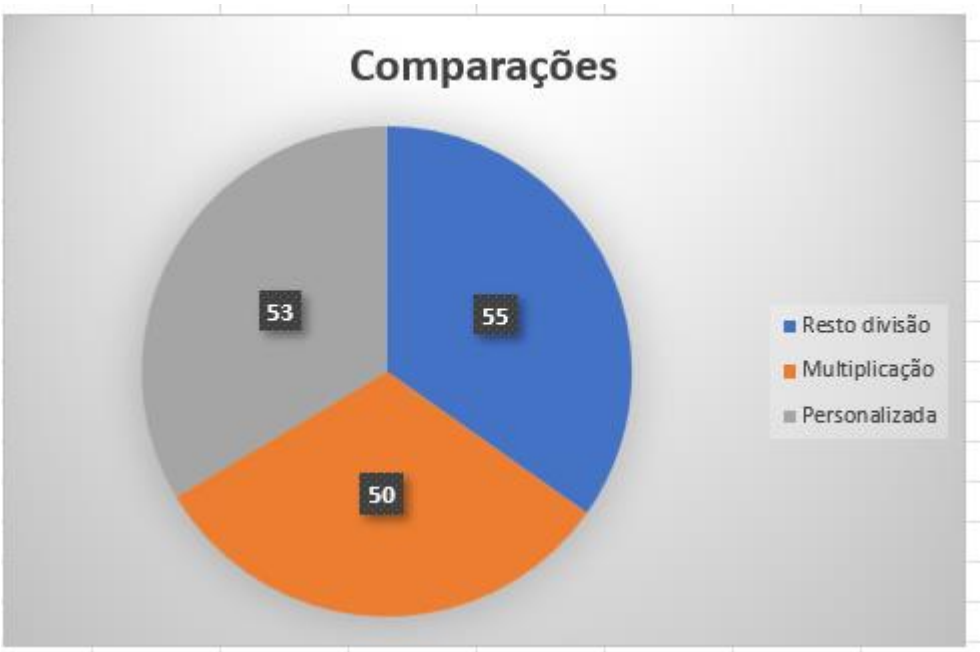


Tabela Hash de tamanho 40000 (busca em 1 milhão de dados)

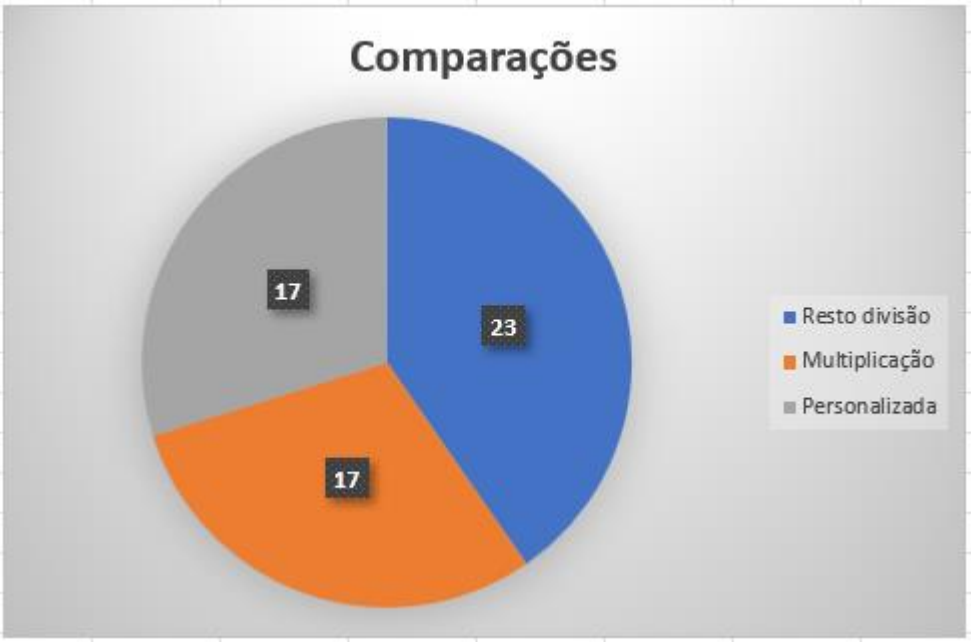
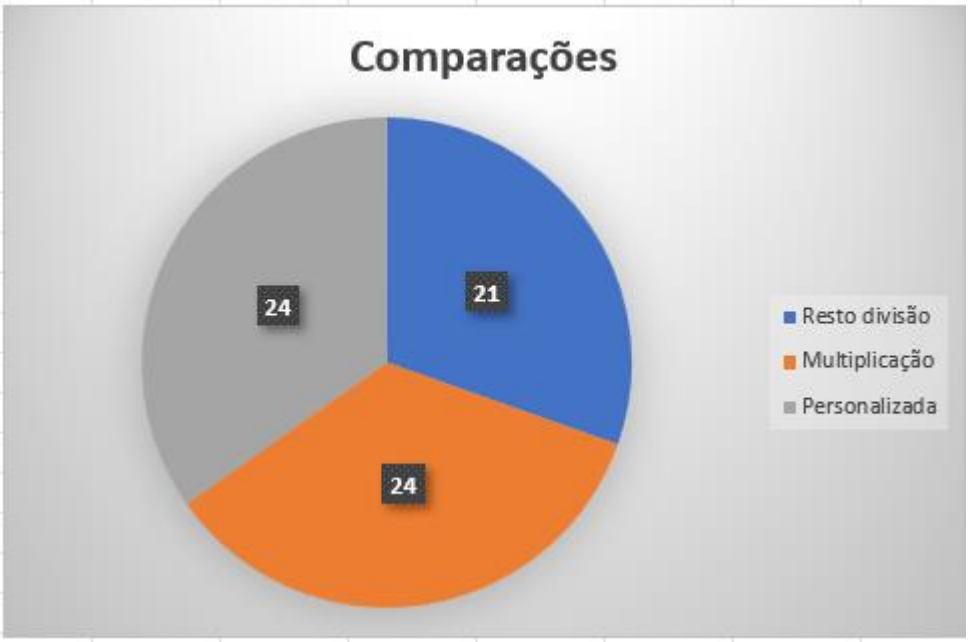


Tabela hash de tamanho 20000 (busca em 1 milhão de dados)



4. Conclusão

Para a realização de inserção na tabela hash, a função hash de resto da divisão e multiplicação tiveram desempenhos parecidos, no que se diz a colisões e tempo de execução. A função hash por multiplicação, pode ter um menor número de colisões por conta de que um número de 9 dígitos ao ser multiplicado por um valor tende a ter um valor diferente de outro número de 9 dígitos ao ser multiplicado por este mesmo valor. Já a função hash personalizada, tende a ter um melhor desempenho em cenários de inserção de menores massas de dados e em tabelas de maior capacidade, no entanto, o tempo de execução pode ser maior por eventualmente ter que executar duas funções hash. A função que utiliza do resto da divisão, por sua vez, pode ter seu maior ganho no quesito tempo de execução, uma vez que é simples executar a operação.

Função hash	Tamanho tabela hash	Número de dados	Colisões	Tempo de execução
Resto divisão	50000	20000	3517	246
Multiplicação	50000	20000	3474	323
Personalizada	50000	20000	1002	262
Total				3

Função hash	Tamanho tabela hash	Número de dados	Colisões	Tempo de execução
Resto divisão	50000	500000	450001	1789
Multiplicação	50000	500000	450001	1662
Personalizada	50000	500000	450000	1761
Total				3

Função hash	Tamanho tabela hash	Número de dados	Colisões	Tempo de execução
Resto divisão	10000	20000	11352	232
Multiplicação	10000	20000	11298	266
Personalizada	10000	20000	10382	243
Total				3

Por outro lado, essa função hash personalizada além de nem sempre ser tão eficiente para inserção, temos problemas ao realizar a busca em elementos, por conta de que é impossível saber qual foi a função hash utilizada para inserir o elemento, logo, devemos executar uma função hash, percorrer a lista encadeada e caso não seja encontrado o item, devemos executar outra função hash e percorrer outra lista encadeada, ocasionando em um número maior de comparações e um tempo de execução maior. O melhor cenário para busca dessa função, é apenas não ser pior do que uma das outras funções hash.

Função hash	Tamanho tabela hash	Número de dados	Comparações	Tempo de execução
Resto divisão	20000	100000	2	1
Multiplicação	20000	100000	2	2
Personalizada	20000	100000	5	3
Total				3

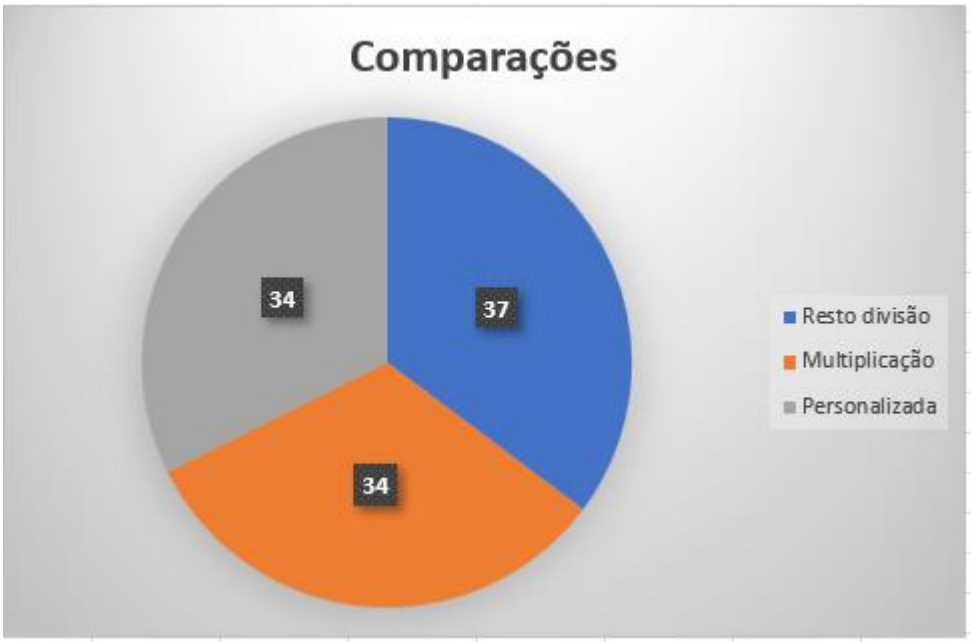
Função hash	Tamanho tabela hash	Número de dados	Comparações	Tempo de execução
Resto divisão	50000	5000000	36	4
Multiplicação	50000	5000000	39	5
Personalizada	50000	5000000	49	8
Total				3

Função hash	Tamanho tabela hash	Número de dados	Comparações	Tempo de execução
Resto divisão	10000	500000	37	1
Multiplicação	10000	500000	34	1
Personalizada	10000	500000	34	2
Total				3

Já a função hash multiplicação, foi um pouco melhor do que a função resto da divisão no cenário de busca pela baixa probabilidade de dois números de 9 dígitos ao serem multiplicados por um valor terem valores iguais, ocasionando em menos colisões que resultam em menos comparações para encontrar um valor.

De um modo geral, podemos concluir que as funções por resto da divisão e multiplicação terão um desempenho parecido, no que se diz a colisões e tempo de execução. Já a função personalizada, nem sempre vai ser melhor na inserção e tende a ter um desempenho pior na busca.

Cenário de busca – hash personalizado



Cenário de busca – hash personalizado