Universidade Federal da Bahia
Ciência da Computação - 2023.1
Estruturas de Dados e Algoritmos II

Avaliação experimental de métodos hashing

Elaborado por Igor Dantas Gomes Franca

Salvador, Bahia
2023

INTRODUÇÃO

Hashing é uma técnica amplamente utilizada para indexação, recuperação e armazenamento eficiente de dados, sendo aplicada em bancos de dados, algoritmos de busca e processamento de grandes volumes de informações. A eficácia e a eficiência dos métodos de hashing desempenham um papel crítico no desempenho e na escalabilidade desses sistemas.

A avaliação experimental dos métodos de hashing a serem citados busca quantificar o desempenho dessas técnicas em diferentes cenários. Métricas como o fator de carga, tamanho de uma tabela e dispersão dos dados são comumente utilizadas para avaliar e comparar o desempenho dos algoritmos hashing. Com base nessas métricas, é possível determinar quais métodos são mais adequados para um determinado conjunto de dados ou aplicação específica.

O objetivo deste relatório é apresentar uma análise detalhada da avaliação experimental de métodos de hashing, destacando sua importância, as principais métricas utilizadas, os desafios enfrentados, as abordagens comumente empregadas para realizar tais avaliações e comparações com outras pesquisas já realizadas. Além disso, serão discutidos os critérios para a seleção de métodos de hashing, bem como as técnicas utilizadas para realizar experimentos comparativos. Busca-se, através desse experimento, responder às seguintes questões:

- Quais as diferenças entre os desempenhos dos diferentes métodos?
- Em quais circunstâncias um método se sobressai pelo outro?
- O desempenho esperado, deduzido teoricamente, reflete o que se observa na prática?
- Estudos similares anteriores, conduzidos por outros autores, estão em consonância com o que se observou no trabalho?
- Quais as explicações para as possíveis diferenças encontradas?

Por fim, é importante ressaltar que a avaliação experimental de métodos de hashing desempenha um papel fundamental no avanço da área de ciência da computação e no desenvolvimento de soluções eficientes e escaláveis. Ao compreender os desafios e as abordagens relacionadas à avaliação experimental de métodos de hashing, é possível tomar decisões informadas na seleção e implementação dessas técnicas, contribuindo para o progresso contínuo nesse campo de estudo.

MÉTODOS

Neste experimento, foram utilizados os seguintes algoritmos de hashing como objetos de estudo:

1. Encadeamento explícito com o uso de ponteiros usando alocação estática de memória
2. Endereçamento aberto com sondagem linear

3. Endereçamento aberto com duplo hashing
4. Endereçamento aberto com sondagem quadrática

Dentre os quatro algoritmos listados, o quarto foi deixado como livre escolha pela determinação do trabalho. No caso, o algoritmo hashing com sondagem quadrática foi escolhido e é citado juntamente aos algoritmos restantes por Cormen *et al.* para endereçamento aberto. Diferentemente do restante, o encadeamento explícito utiliza espaços além da própria tabela. Como seu próprio nome define, utilizam-se ponteiros para estender endereços presentes em uma tabela. O restante dos algoritmos ficam no escopo do endereçamento aberto, definido por Cormen *et al.*, onde tabelas são criadas com espaços absolutos, sem encadeamento explícito com ponteiros alocados estaticamente na memória.

A fim de analisar a diferença de desempenho entre esses algoritmos ao acessar a tabela em busca de um registro, foram definidas métricas que possam parametrizar os resultados obtidos no experimento.

Todos os algoritmos foram testados, por inserção e busca, com diferentes valores para o fator de carga ($\alpha = \frac{n}{m}$, em que m indica o tamanho da tabela hashing e n o número de registros armazenados na tabela). Posteriormente, será analisado como o fator de carga pode inferir os resultados de busca do algoritmo, pois indica, na teoria, um número maior de colisões (números que colidem durante a inserção na tabela) a medida que esse valor aproxima-se de um (Liu, Dapeng & Xu, Shaochun. 2015).

Através da linguagem C (MinGW GCC-6.3.0-1), foi implementado um gerador de números aleatórios a serem inseridos na tabela hash de forma que estivessem bem distribuídos, analogamente, para que tivessem diferentes valores na função hash, ao dividir por m.

Nesse estudo, o tamanho m da tabela também será explicitado nos gráficos como *TABLE_SIZE*, nome da variável nos códigos em C.

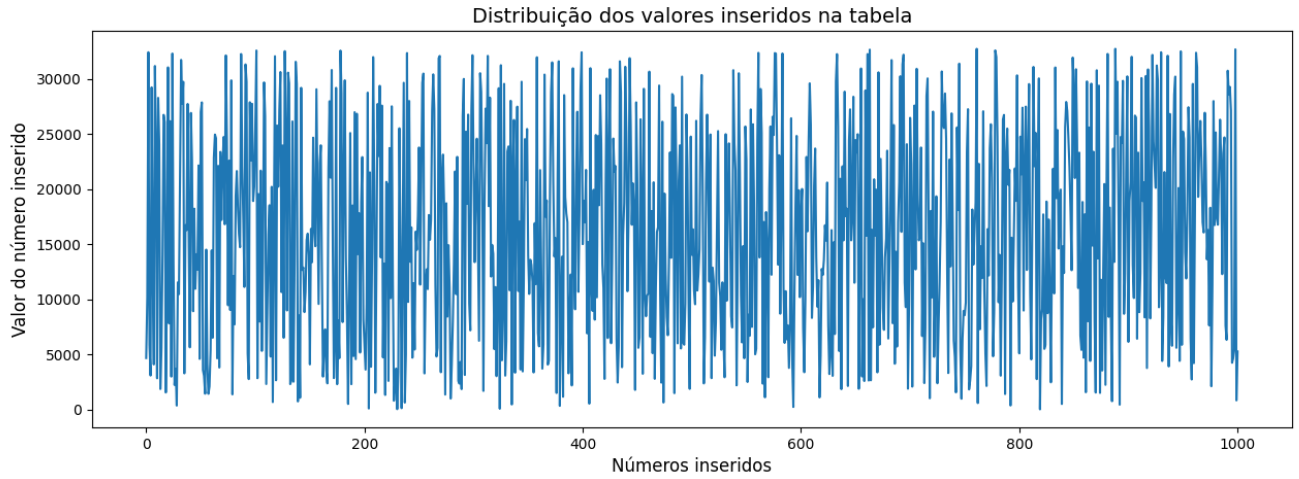

Figura 1: Exemplo de distribuição de *TABLE_SIZE* valores inseridos na tabela

Na figura abaixo, é possível observar a ampla distribuição dos valores gerados entre quase todos os m índices da tabela para todas as funções, pois sua divisão por m indicará o primeiro índice a ser comparado na tabela hash. Esse cenário evita a criação de padrões e tendências no teste, gerando valores aleatórios que diferem tanto no valor em si, quando no seu resto de divisão por m (Cormen *et al.*)

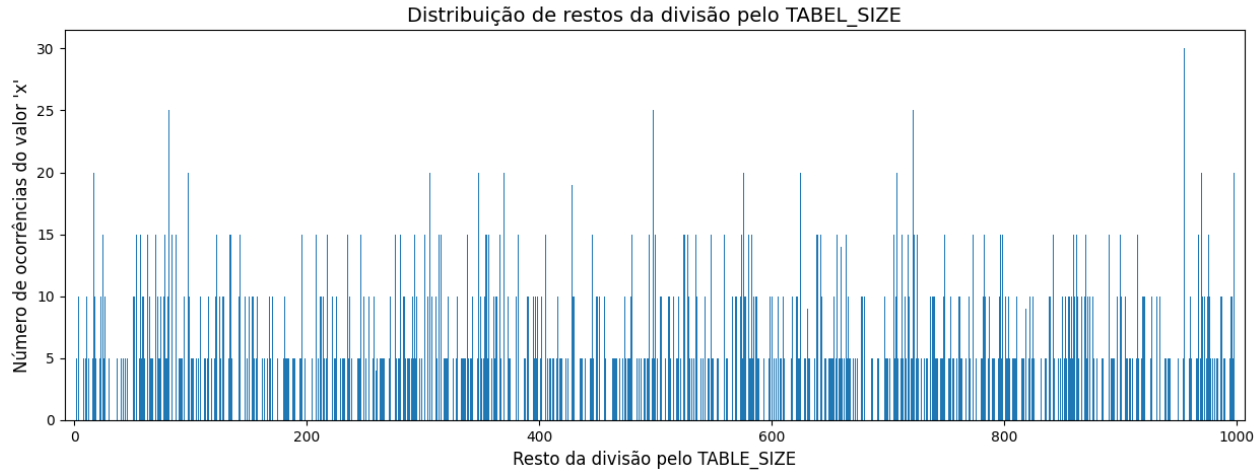

Figura 2: Distribuição dos restos das divisões por TABLE_SIZE

A avaliação de desempenho dos algoritmos será pela média de acessos para todos os valores que foram inseridos em uma tabela hash para um determinado fator de carga α, ao invés de ser medido através do tempo de execução. Torna-se uma métrica mais objetiva para medir a complexidade dessas soluções e independente do contexto de hardware.

Todos os valores obtidos são armazenados em arquivo e importados em um código Python (3.8.5) para manipulação dos dados e análise gráfica e analítica nas circunstâncias citadas anteriormente.

EXPLICAÇÃO DOS EXPERIMENTOS

Propositalmente, a escolha do valor de m para um número primo (de início, 997) foi feita pelo seguinte:.

Deve-se saber que um número inteiro que possui um divisor comum com o tamanho m da tabela irá retornar um valor múltiplo de m na função hash. Nisso, observa-se que certas buscas podem acabar entrando em *loop's* enormes, senão inacabáveis, ao buscar um valor, o que também é chamado de *clusterização primária* na abordagem do algoritmo de sondagem linear (Cormen *et al.*). Para resolver tal questão, simplesmente usa-se um número m primo, visto que não será divisor/múltiplo comum de nenhum número a ser inserido da tabela, evitando este problema.

Com o intuito de promover mais questionamentos através do experimento, outros valores de m (também primos) serão utilizados, podendo gerar diferentes resultados, assim como observado em outras pesquisas (Liu, Dapeng & Xu, Shaochun. (2015)).

O experimento foi realizado com testes individuais e características diferentes de armazenamento a depender do algoritmo em questão. Houve a necessidade de implementar uma lista com encadeamento através de ponteiros alocados estaticamente para a aplicação do algoritmo de encadeamento explícito. Já para o restante, bastou alocar uma lista de inteiros. Ambas as situações presumem uma tabela inicial de tamanho m.

No primeiro algoritmo usado nos testes, com encadeamento explícito de chaves através de ponteiros, é alocada na memória principal uma tabela de tamanho m do tipo *Node*, onde armazena um número inteiro e um ponteiro para um próximo registro também do tipo *Node*. Ao obter o índice na tabela através da função hash primária;

$$h_1(k) = k \bmod m \text{ ,}$$

Um *Node* será criado com o valor da chave k, e será colocado no índice retornado pela função $h_1(k)$, caso ali esteja vazio. Senão, será apontado pelo último *Node* inserido na mesma cadeia daquele índice. No caso de busca, seguirá o mesmo caminho.

Nos algoritmos de sondagem linear, quadrática e hashing duplo, utiliza-se a alocação de uma tabela de tamanho m do tipo inteiro (sem encadeamento). Para todos, utiliza-se a função hash primária. Nos casos de inserção e busca para sondagem linear ou quadrática, temos as seguintes abordagens para conflitos no índice retornado pela hash primária:

$h(k, i) = (h_1(k) + i) \bmod m$, para sondagem linear;

$h(k, i) = (h_1(k) + i^2) \bmod m$, para sondagem quadrática;

para $i = 0, 1, \ldots, m - 1$. Para o duplo hashing, temos, além da função primária, em caso de conflitos, a seguinte função hash secundária aplicada:

$$h_2(k) = 1 + (k \bmod (m - 1))$$

Outras abordagens podem ser feitas para a função secundária, mas pela simplicidade do experimento, foi adotado dessa forma, garantindo que todos os elementos da tabela sejam acessados ($h_2(k)$ e m são primos entre si).

Para cada um dos quatro algoritmos, foi realizada a seguinte rotina de execução:

1. Aloca-se uma tabela de tamanho m estaticamente na memória

2. São gerados n números aleatórios (inicialmente “um”)
3. Aplica-se o algoritmo de inserção para todos os n números gerados
4. Aplica-se a função de procura para todos os n números, e incrementa um contador c, inicialmente igual a um, a quantidade de acessos até encontrar este valor
5. É armazenado o valor c (número total de busca para n chaves) em uma lista
6. Todos os valores da tabela são excluídos
7. Repete-se o ciclo até que $n = m$

Tabela 1: Rotina de testes

Tipo de endereçamento	Solução para colisão	Tamanho da tabela (primo)	Valor da chave	Quantidade de chaves geradas	Tamanho da tabela	Fator de Carga
Fechado	Encadeamento explícito	997 983 991 … … … 2339 2341 2347	$1 \leq x \leq 32744$	1 2 3 … … … 995 996 997	997	0,001 % … … … 100%
Aberto	Sondagem Linear Duplo Hashing Sondagem Quadrática					

Tabela 2: Combinação de testes

Através dessa execução, obtemos a média de acessos cada cada série de $1 \leq n \leq m$ números gerados aleatoriamente e inseridos na tabela hashing em cada contexto de algoritmo, logo, com o fator de carga variando em $0 < \alpha \leq 1$. O fator de carga será um parâmetro utilizado nos resultados finais durante a análise de desempenho.

Seguidamente, é necessário a comparação com outros autores, outras pesquisas e deduções teóricas. A complexidade de cada algoritmo será analisada e comparada com fundamentos de outros autores, assim como os resultados e suas possíveis diferenças também serão comparadas com outras pesquisas realizadas previamente. A priori, será observado:

1. Porcentagem de colisões por número de inserções (encadeamento explícito)
2. Média de acessos por fator de carga
3. Número de chaves com colisão por fator de carga
4. Média de acessos por tamanho de tabela (encadeamento explícito)

DISCUSSÃO DOS RESULTADOS

1. ENDEREÇAMENTO ABERTO

- Média de acessos

Uma primeira análise do experimento pode ser feita na média de acessos para todas as chaves inseridas por fator de carga. O corolário apresentado por Cormen *et al.* (11.7) em “Algoritmos: Teoria e Prática” mostra que, dado uma tabela hash com endereçamento aberto, o número médio de sondagens é, no máximo, $\frac{1}{1-\alpha}$.

É evidente que há uma tendência das funções hash à função citada no livro, embora exista uma descontinuidade do salto à depender da função.
Dada a fórmula e sua prova, foi posto em comparação com os algoritmos de endereçamento aberto. A priori, **foi possível visualizar uma média maior e mais próxima à função máxima citada no método de sondagem linear**, tendo um salto consideravelmente maior que os métodos hashing duplo e quadrático à partir do entorno de 70%, como pode-se observar na figura 4.

- Tamanho da tabela

Em busca de comparar os resultados dos experimentos com outros já feitos anteriormente (Liu, Dapeng & Xu, Shaochun. (2015)), também foi posto em comparação o tamanho da tabela m, variando seu tamanho em dois números primos: 997 e 4129. É notório que **não houve mudança significativa na média de acessos ao preencher uma tabela hash de tamanho superior**.

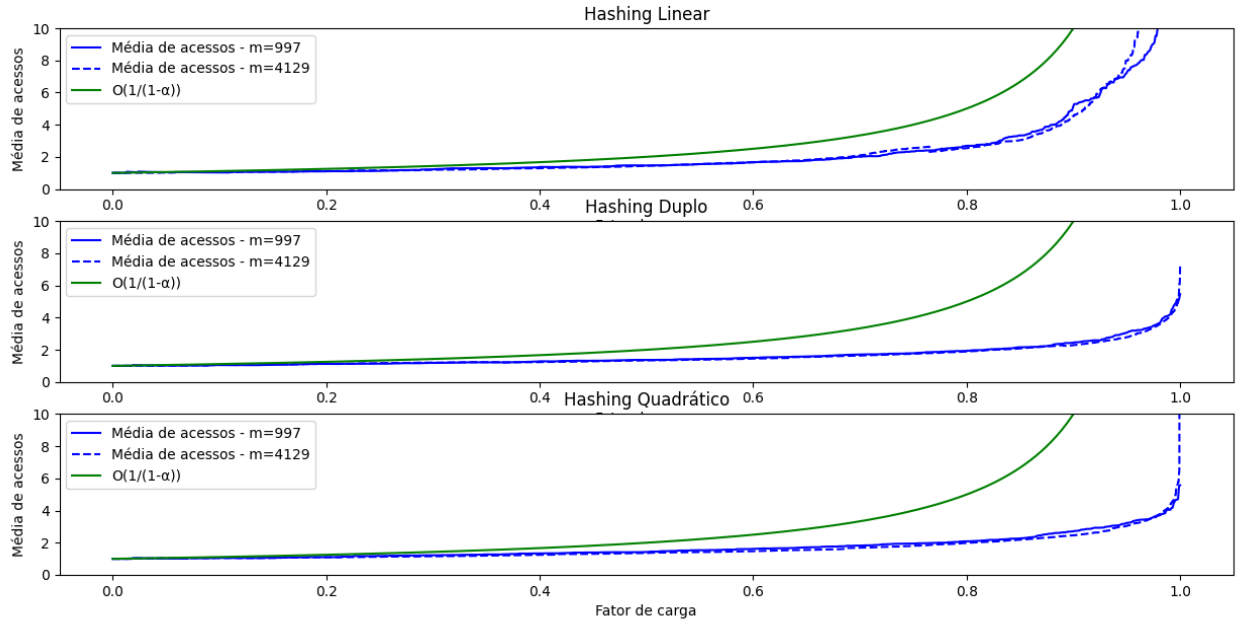

Figura 3 - Média de acessos por cada algoritmo de endereçamento aberto separadamente

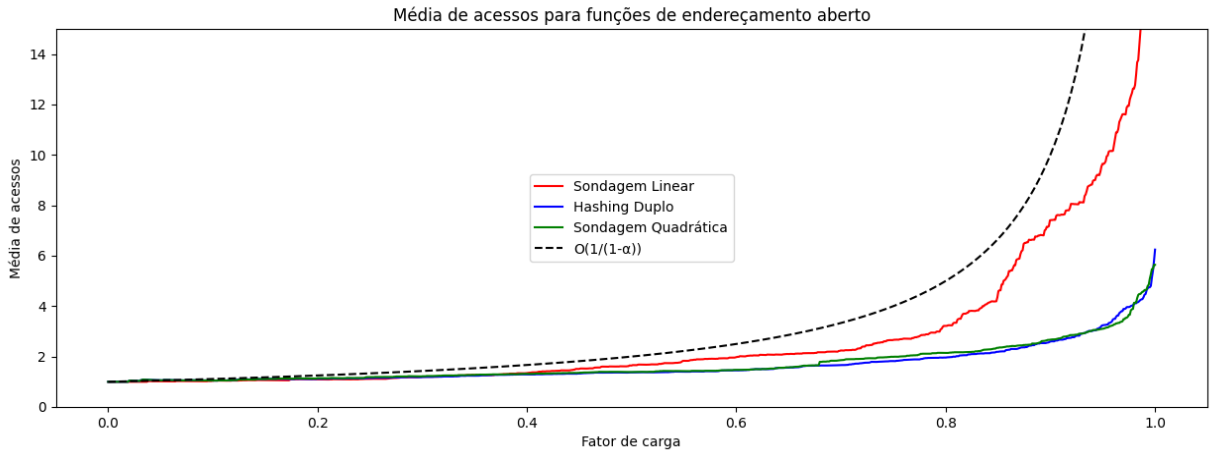

Figura 4 - Média de acessos por cada algoritmo de endereçamento aberto

Apesar dos resultados visíveis na figura 3, outros testes demonstraram algo intuitivamente óbvio: **a inserção dos mesmos elementos, porém, em uma tabela de tamanho superior, implica a redução do número médio de acessos**. Como o fator de carga não tem o mesmo valor para a mesma inserção total em tabelas de tamanhos diferentes, pelos resultados analisados anteriormente, observa-se que haverá melhor desempenho conforme menor for o fator de carga. A partir do entorno de 70%, torna-se visível o salto no número médio de acessos.

- Número de chaves com colisão

Em acordo com as pesquisas realizadas nos experimentos de outros autores, como citado acima, a porcentagem de colisão para a porcentagem de chaves é assonante com resultados anteriores, afirmando que **a taxa de colisão não muda à depender do algoritmo utilizado**, dado o cenário de chaves muito bem distribuídas num espaço suficientemente grande. Como demonstra a figura abaixo, há uma disparidade insuficientemente considerável entre as funções.

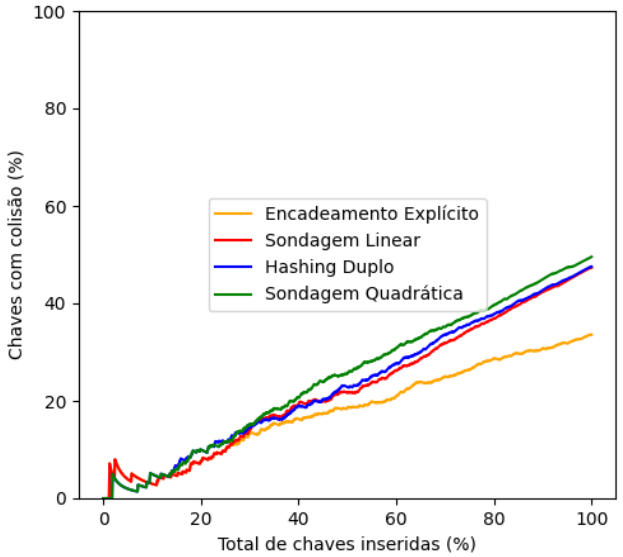

Figura 5 - Porcentagem de chaves com colisão ao longo das inserções

2. ENDEREÇAMENTO FECHADO

- Média de acessos

Para o algoritmo de endereçamento fechado, com encadeamento, os mesmos autores (Cormen *et al.*) provam que, em tabelas hash com encadeamento para problemas de colisão, o caso médio de acessos para uma busca com sucesso é:

$\Theta(1 + \alpha)$.

Foi observado um desempenho melhor que o deduzido teoricamente para caso médio, possívelmente por conta da dispersão dos valores inseridos na tabela, causando maior uniformidade na distribuição dos valores e nos índices da tabela.

- Tamanho da tabela

Assim como para algoritmos de endereçamento aberto, percebe-se que também não houve diferença significativa de desempenho para valores m maiores

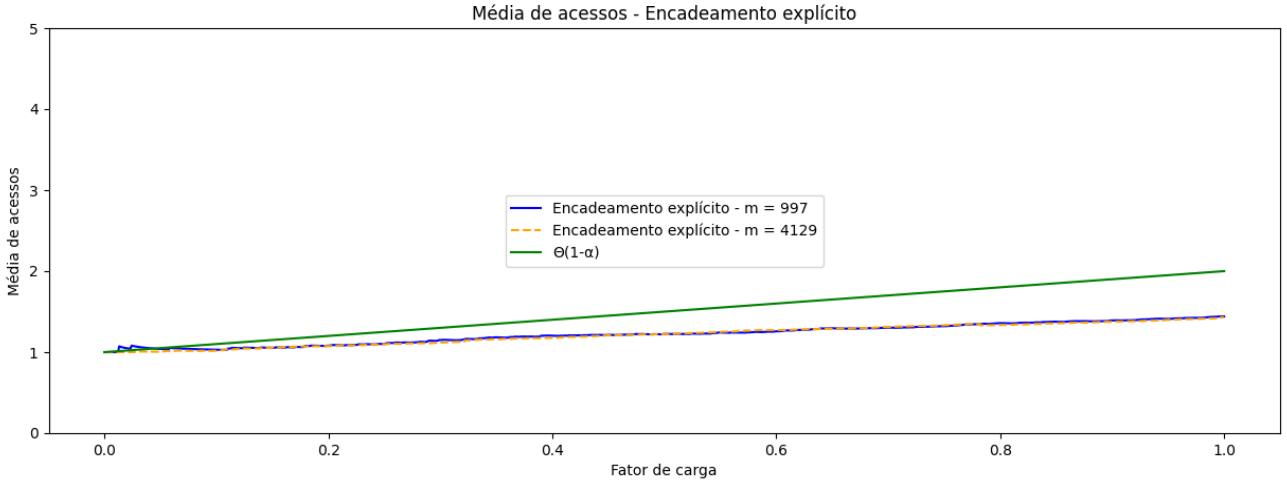

Figura 5 - Média de acessos para encadeamento explícito

No entanto, é importante destacar a diferença no caso de uma tabela hash maior, com o mesmo número de inserções (997). É possível perceber que a diminuição na média de acessos já é significante à partir das primeiras 200 inserções.

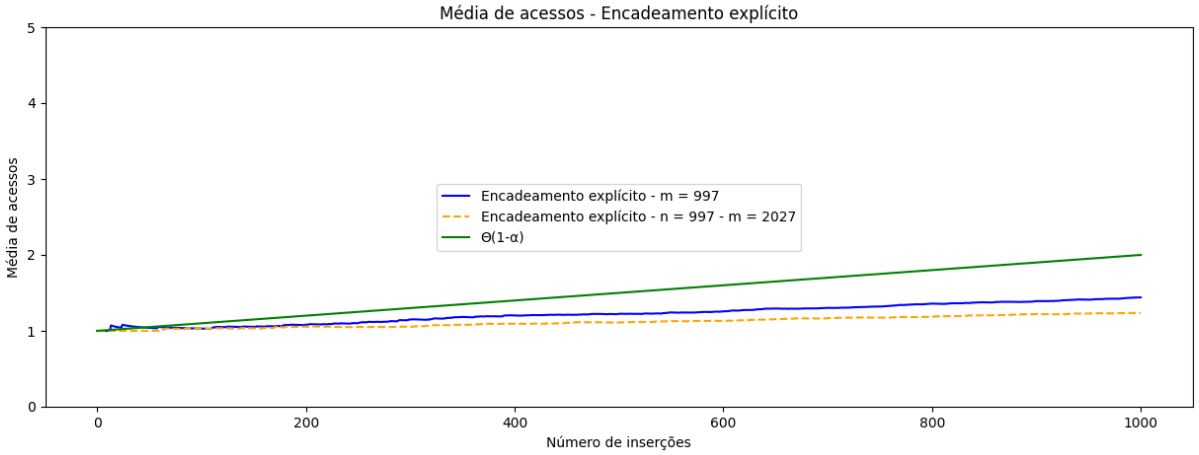

Figura 6 - Média de acessos pelo número de inserções

CONCLUSÕES

Através dos testes experimentais realizados acima e dos resultados obtidos, é possível atestar algumas seguintes conclusões sobre o uso dos algoritmos de hashing analisados:

- Para um cenário semelhante de valores a serem inseridos em uma tabela hash, para algoritmos de endereçamento aberto, o hashing linear torna-se uma opção trabalhosa à depender da aplicação
- Os algoritmos de hashing duplo e quadrático tornam-se mais sustentáveis em relação à sondagem linear
- É recomendado manter um fator de carga inferior a 70% em algoritmos de endereçamento aberto para desempenho satisfatório

É evidente que a melhor escolha de algoritmo para resolução de conflitos depende da aplicação. Num contexto de memória limitada, os métodos de hashing duplo e quadrático servem um desempenho satisfatório, com média de acessos menor que 2, para fator de carga $\alpha < 0.7$. No caso citado no livro Algorithm Design and Applications (GOODRICH, M., TAMASSIA, R.(2015)) sobre o hashing quadrático, não foi observado a questão do clustering secundário em desavanço com o hashing duplo, pois houve um desempenho consideravelmente semelhante. Pela característica da distribuição dos valores inseridos, é possível que tenha evitado uma dissonância maior entre os dois algoritmos.

Em concordância com Liu, Dapeng & Xu, Shaochun. (2015), observa-se que o desempenho de uma tabela hash está relacionado com o seu tamanho m, e quanto maior esse tamanho, menor será a média de acessos para n chaves inseridas.

O método de endereçamento fechado torna-se uma opção mais estável, embora seja uma aplicação mais sensível pela questão de uso de memória.

REFERÊNCIAS

- Liu, Dapeng & Xu, Shaochun. (2015). Comparison of Hash Table Performance with Open Addressing and Closed Addressing: An Empirical Study. International Journal of Networked and Distributed Computing. 3. 60. 10.2991/ijndc.2015.3.1.7.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Algoritmos: Teoria e Prática. 3a edição. Elsevier, 2012. ISBN 9788535236996
- GOODRICH, M.; TAMASSIA, R. Linear Probing. Algorithm Design and Applications, Wiley, p. 200-203, 2015.