Análise de Tabelas Hash: Estrutura, Eficiência e Aplicações

Ícaro Cardoso da Silva, Yuri Bacelar, Breno Ferreira, Pedro Henrique
Centro Universitário Unieuro, Águas Claras, DF, Brasil
Emails: {icarocardoso299, pedrodsousa10, brenoferreiraa1905, yuribacelar1605}@gmail.com

***Resumo*—Este artigo apresenta uma análise da estrutura de dados tabela hash, abordando seu funcionamento, métodos de resolução de colisões, e aplicações em sistemas computacionais modernos. São exploradas diferentes técnicas de implementação, complexidade computacional e casos práticos de uso. Os resultados indicam que, apesar de limitações como colisões e dependência da função hash, as tabelas hash oferecem desempenho superior em operações de inserção, busca e remoção. O estudo reforça sua relevância em linguagens de programação e bancos de dados.**

***Index Terms*—Tabela hash, estrutura de dados, colisão, função hash, complexidade computacional.**

I. Introdução

Tabelas hash são estruturas de dados amplamente utilizadas para armazenamento e recuperação rápida de informações. Por meio de uma função hash, é possível mapear chaves a índices de um vetor, o que permite acesso quase constante em tempo médio (O(1)) [1]. Essa eficiência faz com que as tabelas hash sejam ideais para aplicações como dicionários, caches e sistemas de indexação.

No entanto, sua eficácia depende da qualidade da função hash e das estratégias adotadas para resolução de colisões. O objetivo deste artigo é investigar essas estratégias e discutir a aplicabilidade das tabelas hash em contextos reais.

II. Metodologia

A metodologia adotada baseou-se em experimentação computacional com foco na comparação de desempenho entre diferentes estratégias de implementação de tabelas hash. As etapas seguiram o seguinte processo:

1) Implementação de tabelas hash em Python e C++;
2) Utilização de três funções de dispersão: modular, multiplicativa e universal;
3) Aplicação de dois métodos de tratamento de colisões: encadeamento separado e endereçamento aberto (linear, quadrático e duplo hashing);
4) Execução dos testes com dois conjuntos de dados:
   - Dados sintéticos gerados aleatoriamente (100.000 registros);
   - Dados reais: listas de nomes e palavras extraídas de repositórios públicos;
5) Coleta das métricas de tempo de inserção, tempo de busca e taxa de colisões;
6) Execução dos testes em ambiente controlado (máquina com processador Intel i5, 8 GB de RAM).

III. Resultados

Os resultados obtidos a partir dos testes são apresentados na Tabela I. Ela resume os tempos médios de busca, taxa de colisões e eficiência relativa de cada abordagem.

Tabela I
Comparação entre técnicas de resolução de colisão

Técnica	Tempo médio (ms)	Colisões (%)	Eficiência
Encadeamento	0.45	12%	Alta
Aberto Linear	0.78	25%	Média
Aberto Quadrático	0.69	18%	Alta
Duplo Hashing	0.82	27%	Média

Verifica-se que o encadeamento separado apresenta menor tempo médio de busca em cenários de alta ocupação (¿75%) [2]. O endereçamento aberto sofre degradação proporcional ao número de colisões.

IV. Conclusão

Tabelas hash continuam sendo ferramentas essenciais para a eficiência de algoritmos e sistemas. A escolha adequada da função hash e da estratégia de colisão pode melhorar significativamente o desempenho. Como trabalho futuro, sugere-se o uso de funções hash criptográficas ou adaptativas em ambientes de alta variabilidade de dados. [3].

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest e C. Stein, *Introduction to Algorithms*, 3ª ed., MIT Press, 2009.
[2] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3, Addison-Wesley, 1998.
[3] R. Sedgewick e K. Wayne, *Algorithms*, 4ª ed., Addison-Wesley, 2011.
[4] A. V. Aho, J. E. Hopcroft e J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1983.
[5] M. Dietzfelbinger, "Universal hashing and k-wise independent random variables via integer arithmetic without primes," in *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, 1996.
[6] P. Larson, "Dynamic Hash Tables," *Communications of the ACM*, vol. 31, no. 4, pp. 446–457, 1988.
[7] T. Wang, "Integer Hash Function," 2007. [Online]. Disponível em: http://www.concentric.net/ Ttwang/tech/inthash.htm