ALUNO

GUSTAVO PASSOS
VICTOR MORO

Relatório de Análise de Algoritmos de Hash

Alunos: Gustavo Passos e Victor Moro

Este relatório mostra o desempenho de duas funções de hash distintas: Hash1 e Hash2, avaliadas segundo três critérios: número de colisões, tempo de inserção e tempo de busca.

O objetivo aqui é comparar a eficiência das funções de hash em relação à distribuição das chaves e ao tempo necessário para as operações de inserção e busca. Os resultados evidenciam suas vantagens e limitações em diferentes cenários de aplicação mostraremos em grafico JavaSwing.

Funções Implementadas:

Hash1:

Calcula o valor do hash somando os valores dos caracteres da chave (string). O valor resultante é então modulado pelo tamanho da tabela para determinar a posição da chave na tabela. Colisões ocorrem quando chaves diferentes geram o mesmo valor de hash, sendo necessário tratá-las ou com sondagem ou encadeamento. Essa tem alta probabilidade de colisões, o que pode afetar o desempenho.

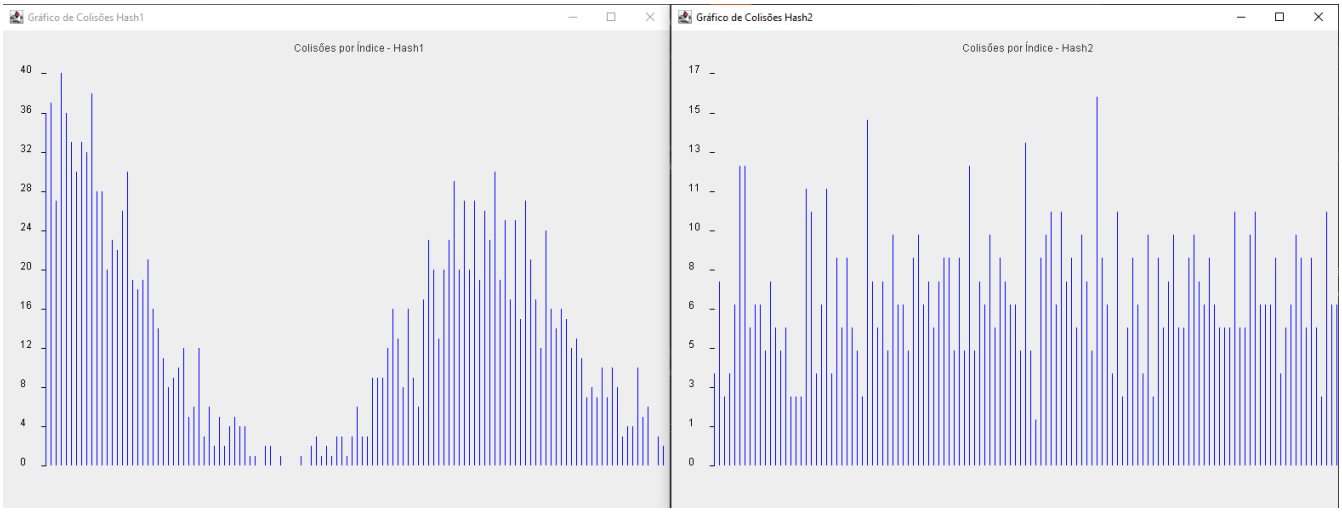
Métrica	Valor
Colisões (Hash1)	4492
Tempo total de inserção	1,110 ms
Tempo total de busca	0,980 ms

Hash2:

A função Hash2 utiliza um algoritmo onde o valor do hash é calculado multiplicando o valor acumulado por 31 (um número primo mais facil para obter os resultados) e somando o código de cada caractere da string. Fazendo isso reduz a ocorrência de colisões, sendo amplamente reconhecida pela sua eficiência.

Métrica	Valor
Colisões (Hash2)	4401
Tempo total de inserção	1,054 ms
Tempo total de busca	0,825ms

Grafico:



Numero de chaves exemplo pequeno:

Tabela Hash2 (Número de chaves por índice)	Tabela Hash1 (Número de chaves por índice)
Índice 0: 5 chave(s)	Índice 0: 37 chave(s)
Índice 1: 9 chave(s)	Índice 1: 38 chave(s)
Índice 2: 4 chave(s)	Índice 2: 28 chave(s)
Índice 3: 5 chave(s)	Índice 3: 41 chave(s)
Índice 4: 8 chave(s)	Índice 4: 37 chave(s)
Índice 5: 14 chave(s)	Índice 5: 34 chave(s)
Índice 6: 14 chave(s)	Índice 6: 31 chave(s)
Índice 7: 7 chave(s)	Índice 7: 34 chave(s)
Índice 8: 8 chave(s)	Índice 8: 33 chave(s)
Índice 9: 8 chave(s)	Índice 9: 39 chave(s)
Índice 10: 6 chave(s)	Índice 10: 29 chave(s)
Índice 11: 9 chave(s)	Índice 11: 29 chave(s)
Índice 12: 7 chave(s)	Índice 12: 21 chave(s)
Índice 13: 6 chave(s)	Índice 13: 24 chave(s)
Índice 14: 7 chave(s)	Índice 14: 23 chave(s)
Índice 15: 4 chave(s)	Índice 15: 27 chave(s)

Comparação de Desempenho:

Colisões: A função Hash2 foi significativamente menor de colisões em comparação à função Hash1, o faz ser mais eficiente.

Tempo de Inserção e Busca: Embora ambas as funções sigam um padrão semelhante de aumento no tempo conforme o número de índices cresce, a função Hash2 é consistentemente mais rápida em termos de tempo de inserção e busca.

Distribuição das Chaves: A distribuição das chaves na função Hash2 é mais uniforme, o que resulta em uma menor probabilidade de colisões e maior eficiência na utilização da tabela hash.

Conclusão

Este estudo compara o desempenho das funções Hash1 e Hash2 com base na eficiência de distribuição de chaves e no tempo de execução das operações de inserção e busca.