Relatório de análise Algoritmos Hash

João Guilherme Squaris Merlin [1]

[1]PUCPR - Pontifícia Universidade Católica do Paraná (PUCPR)
R. Imac. Conceição, 1155 - Prado Velho, Curitiba - PR, 80215-901

Resumo. *O relatório visa fornecer insights sobre como o desempenho de tabelas hash é afetado por diferentes tamanhos de tabela, funções de hash e tamanhos de conjuntos de dados.*

1. Ambiente Utilizado

• Processador: Ryzen 5 5500 • Placa Gráfica: Rx 5500xt • Memória: 16 Gb • SSD: 500 Gb • Fonte: 500W • IDE: Intellij

2. contexto de analise

Esse relatório foi feito com base no enunciado proposto. Serão avaliados 3 Funções Hash (divisão, multiplicação e dobramento), 5 vetores de tabela Hash (10, 100, 1000, 10000 e 100000) e 5 conjunto de dados (20 mil, 100 mil, 500 mil, 1 milhão e 5 milhões). Afim de garantir resultados coerentes será utilizado os mesmos conjunto de dados para função Hash. Além da inserção das funções Hash, também será avaliado a busca dos conjuntos (será feito 5 buscas para evitar resultados arbitrários)

3. Considerações e Descrição da analise

Todos os valores obtidos durante a execução dos testes (tempo e numero de colisões) será disponibilizado no mesmo repositório deste relatório. Entretanto, nem todos os valores estarão presentes neste documento, visto que, há uma grande quantidade de casos a serem analisados e o resultado obtido deles não é divergente ou relevante o suficiente para detalhar-lhos. Além disso, o número de colisões obtido durante os testes não é coerente com o contexto de analise, portanto, não terá peso nesse relatório.

Nesse sentido, a analise será feita da seguinte forma, para cada função hash, será comparado tempo de execução nos casos de menor tamanho de tabela com o conjunto de dados variados e o contrário, ou seja, será apresentado os gráficos medindo o tempo com tamanho de tabela Hash igual a 10 e conjunto de dados variados (20 mil, 100 mil, 500 mil, 1 milhão e 5 milhões) e outro com tamanho variado (10, 100, 1000, 10000 e 100000) e conjunto fixo igual a 20000. Além disso também será avaliado seu tempo médio de busca para cada caso.

4. Mudanças no decorrer do desenvolvimento

Durante testes da primeira versão deste trabalho, havia um grande problema durante a execução do código, conforme o tamanho da tabela e dos conjuntos de dados aumentavam, o tempo de execução também aumentava exponencialmente (mais de uma hora em um único caso especifico). Com base nisso, foi necessário fazer uma mudança não planejada. A fim de melhorar a performance, possibilitar os testes e aproveitar de conteúdos já estudados na disciplina, foi implementado um tratamento de colisões por arvore AVL, visto que, esse problema provavelmente, decorria do número crescente de colisões.

5. Função Hash - Resto da divisão

No caso abaixo (Figura 1), representa a variação do tempo de execução durante a mudança de Conjunto de dados na função de divisão. Nesse contexto, nota-se uma tendencia muito forte de um aumento de tempo, isso provavelmente acontece por causa do tamanho dos últimos conjuntos e do suposto aumento do número de colisões.

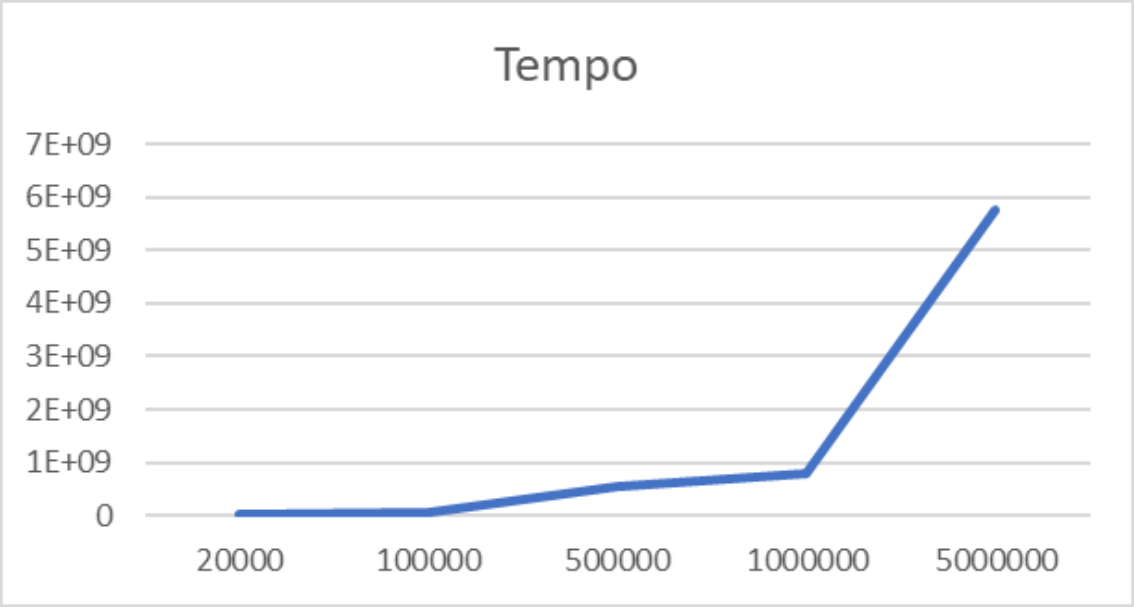

Figure 1. Função de Divisão com tamanho da tabela constante

Segue abaixo o gráfico de analise com a média dos tempos de busca referentes ao caso anterior (Figura 2):

Figure 2. Média tempo de busca da tabela de divisão com tamanho da tabela constante

O gráfico apresenta uma tendencia constante mais crescente do tempo médio de busca pelos mesmos motivos explicados anteriormente. Entretanto, essa média de buscas apresenta um resultado menor do que o esperado, isso porque o balanceamento AVL ajuda consideravelmente nos casos de colisão, uma vez que a altura da arvore esta sempre a menor possível.

Ainda na mesma função Hash de divisão, agora será analisado o contexto de conjunto de dados contantes (20000) e tamanho da tabela variável (10, 100, 1000, 10000 e 100000), respectivamente. Segue abaixo o gráfico dos tempos (Figura 3):

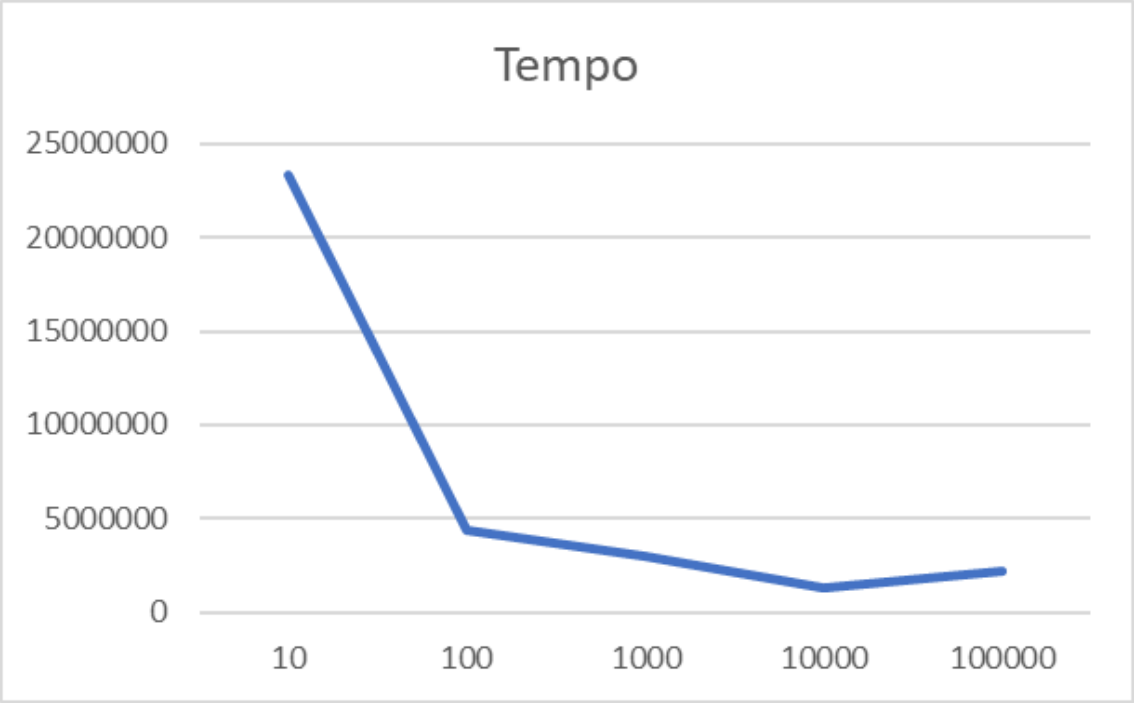

Figure 3. Função de Divisão com tamanho do conjunto de dados constante

Ao contrario do último teste, por esse apresentar um número pequeno de conjuntos que possibilita uma eficiência maior visto que o é um tamanho menor a ser passado pra tabela e por consequência, um número menor de colisões.

Por fim, temos a analise da média de buscas para esse contexto (Figura 4):

Figure 4. Média tempo de busca da tabela de divisão com tamanho do conjunto de dados constante

A queda do tempo das buscas segue uma linha muito similar ao tempo de execução deste mesmo caso, isso provavelmente acontece por causa dos mesmo motivos citados anteriormente.

6. Função Hash - Multiplicação

O caso a seguir refere a função Hash de multiplicação com a tabela Hash contante (Figura 5):

Figure 5. Função de Multiplicação com tamanho da tabela constante

É possível notar que, apesar de diferente, o gráfico apresentado apresenta um similaridade com o mesmo caso da função analisado anteriormente. E isso é uma ocorrência que irá persistir em todos os casos a serem analisados, por esse motivo, as avaliações a seguir serão feitas de forma mais breve.

Segue abaixo o gráfico da média de tempo de buscas do caso anterior (Figura 6):

Figure 6. Média tempo de busca da tabela de multiplicação com tamanho da tabela constante

Conjunto de dados constantes (Figura 7):

Figure 7. Função de Multiplicação com tamanho do conjunto de dados constante

Segue abaixo o gráfico de buscas para o caso a cima (Figura 8):

Figure 8. Média tempo de busca da tabela de multiplicação com tamanho do conjunto de dados constante

7. Função Hash - Dobramento

O caso a seguir refere a função Hash de Dobramento com a tabela Hash contante (Figura 9):

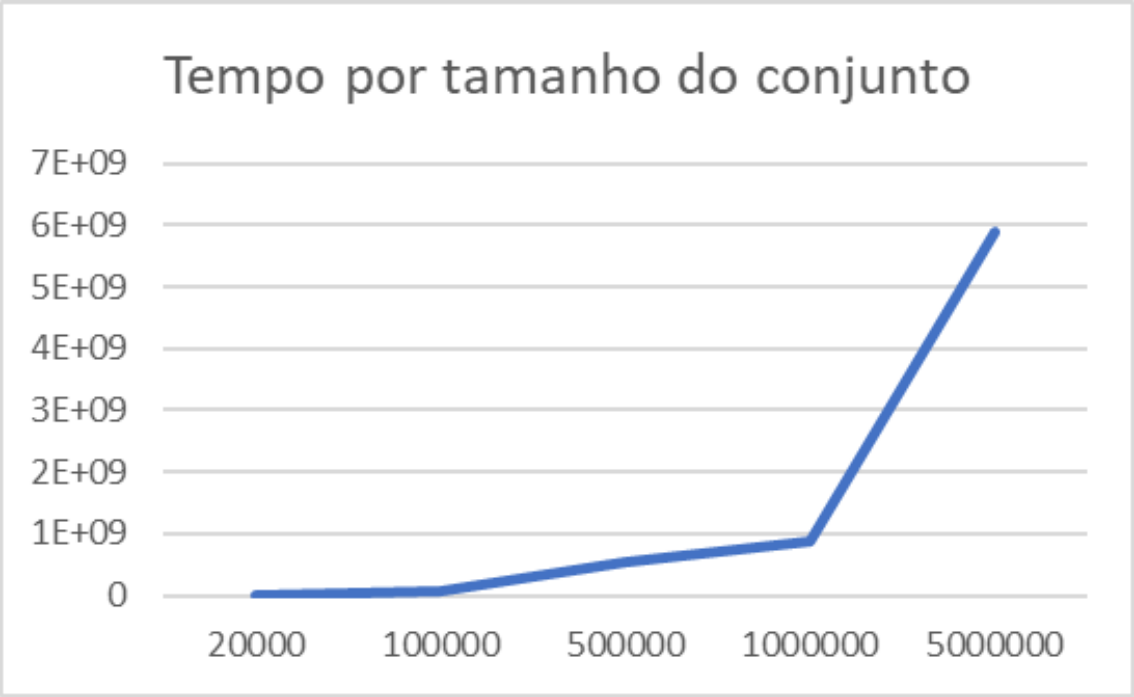

Figure 9. Função de Dobramento com tamanho da tabela constante

Segue abaixo o gráfico da média de tempo de buscas do caso anterior (Figura 10):

Figure 10. Média tempo de busca da tabela de dobramento com tamanho da tabela constante

Conjunto de dados constantes (Figura 11):

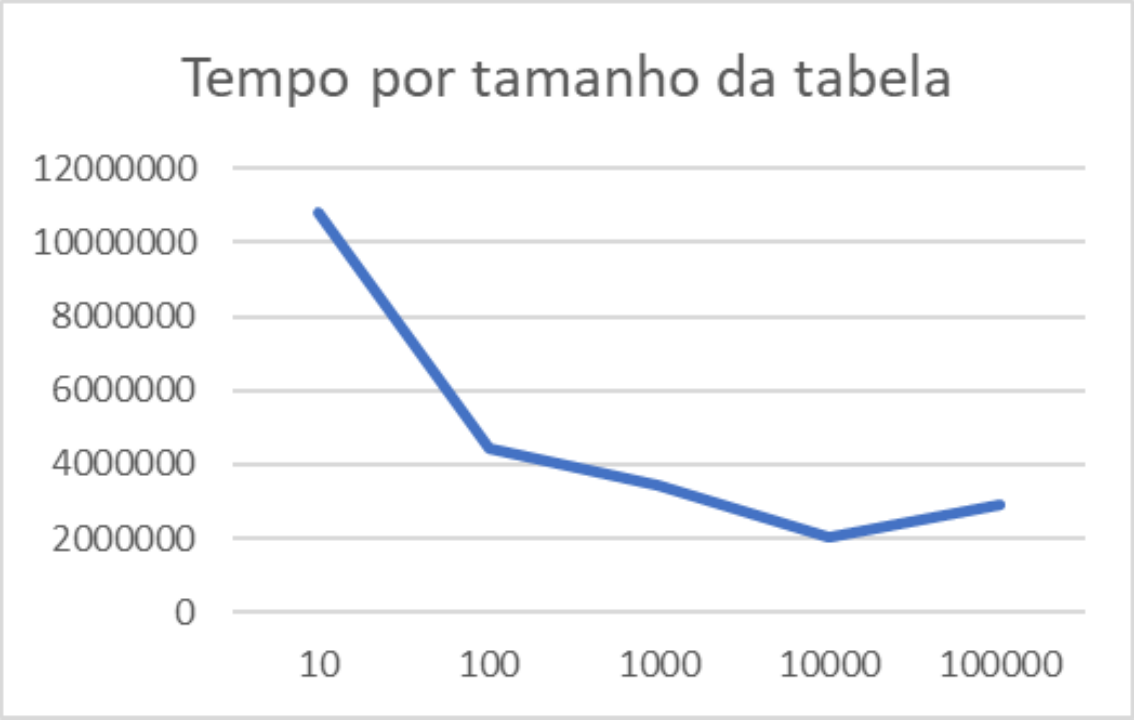

Figure 11. Função de Dobramento com tamanho do conjunto de dados constante

Gráfico referente as buscas do caso anterior(Figura 12):

Figure 12. Média tempo de busca da tabela de dobramento com tamanho do conjunto de dados constante

8. Conclusão

Apos uma analise dos contextos estudados, as fincões de tiveram um desempenho muito similar nos casos de tabela constante (tanto para tempo de execução quanto para tempo de busca), tendo uma diferencia que pode ser considerada uma taxa de erro. Entretando, a função de divisão teve um resultado melhor nos casos de conjunto de dados fixo para tempo de execução, já para o tempo de busca, o resultado também foi muito similar entre as funções.

9. Considerações finais

Link do github: https://github.com/Merlin262/AvaliacaoHash

Após a analise dos resultados, a implementação do tratamento de colisões por arvore AVL agregou significamente para a eficiência do algoritmo, o que antes da implementação era de complexidade

$$O(n^2)$$

agora é de algo mais próximo a O(n log n).

Além disso, tive uma dificuldade para formatar esse documento adequadamente, nesse sentido, deixei os gráficos devidamente referenciados.

Esta avaliação como um todo foi uma ótima oportunidade de estudar, aplicar conceitos estudados e refletir quanto ao resultado obtido. Entre tando, o tempo foi demasiado

curto para o resultado esperado, principalmente se considerarmos que há outro trabalho desta mesma disciplina e a aula referente a esses conteúdos, apesar de muito boa, foi breve.

10. Referencias

Título da Página. Instituto de Matemática e Estatística da Universidade de São Paulo, Disponível em: https://www.ime.usp.br/ pf/estruturas-de-dados/aulas/st-hash.html. Acesso em: 07 de novembro de 2023.

Título da Página. e-Disciplinas - Ambiente Virtual de Aprendizagem da Universidade de São Paulo, Disponível em: https://edisciplinas.usp.br/mod/hvp/view.php?id=3305382. Acesso em: 07 de novembro de 2023.

Autor(es) (se disponível). Gist - 11237839. GitHub Gist, Disponível em: https://gist.github.com/macroxela/11237839. Acesso em: 07 de novembro de 2023.