Tabela de espalhamento (Hashing)

Mateus Vinicios Sorgatto Giroletti

Instituto Federal Catarinense - Câmpus Videira
Bacharelado em Ciência da Computação
Estrutura de Dados I

Para a resolução proposta do trabalho final foi escolhido a lista encadeada dupla, foi implementado então a inserção da lista de nomes disponibilizados pelo professor, consulta e remoção pelo nome e a quantidade de elementos por chave. Na geração do hash foi escolhido a função modular, onde é considerado a primeira letra do nome convertido para inteiro seguindo a tabela ASCII, como na Figura 01 e fazendo o resto de divisão pelo tamanho M da tabela hash.

Figura 01 - Tabela ASCII

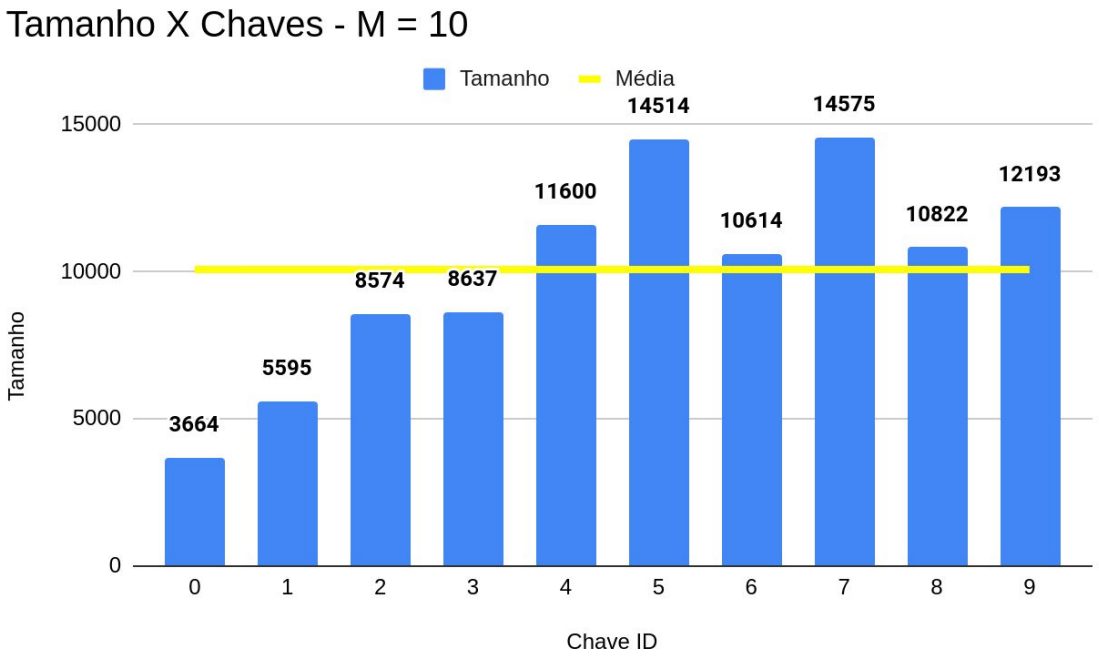
DEC	OCT	HEX	BIN	Símbolo	Número HTML	Nome HTML	Descrição
65	101	41	01000001	A	&#65;		A Maiúsculo
66	102	42	01000010	B	&#66;		B Maiúsculo
67	103	43	01000011	C	&#67;		C Maiúsculo
68	104	44	01000100	D	&#68;		D Maiúsculo
69	105	45	01000101	E	&#69;		E Maiúsculo
70	106	46	01000110	F	&#70;		F Maiúsculo
71	107	47	01000111	G	&#71;		G Maiúsculo
72	110	48	01001000	H	&#72;		H Maiúsculo
73	111	49	01001001	I	&#73;		I Maiúsculo

Fonte: Fernandes, 2020

No total foi gerado cinco tabelas hash com M diferentes, buscando encontrar um hashing ideal. A primeira tabela pode ser visualizada na Figura 02, onde foi usado o exemplo passado pelo professor M = 10, mesmo não sendo um número primo, a intenção foi ter um valor inicial para comparação com as outros valores gerados posteriormente. A média dos

valores foi calculado pelo total de nomes adicionados que foi 100788, dividido pelo M, nesse caso 10, sendo assim o valor ficou ≅ 10.078, os tamanhos não ficaram muito bem distribuídos.

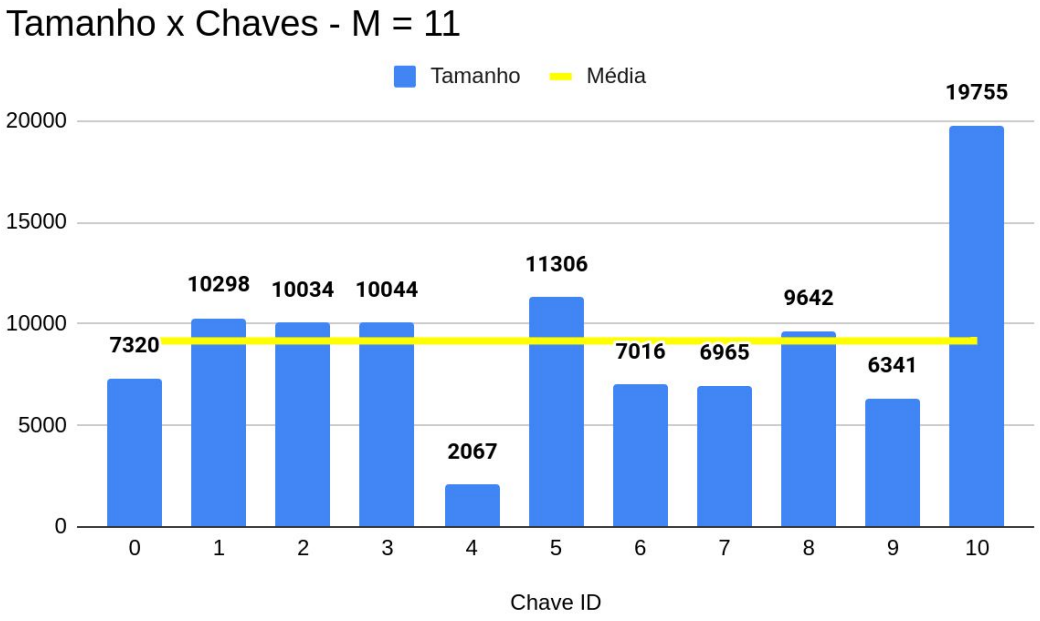
Figura 02 - Tabela Hash gerada com M = 10



Fonte: O autor.

O próximo valor testado foi o de 11 foi escolhido pois é um número primo e de valor próximo ao 10, o resultado pode ser visualizado na Figura 03, a média foi de ≅ 9.162, obtivesse chaves bem próximas a esse valor mas as chaves de ID igual a 4 e 10 tiveram valores bem diferentes, na ID igual a 4 foi de 4 vezes menor que a média e a ID igual 10 quase o dobro do valor da média.

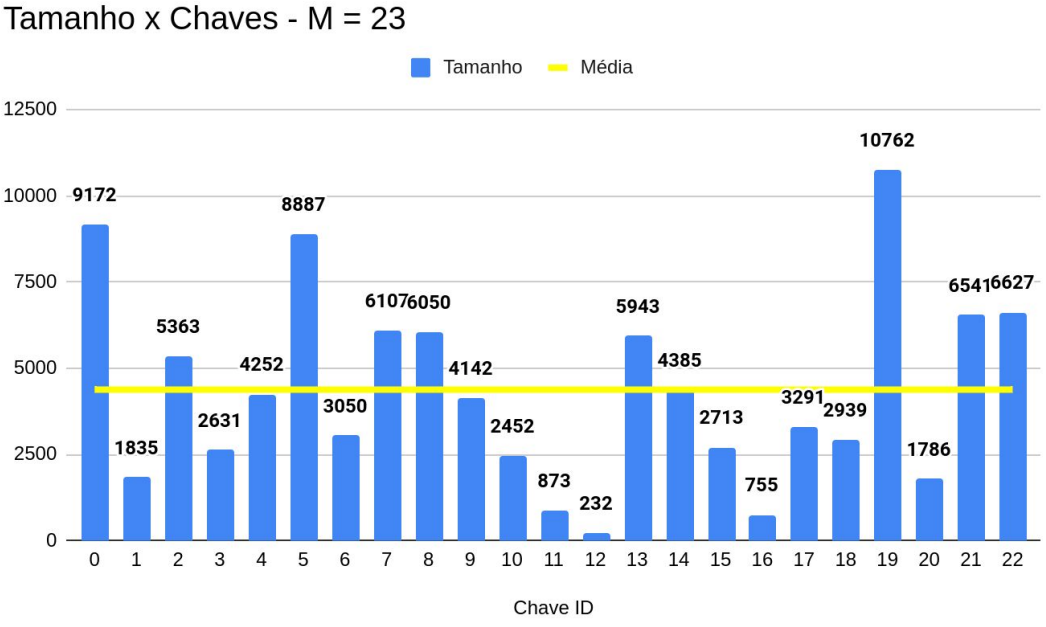
Figura 03 - Tabela Hash gerada com M = 11



Fonte: O autor.

Visando uma maior distribuição dos elementos por chave, foi testado com M = 23, sendo assim cada palavra do alfabeto estaria em uma chave, o resultado pode ser visualizado na Figura 04, houve muita variação no tamanhos das chaves sendo que a média para essa tabela foi de ≅ 4.382.

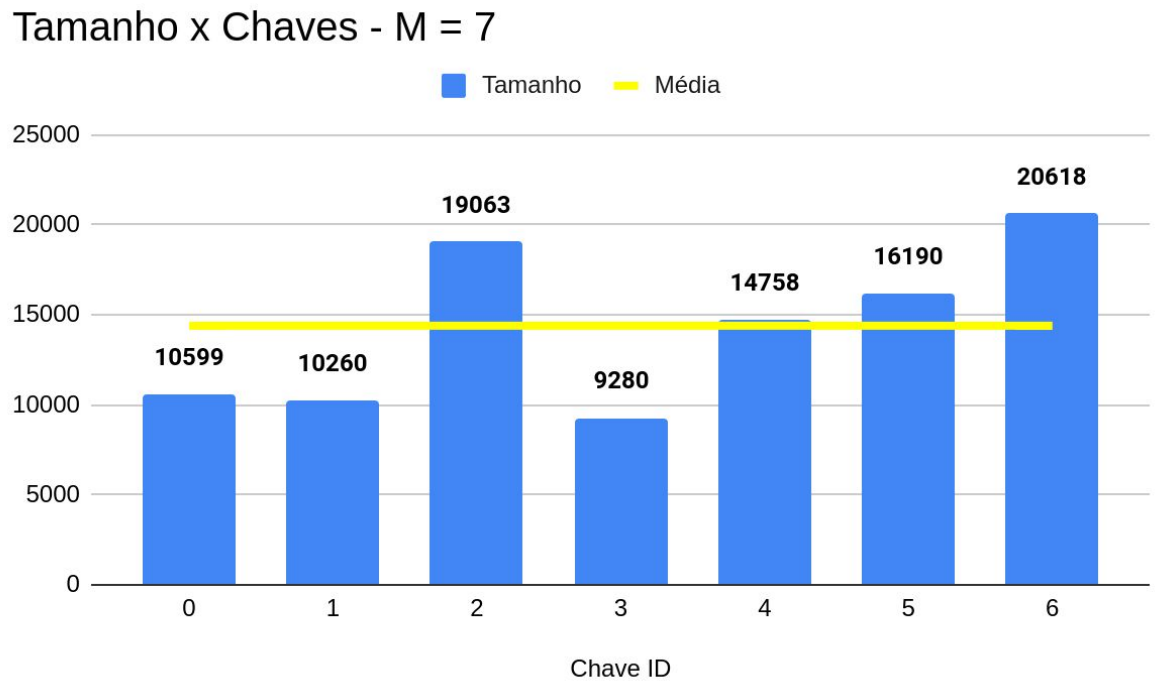
Figura 04 - Tabela Hash gerada com M = 23



Fonte: O Autor

Como pode ser observado até agora, quanto mais se aumentou o número de chaves mais ficou desbalanceado, então foi proposto diminuir esse número para um primo menor que 10, o primeiro então foi o 7, na Figura 05 pode ser observado a tabela hash de M = 7. Neste caso a média ficou ≅ 1.4398, assim houve mais chaves aproximadas desse valor, mais ainda com algumas bem acima desse número como a chave de valor ID igual 2 e 6.

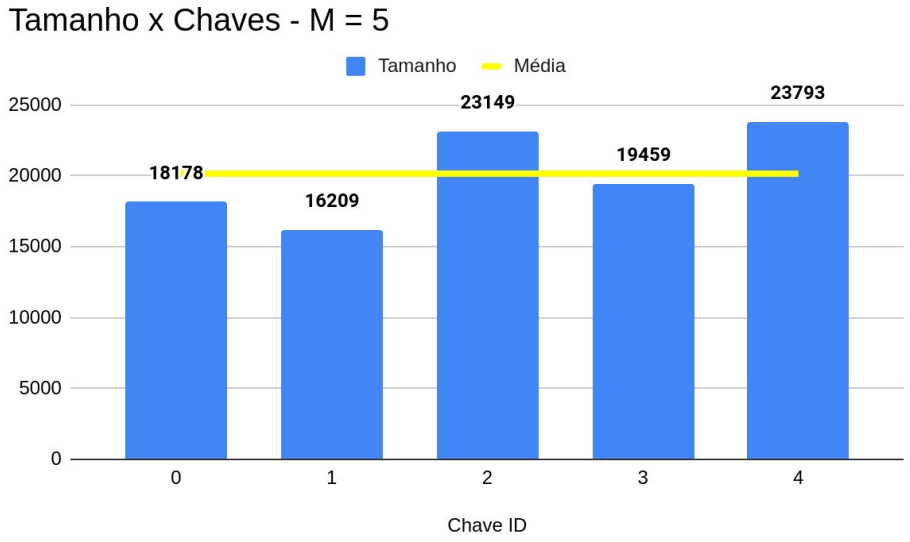
Figura 05 - Tabela Hash gerada com M = 7



Fonte: O autor.

Visando aprimorar ainda mais a distribuição de nomes por chaves foi gerado uma nova tabela com M = 5, que pode ser observado na Figura 06. A média foi de ≅ 2.0157, com essa quantidade de chaves houve uma maior distribuição de elementos/chave assim nessa função aparenta ter um hashing mais uniforme se comparado com os outros valores mostrados anteriormente.

Figura 06 - Tabela Hash gerada com M = 7



Fonte: O autor.

Como foi implementado utilizando listas encadeadas duplamente o tratamento de colisão já está sendo resolvido automaticamente, uma vez que existe uma lista com 5 chaves e cada chave é ligada a outra lista encadeada que tem os elementos esses são encadeadas entre si, sendo assim não haverá colisão, esse método é conhecido como hashing com encadeamento, um metodo mais soficitificado se comparado a sondagem linear onde teria muitos problema redimensionar o tamanho da tabela uma vez que nesse problema N é maior que M.

Em relação a ordenação dos elementos de uma chave, não houve sucesso nesse item, encontrou-se dificuldades para implementar o método e fazer a troca de posição dos elementos, no código fonte até tem o método mas não está funcionando corretamente.

Referências

FERNANDES, Henrique Marques. **Código ASCII – Tabela ASCII Completa**. 2020. Disponível em: https://marquesfernandes.com/codigo-ascii-tabela-ascii-completa/. Acesso em: 11 ago. 2020.