A busca linear peca em um grave aspecto. Embora seja fácil de implementar, não é muito eficiente, visto o tempo gasto ao percorrer uma lista sequencial no pior caso será O(log n). Infelizmente, as árvores binárias de busca só conseguem ser implementadas em sequências cujas chaves sejam distribuídas de forma ordenada, logo, não resolveriam esse problema.

Para solucionar o problema foi criado um algoritmo não baseado em comparação que provê operações de busca mais eficientes – as Tabelas Hash. Python, por exemplo, utiliza-se desta técnica para sua implementação de dicionários.

As tabelas de dispersão (hash) consistem em um tipo de estrutura de dados que associam chaves de pesquisa a valores, tendo por objetivo fazer uma busca rápida e obter o valor desejado a partir de uma chave simples.

*Hashing* é o processo de mapear uma chave de busca para um número limitado de índices de um array com o objetivo de prover acesso direto a estas chaves, de forma rápida e concisa. As chaves são armazenadas em um outro array chamado tabela hash (ou de dispersão) e uma função de dispersão é associada a esta tabela. A função converte (mapeia) as chaves de busca em entradas específicas na tabela.

Tomemos como base as chaves de busca a seguir e uma tabela de dispersão T que contenha M = 13 elementos.

765, 431, 96, 142, 579, 226, 903, 388

Poderemos então definir uma função simples de dispersão h(.) que mapeie as chaves em entradas da tabela hash:

*h(chave) = key % M*

O resto da divisão da chave pelo tamanho da tabela gerará sempre um valor válido entre os índices da tabela de dispersão:

h(431) => 2 h(96) => 5 h(142) => 12 h(579) => 7