1. A função Hash que utilizamos foi a “MAD” (Multiply, add and divide):

[(*ai*+*b*) mod *p*] mod *N*

Sendo **N** o número de elementos do Array Bucket, **p** um número primo maior que **N**, **a** um número inteiro aleatório entre **1** e **p-1**, **b** um número inteiro aleatório entre **0** e **p-1**, **i** é a chave do Hash. Desta forma teremos um menor número de colisões, conforme <http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/Map/hash-func.html>

1. O tratamento de colisões é feito alocando o elemento colidido na posição livre seguinte do Array. Caso o final do Array seja atingido, continua-se a busca pelo início do Array, até alocar o elemento ou retornar uma mensagem de que o Array está cheio.
2. A busca de itens é coerente com a inclusão de elementos.
3. Os elementos a serem alocados nos Arrays são objetos de uma classe, com atributos Chave e Valor.
4. A classe HashTable é muito robusta, testada de todas as formas.