Exercises about Hash Tables

Exercise 1

Given a hash table with 11 buckets, and the following hash function h:

$$h(k) = k \bmod 11$$

Shows the final state of this hash table after the insertion of the following keys (that were inserted in the left-to-right order) [22, 1, 13, 11, 24, 33, 18, 42, 31]. Assume a first case where the hash table handles collision through chaining. In a second case, assume that the hash table handle collision through open addressing (linear probing).

Exercise 2

You have to implement a dictionary. The key of each item is composed by a pair of uppercase letters. The only characters allowed in a key are 'A' (ASCII 65), 'B' (ASCII 66), 'C (ASCII 67) and 'D' (ASCII 68), giving rise to 16 possible distinct keys.

You have decided to solve this problem implementing the dictionary with a hash table, where the collisions are handled through chaining. However, due space constraints, the hash table has only 8 buckets. Since you want the search to present constant time, you have decided that no more than two keys will be mapped to the same bucket (in other words, there will be, at most, one collision per bucket).

Considering the above scenario, describe a hash function $h(k)$, where k is the key, that maps all the 16 keys to the hash table, ensuring that no more than 2 keys will be mapped to the same bucket.

Exercise 3: *from the book Introduction to Algorithms, Cormen T.H. et al.*

Consider a hash table of size $m = 1000$ and a corresponding hash function $h(k) = \texttt{floor}(m(kA \bmod 1))$ for $A = (\texttt{sqrt}(5) - 1)/2$. Compute the locations to which the keys 61, 62, 63, 64, and 65 are mapped.

Exercise 4

Once you understand the fundamentals of the hash tables (*i.e.* their concept and inner workings), one interesting approach to get a deeper understanding of how they are used in practice is to check how they are actually implemented in current applications. Thus, in this exercise we suggest that you take a look at how hash tables are actually implemented in two scripting languages that make heavy use of hash tables : Python and Lua.

As starting points, we suggest:

- Python: https://hg.python.org/cpython/file/52f68c95e025/Objects/dictobject.c
- Lua: http://webserver2.tecgraf.puc-rio.br/~lhf/ftp/doc/jucs05.pdf (Chapter 4)