

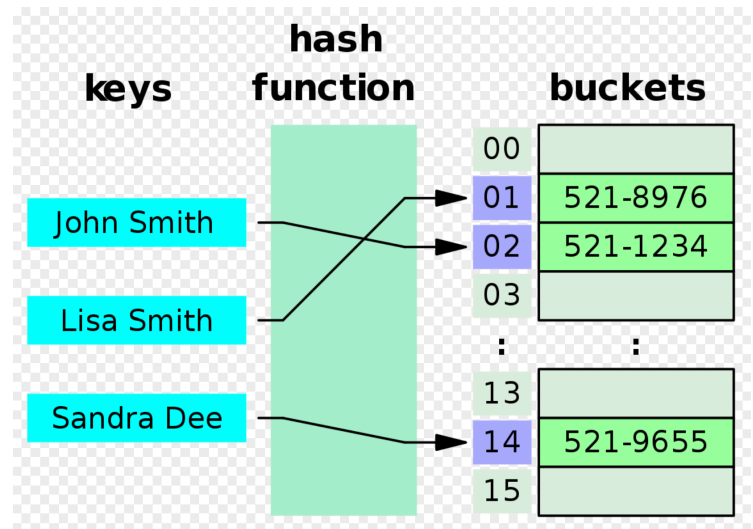
1) Arrays and Strings

Hash Tables

- Stores pairs of (key, value) in a way that is very efficient to lookup for the value using the key.

Below there is an implementation example using an array of linked lists:

- A hash function receives the key and compute an integer. (Simple example could be return the length)
- When stores a pair (k,v), compute the hash of the key, go to that index in the array and add the pair (k,v) to the linked list.
- To lookup with the key, compute the hash, get the linked list and search.
- Worst case $O(n)$, best case $O(1)$, depends on the design of the hash function.



Array List

- An array list (or resizable array) is an array that augment their size every time that the user try to append an element and the array is full.
- Still provides $O(1)$ access but every time it gets full it cost $O(n)$ the next insertion. In the end the insertion cost is $O(1)$ because the cost of increasing the array is amortized.
- Simple implementation for the auto augment is create a new array of double the size and put every thing in this new one.

String Builder

- Concatenation of 2 strings is in general expensive. Think of two string one of size n and the other of size m . Then to concatenate them you usually create a new string and put both there character by character $O(n+m)$
- String Builder data structure use a resizable array instead. Then to concatenate 2 string append each string in the string builder. So the array will be doubling their size but we know that is still $O(1)$ each insertion.