The frequency of collisions depends on the complexity of the hash function. Say we associated a number with each character. (a=1, b=2, c=3). To hash a string we could sum the characters. (bye=2+25+5=32). The hash value for “bat” and “tab” would be the same. We could also hash these strings by concatenating the integer values of the characters. (bye=022505). This way “bat” and “tab” would not have the same hash value. Collisions would occur much less frequently in the second hash function than the first.

The time it takes to insert or search for a value in a hash table depends on how the table deals with collisions. My hash table looks like a binary tree. I deal with collisions by chaining a linked list to a node. My insert function can take much longer depending on how many collisions occurred previously. It would have to iterate through the entire linked list to insert a colliding value. My search function is not impacted by collisions. The search function will just go through the binary search tree and not the linked lists.