Key:

size -> size of the hash table

n -> number of inputs to the hash table

list -> bucket in the hash table

1. If the hash function hashes everything to the same bucket then the complexity of insertion and search will be independent of the size of the hash function, *size* and will only depend on the number of inputs, *n*.

Insertion: Insertion is O(1) if new items are added to the beginning of the list that corresponds to a bucket in the hash table and O(n) if new items are being added to the end of the list.

Search: Search in the list should be O(n) where n is the number of inputs.

1. If the hash function spreads the input perfectly evenly over all the buckets then we may have to think about the number of inputs, n relative to the size of the hash table.

Insertion: Insertion is O(1) if we insert to the beginning of the list and n is of the same order as size.

Search: Search is O(1) if number of input is the same as the size of the

1. If the hash function never hashes two things to the same position, then it would mean that number of inputs must be less than the size of the hash table otherwise the collision-resistance of the hash function would not hold. If *n* < *size* then the insertion and look-up would be O(1).
2. This would be a bad hash function because we would be determining the hash for a given word using the first letter of the word, implying that if the input contains words that all start with the same letter then our hash function would cause our hash table to again turn into a fancy linked list. For example if our input was all the cities in the world with names beginning with M the search would thus be O(n) and insertion would be O(1) or O(n) depending on the implementation. For example: Melbourne, Macau, Madrid, Mumbai etc.
3. We know that if we implement a Universal Hashing function the probability of a collision is 1/size where size is the size of the hash table. Now, we can say on average we would *expect* to get *size* number of inputs (E[X]] of a geometric random variable) before we get a collision. So if we set-up the size of the hash table correctly to be larger than the expected number of inputs we should have O(1) time insertion and look-up operations.