Dictionary

○ Look-up: O(1)

○ Length: O(1)

○ Insert: O(1)

○ Delete: O(1)

○ dictionary.keys(): O(n) - because a list is generated

○ Check if a key is in the dictionary: O(1)

List

○ Append: O(1)

○ Length: O(1)

○ Insert: O(n)

○ Delete: O(n)

○ Copy: O(n)

○ Sort: O(n log n)

○ Check if an item is in the list: O(n)

Total Time = Time per Iteration \* # of Iterations or Time per Call \* # of Calls

Linear search

○ Brute force search

○ List doesn’t have to be sorted

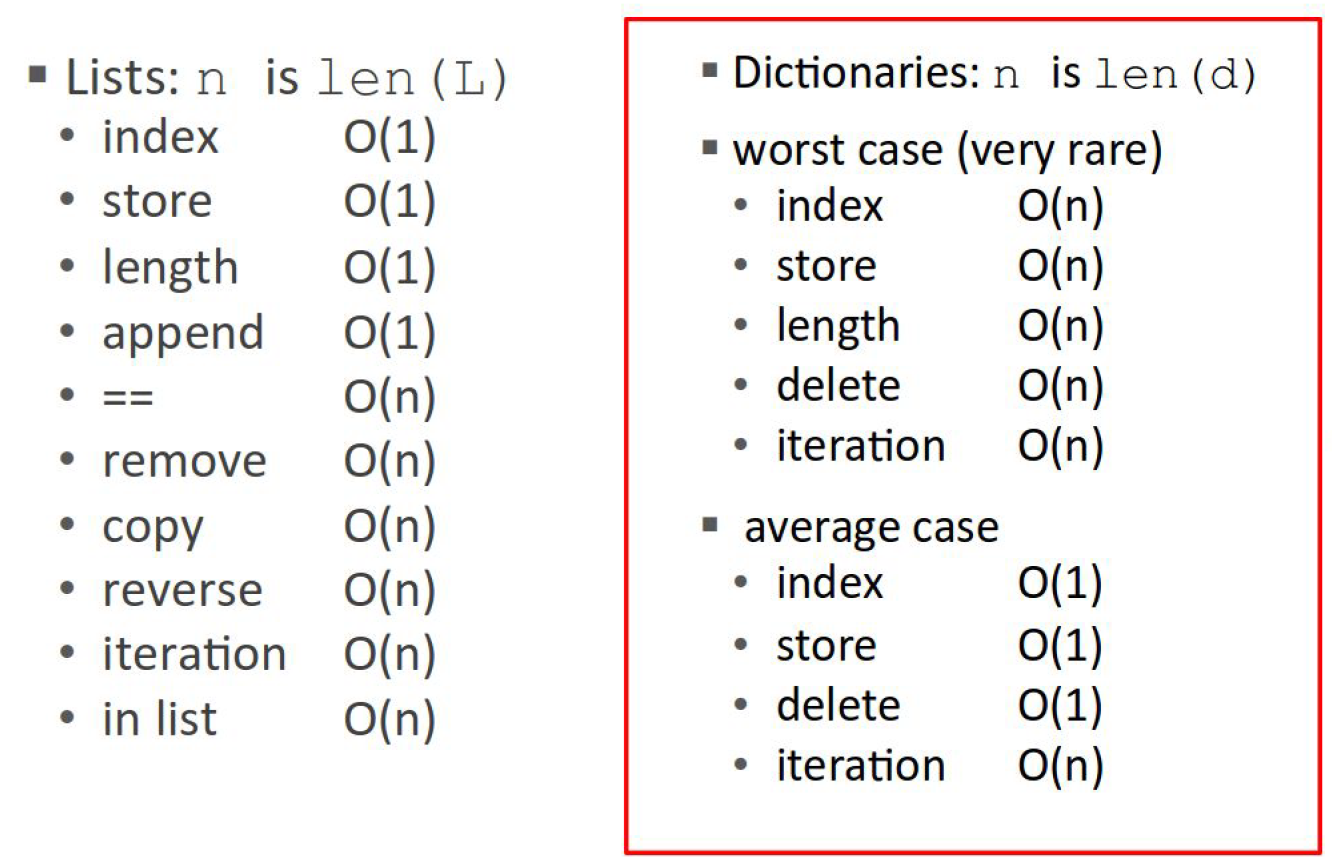
○ O(n)

● Bisection search

○ List must be sorted to give correct answer

○ O(log n)

Merge Sort ○ How many levels of the recursive tree? O(log n) ○ How much computation of each level of the tree? O(n) ○ Complexity? O(n log n)

If all keys hash to the same index Hash table large relative to number of keys Hash function good enough

A good hashing function should

● Be deterministic, i.e. it should not use any randomness (why?)

● Use the entire input in the hash computation

● Should map possible inputs to outputs uniformly (why?)