

# CS 1.2: Intro to Data Structures & Algorithms

## Hash Table Time Complexity Worksheet

Name: Marco Yip

### Given: Linked List Solutions – implementation and time complexity

The variable  $n$  represents the number of items stored in the list (or equivalently, number of nodes).

<i>Linked List operation</i>	<i>short summary in pseudocode (English) of the major steps performed in the implementation</i>	<i>best case running time</i>	<i>worst case running time</i>
is_empty	check if head node exists (None or not None)	$O(1)$	$O(1)$
length	traverse all nodes; count 1 for each node	$O(n)$	$O(n)$
append	add new node to end (after tail node); update tail property to point to new node	$O(1)$	$O(1)$
prepend	add new node to beginning (before head node); update head property to point to new node	$O(1)$	$O(1)$
find	traverse all nodes until matching data is found; if found, return matching data; if not, return None	$O(1)$	$O(n)$
delete	traverse all nodes until matching data is found; if found, set previous node to point to next node	$O(1)$	$O(n)$

### New: Hash Table Operations – implementation and time complexity

Use the variable  $n$  for the number of key-value entries stored and  $b$  for the number of buckets.

<i>Hash Table operation</i>	<i>short summary in pseudocode (English) of the major steps performed in the implementation</i>	<i>best case running time</i>	<i>average case running time</i>
length	Traverse through each node, and add 1 to the counter for each node.	$O(n)$	$O(n)$
items	Iterates through buckets, appends each bucket to an array, and returns the array.	$O(n)$	$O(n)$
contains	Takes in 'key', searches for the index of the bucket, uses that index to search the linked list with the key, and returns a boolean if the 'key' exists.	$O(1)$	$O(n)$
get	Takes in 'key', searches for the index of the bucket, uses that index to search the linked list with node with the key, iterates through the node and returns value associated with 'key'.	$O(1)$	$O(n)$
set	Takes in 'key' and 'value', searches for the index of the bucket, uses that index to search the linked list with node with the key, iterates through the head node and returns value associated with 'key', if not found, set next node as current and repeat. If key, value pair not found in linked_list, append the key-value pair to the end.	$O(1)$	$O(n)$
delete	Takes in 'key', searches for the index of the bucket, uses that index to search the linked list with node with the key, iterates through the node. if the current node matches the key, and if there is a previous node, set the previous node's next node as the next node of the current node. If the current node matches the key, and the head is none, set tail as none.	$O(1)$	$O(n)$