Algorithms and Data Structures

Assignment 2, Orders of Functions of Lists and Dictionaries

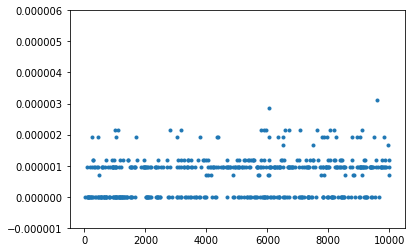
Devon Olds, Seth Grinstead, Douglas Mellon, Lucas Sanderson

This report details our findings on the BigO efficiency of various built-in python functions relating to lists and dictionaries.

**List Functions**

**Pop()**

The following is a graph of the run time of the pop() method as a function of list size:



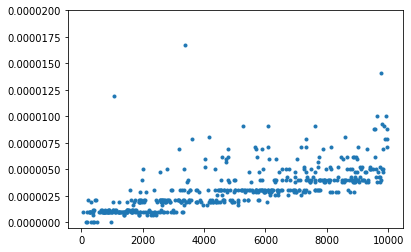
The data follows a constant pattern, close to Y=0, or O(1). This is because it will only require one “step” to remove the last element of a list.

Other O(1) operations:  
Index[]—returns item at a specific index, only requires one “step”

Append()—appends one item to the end of the list, only requires one “step”

**Pop(i)**

The following is a graph of the run time of the pop(i) method as a function of list size where i=0:



The data follows a linear pattern, or O(n). When the function removes the first element of the list, all the subsequent elements must have their indices decreased by one.

Other O(n) operations:

Insert(i,item)—inserts an element at a certain index, and shifts the indices of all subsequent elements

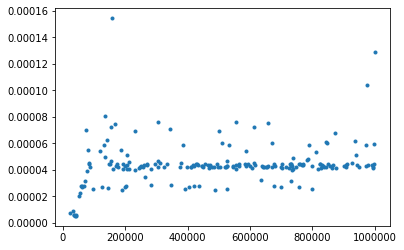
Del—deletes an element and shifts the indices of all subsequent elements

Iteration—iterates a function over n elements in a list

Contains—must search through each element of the list until the element in question is found, or until the end of the list is reached

**Get slice**

The following is a graph of the run time of the get slice method as a function of the length of the slice:



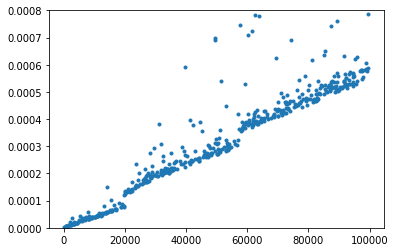
The data shows the method is O(k), or constant time. It runs through the elements of the slice and copies them to a new list. However, since the length of a slice is almost always less than n, the length of the list, the order is k.

Other O(k) operations:

Concatenate—Combines two lists, and k is the length of the second list

**Set slice**

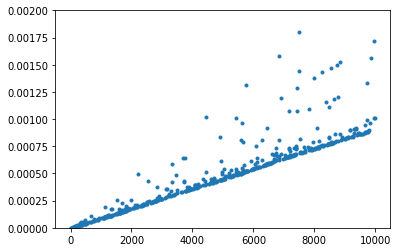
The following is a graph of the run time of the set slice method as a function of the length of the slice:



The data shows the method is O(n+k). Concatenating the beginning of the first list with the new list takes k steps, where k is the length of the new list. Reassigning the indices of the rest of th first list then takes n steps, where n is the length of the rest of the list.

**Sort**

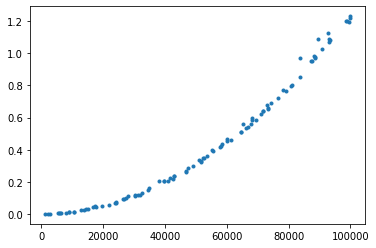
The following is a graph of the run time of the sort method as a function of the length of the list:



The data shows the method is O(n log n). This is the fastest possible method in python to sort a random list.

**Multiply**

The Following is a graph of the run time of the multiply method as a function of the length of the list:



The data shows the method is O(nk). The method appends n (the length of the list) elements k times to the list.

**Dictionary Functions**

**Copy**

The following is a graph of the run time of the copy method as a function of the dictionary size:

The data shows the method is O(n). This method copies n elements of the dictionary, and thus is linear.

Other O(n) operations:

Iteration—as a function iterates over all elements of a dictionary, it will perform the task n times.

**Get**

The following is a graph of the run time of the get method as a function of the size of the dictionary:

The data shows the function is O(1). Only one step is required to fetch the key from the dictionary.

Other O(1) operations:

Set item—Only one step is required to fetch the key and set its associated value.

Delete item—only one step is required to fetch the key and delete it.

Contains—Only one step is required to fetch the key in question and return if it is present.