Jack Micher

9/10/2020

CSIT 200

Lab 1 Part III Questions

1. After modifying \_\_getitem\_\_ to allow negative indices which behave like in Python’s list[] method, the speed would be slightly slower. The method ran at O(1) time before and still runs at O(1) time after modification, however the few extra lines added still affects the speed and makes it technically slower.
2. The running speed of \_\_setitem\_\_ is O(1), as the array’s index is immediately jumped to and has its value replaced.
3. The running time of pop() is O(1), as the method immediately attempts to jump to the last index to remove the value and index, then returns the value that it had removed.
4. The running time of \_\_eq\_\_ is O(n), as it would have to go through each index of both lists to compare values. However, if \_\_eq\_\_ finds that both lists are different in size, it immediately returns false and the running time becomes O(1).
5. Both lab1\_v1 and lab1\_v2 take the current array stored in the dynamic\_array module and then randomly scramble the values stored in each index of the array, while making sure that each value remains. A good name for these methods would be scramble\_v1 and scramble\_v2.
6. The running time of lab1\_v1 is O(n­2), because even if the deeper nested for-loop does get smaller proportionally to the upper nested for-loop with each upper for-loop iteration, the inner loop is still contributing to an O(n2) time as it would excute n times, then n-1 times, then n-2, … for each x in the upper for-loop (for x in range(self.\_n)).  
     
   The running time of lab1\_v2 is O(n) as the upper for-loop (for i in range(10)) takes up O(1) time, while the for-loop inside (for j in range(self.\_n)) is proportional to the current length of the array (O(10n) if going for more detail).