Kern Espinoza

CS260

25 October, 2021

**Assignment 2 Write-Up**

1. ArrayList “combine” function:
   1. *ArrayList<Address>::combine* works by looping through the other list and calling *insertEnd()* on each item in the other list. If this would add more items than the arraylist can currently hold, it will first invoke *grow()*, which simply copies the array over to a new array of double the size.
   2. Since both of the underlying operations (*insertEnd()* and *grow()*) should be O(n), I would consider this function to be **O(n)**.
2. “extractAllMatches” function:
   1. The function begins by creating two new ArrayLists - one to hold extracted elements (to be returned), and one to hold all non-extracted elements (to replace the original list). After creating these two empty lists, it iterates through the original list. Elements which match the criteria for extraction are moved into the new list, while non-matching elements are moved into the other list.
   2. This function is **O(n)** because it simply iterates through the original list two different times (non-nested loops).
3. LinkedList “combine” function:
   1. The function *AddressLinkedList::combine* works by traversing the other linked list, inserting each element at the end of the original list before removing it from the list to be combined.
   2. Because the *insertEnd* and *removeFirst* functions should be O(1) and the main *combine* function contains only a single loop, it should be **O(n)**.
4. LinkedList “extractAllMatches” function:
   1. The function begins by creating two node pointers and an empty list. It then traverses the original list. Matching elements are inserted into the new list and then removed from the old list, one at a time.
   2. Both of the inner functions (*insertEnd()* and *removeFirst()*) should be O(1) because they perform simple operations without any loops. Therefore, since the *extractAllMatches()* function only traverses the list a single time, it should be **O(n)**.