COSC 2P03

Assignment 1

May 25th, 2018

Sawyer Fenwick

6005011

```
public void addTriple(E first, E second, E third){
    List<E> l = new ArrayList<>();
    l.add(first);
    l.add(second);
    l.add(third);
    count += 3;
    Node<E> p = head;
    Node<E> q = null;

    while(p!= null){
        q=p;
        p=p.next;
    }
    if(q==null){
        head = new Node<>((E)l, 3, null);
    }
    else{
        q.next = new Node<>((E)l, 3, null);
    }
    System.out.println("\t" + "Added [" + first + "," + second + ","
            + third + "]" );
}//addTriple
```

My addTriple method is O(n). Adding the elements to the list is O(1), the while loop goes over each node in the list which is O(n). Both cases in the if-then-else statement are O(1) so it is O(1). This gives O(1) + O(n) + O(1) which gives O(n).

```
public boolean hasTriple(){
    Node<E> p = head;
    boolean t = false;
    while(p!=null){
        t = search(3,p);
        if(t){
            break;
        }
        p=p.next;
    }
    return t;
}//hasTriple
```

My hasTriple method is O(n). The while loop is O(n) since it must go through “n” nodes at worst case. The if statement is O(1). Therefore the method is O(n).

```java
public List<E> removeTriple(){
   Node<E> p = head;
    Node<E> q;
    List<E> list = new ArrayList<>();
    List<E> list2;
    q = build(3,p);
    delete(q);
    while(q!=null){
       switch (q.size) {
           case 1:
               list.add(q.item);
               break;
           case 2:
               list2 = (List<E>)q.item;
               list.add(list2.get(0));
               list.add(list2.get(1));
               break;
           case 3:
               list2 = (List<E>) q.item;
               list.add(list2.get(0));
               list.add(list2.get(1));
               list.add(list2.get(2));
               break;

       }
       q = q.next;
    }
    count-=3;
    return list;
}//removeTriple
```

The complexity of my removeTriple method is $O(n^2)$. The while loop is O(n) and so is the switch. Which gives $O(n)*O(n)$ which gives $O(n^2)$.

For the generalized “add(E…item)” I would ask the user how many items they would like to add, and then add them one at a time. Ex/

“How many items?”

6 (n=6)

For(x<n)

Get item input

List.Add(E…item)

Return list

For the generalized “remove(int n)” I would search the list for the number requested but I would follow the same delete method I used, building a new list that holds all the elements I want to delete from the main list and then comparing the two lists. It would just be in a different method call.