Assignment #3

Due Dates: Wednesday, September 23 at 11:59pm

Submit: eLearning

Late Policy: -10 points per hour late

Instructions: This is an individual assignment. Answers should be your own work.

Chapter 3

10 points

1. What are some advantages of a linked list versus an array?

Solution:

Linked list doesn’t need a contiguous memory but an array needs.

In a LinkedList new nodes can be inserted or deleted in a constant time just by re-arranging the links of the linked list i.e. order O(1) while in case of ArrayList the nodes should be moved to make or remove space which takes O(N) time.

15 points

2. What is the Big-O running time of the following code fragment?

Assume lst1 has N items, and lst2 is initially empty.

public static void add( List<Integer> lst1, List<Integer> lst2)

{

for ( Integer x : lst1 )

lst2.add(0, x); // add to front

}

a. If an ArrayList is passed for lst1 and lst2. Explain your answer.

Solution:

Adding one new node at beginning of the ArrayList needs to move all the remaining nodes one step right which takes O(N) and the outer loop runs for N times so it is O(N). Totally it is O(N2).

b. If a LinkedList is passed for lst1 and lst2. Explain your answer.

Solution:

Adding one new node at beginning of the LinkedList needs a constant time which is O(1) and the outer loop runs for N times so it is O(N). Totally it is O(N).

15 points

3. What is the Big-O running time of the following code fragment?

public static void erase( List<Integer> lst )

{

Iterator<Integer> itr = lst.iterator();

while ( itr.hasNext() )

{

Integer x = itr.next();

itr.remove();

}

}

a. If an ArrayList is passed for lst. Explain your answer.

Solution:

Removing one node from the ArrayList needs to shift all the remaining nodes which takes O(N) time and also the outer loop takes O(N) time. So totally O(N2).

b. If a LinkedList is passed for lst. Explain your answer.

Solution:

Removing one node from the LinkedList needs a constant time i.e. O(1) since the position is already known while the outer loop takes O(N) time. So totally O(N).

15 points

4. What is the Big-O running time of the following code fragment?

Assume lst1 has N items, and lst2 has N items.

public static int Count( List<Integer> lst1, List<Integer> lst2)

{

Iterator<Integer> itr1 = lst1.iterator();

int count=0;

while ( itr1.hasNext() )

{

Integer x = itr1.next();

Iterator<Integer> itr2 = lst2.iterator();

while ( itr2.hasNext() )

if ( x.equals( itr2.next()) )

count++;

}

return count;

}

a. If an ArrayList is passed for lst1 and lst2. Explain your answer.

Solution:

The outer while loop runs N times as well as the inner loop runs N times while the equals and itr.next() operation takes only 1 time unit. Thus totally O(N2)

b. If a LinkedList is passed for lst1 and lst2. Explain your answer.

Solution:

The outer while loop runs N times as well as the inner loop runs N times while the equals and itr.next() operation takes only 1 time unit. Thus totally O(N2)

15 points

5. What is the Big-O running time of the following code fragment?

public static int calc( List<Integer> lst )

{

int count = 0;

int N = lst.size();

for ( int i=0; i<N; i++)

{

if (lst.get(i) > 0)

sum += lst.get(i);

else

sum += lst.get(i) \* lst.get(i);

}

return sum;

}

a. If an ArrayList is passed for lst. Explain your answer.

Solution: The outer loop runs for N times and the lst.get(i) takes constant time for ArrayList. In the case if else also lst.get(i) takes constant time which gets added. Thus totally O(N).

b. If a LinkedList is passed for lst. Explain your answer.

Solution: The outer loop runs for N times and the lst.get(i) takes O(N) for LinkedList. In the case if else also lst.get(i) takes O(N) which gets added. Thus totally O(N2).

15 points

6. Suppose a Java method receives a List<Integer> and reverses the order of the items it contains by removing each item from the front of the list, adding each item to a Stack<Integer>, and then removing the items from the stack and inserting each item to the end of the list.

What is the expected Big-O running time if:

a. If an ArrayList is passed. Explain your answer.

Solution:

In an ArrayList the node removal from front takes O(N) time while the Insertion at the end takes only constant time. Also in case of stack (either ArrayList or LinkedList implementation) takes constant time for push or pop. All these removal and insertion happens for N times (for all nodes in the List). Thus totally O(N2) only.

b. If a LinkedList is passed. Explain your answer.

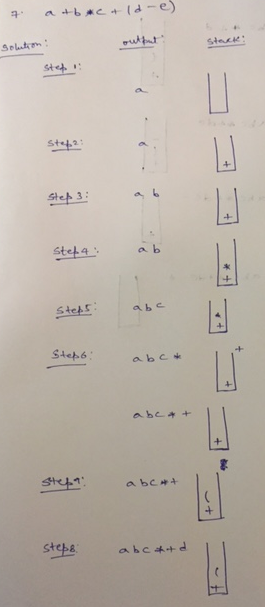
Solution:

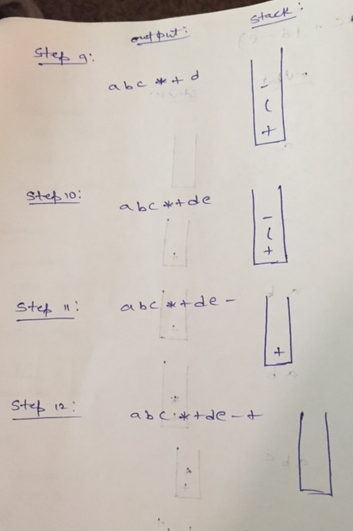
In a LinkedList the node removal from front takes constant time and also the Insertion at the end takes only constant time. Also in case of stack (either ArrayList or LinkedList implementation) takes constant time for push or pop. All these removal and insertion happens for N times (for all nodes in the List). Thus totally O(N) only.

15 points

7. Show each step of converting a+b\*c+(d-e) from infix to postfix notation, using the algorithm described in the textbook that uses a stack.

Solution:





Submit to eLearning:

hw3.doc (.doc can be .txt, .jpg, etc.)