Brandon Kirklen
2015-06-01
CPE 102-22

Home Assignment 6

CPE 102

Chapter 15

1. A gradebook application stores a collection of quizzes. Should it use a list or a set?

A list because the order of the quizzes might be important for comparing quizzes.

2. A student information system stores a collection of student records for a university. Should it use a list or a set?

Set storage would be most useful, there's no value to having them ordered by anything.

3. Why is a queue of books a better choice than a stack for organizing your required reading?

A queue ensures that all the books would be read in approximately the same time whereas only adding them to a stack could prevent older stuff from getting read.

4. As you can see from Figure 1, the Java collections framework does not consider a map a collection. Give a reason for this decision.

A map has key value pairs which make collection type methods not useful on them as both values would be required to do things. Methods like "size" would be ambiguous between the value and the keys.

5. Do linked lists take more storage space than arrays of the same size?

Yes, they have to include pointers to the next element.

6. Why don't we need iterators with arrays?

Because arrays are indexed by integers, so in a way we do need iterators, they just don't need another layer of abstraction like the linkedlist does.

7. Suppose the list letters contains elements "A", "B", "C", and "D". Draw the contents of the list and the iterator position for the following operations:

```
ListIterator<String> iter = letters.iterator();
```

*ABCD

```
iter.next();
```

A*BCD

```
      iter.next();
```
AB*CD

```
      iter.remove();
```
A*CD

```
      iter.next();
```
AC*D

```
      iter.add("E");
```
ACE*D

```
      iter.next();
```
ACED*

```
      iter.add("F");
```
ACEDF*

8. Write a loop that removes all strings with length less than four from a linked list of strings called words.

```
public static void remove(LinkedList<String> list){
    for( int i = list.size()-1; i >= 0; i--){
        if ( list.get(i).length() < 4 ){
            list.remove(i);
        }
    }
}
```

9. Write a loop that prints every second element of a linked list of strings called words.

```
public static void printSecond(LinkedList<String> words){
    ListIterator<String> iterator = words.listIterator();
    while (iterator.hasNext()){
        System.out.println(iterator.next());
        if ( iterator.hasNext() ){
            iterator.next();
        }
    }
}
```

Chapter 16

1. Trace through the addFirst method when adding an element to an empty list.

Brandon Kirklen
2015-06-01
CPE 102-22

At the beginning of the process there is no allocated nodes. The first node is then allocated and the data variable is set. The next instance variable is still null because the list terminates there. The first variable is set to the nodes.

2. Conceptually, an iterator is located between two elements (see Figure 9 in Chapter 15). Does the position instance variable refer to the element to the left or the element to the right?

The iterator is referring to the left element as it starts on the far left.

3. Why does the add method have two separate cases?

Because if the element being inserted is the first element then the first instance variable also has to be set.

4. Assume that a last reference is added to the LinkedList class, as described in Section 16.1.8. How does the add method of the ListIterator need to change?

If the last instance variable is changing then the add method will also need to update the last instance variable.

5. Provide an implementation of an addLast method for the LinkedList class, assuming that there is no last reference.

```
        public void addLast(Object e) {
        if (first == null) {
              addFirst(e);
        }
        else {
              Node last = first;
              while ( last.next != null) {
                    last = last.next;
              }
              last.next = new Node();
              last.next.data = e;
        }
}
```

6. Expressed in big-Oh notation, what is the efficiency of the addFirst method of the LinkedList class? What is the efficiency of the addLast method of Self Check 5?

O(1) and O(n) as the addLast method will have to visit each element of the list.

7. How much slower is the binary search algorithm for a linked list compared to the linear search algorithm?

Brandon Kirklen
2015-06-01
CPE 102-22

Significantly slower as finding the middle element each time will take n / 2 for each set of elements. The linear search algorithm at worst will take n time and average n/2

8. Why is it much more expensive to get the kth element in a linked list than in an array list?

Because there is no way to move directly to the kth element in a linked list, the only way is to look through each element of the list for the next element. For an array list, the object has an array of pointers to each element in the array list.

9. Why is it much more expensive to insert an element at the beginning of an array list than at the beginning of a linked list?

Because inserting into a linked list only requires the changing of the instance variable of the current first element in a doubly linked list, and doesn't require any changes in a single link list. In an array list everything has to be shifted down because all of the elements need to be moved.

10. What is the efficiency of adding an element exactly in the middle of a linked list? An array list?

Assuming that you don't already have a pointer to the middle of the linked list, then you'll have to iterate through n/2 elements in the linked list and then preform a constant time operation. In an array list, while you can get to the middle in constant time, you'll have to move n/2 number of elements in order to add another element in the middle of the list. So functionally they have the same time complexity, assuming that the significant time suck is the iterating over elements and that the actual moving of data and moving of a pointer take equal time.

11. Suppose we insert an element at the beginning of an array list, and the internal array must be grown to hold the new element. What is the efficiency of the add operation in this situation?

The efficiency of the add operation would be equal to adding it without growing the array, moving the elements of an array and rewriting it would take equal amounts of time, O(n). Rewriting has to happen in either case, be it to the end of the array or to a new array all together.

12. Using big-Oh notation, what is the cost of adding an element to an array list as the second-to-last element?

Adding to the penultimate position on an array is going to be constant time assuming that you don't have to reallocate the array.