

Section 7

CSCI E-22

Will Begin Shortly

Searching a list

Suppose we have a list of items *in sorted order*, in both the `ArrayList` implementation and the `LLList` implementation.

If we want to find a single item in the list, what can we do?

- We could make a linear pass through the list, checking whether each item in the list is the one we want.
- Or...?

Searching a list

Suppose we have a list of items *in sorted order*, in both the `ArrayList` implementation and the `LLList` implementation.

If we want to find a single item in the list, what can we do?

- We could make a linear pass through the list, checking whether each item in the list is the one we want.
- Or...? [We could try to take advantage of the fact that the list is sorted by using binary search!](#)

What is the efficiency if we use binary search on the array implementation?

How about the linked list implementation?

What would be a better way to search if we knew we were using the linked list implementation?

Searching a list

However, implementing a linear search with an `LLList` is not as straightforward as it might seem. Let's say that we are using an instance of our `LLList` class in which the items are in sorted order, and that our `search()` method is *external* to the `LLList` class. Let's also assume that the objects in the `LLList` implement the `Comparable` interface.

Searching a list

Here's our `LLList search()` method implemented using an iterator:

```
public static boolean search(Comparable item, LLList list) {
    if (item == null || list == null) {
        return false;
    }

    ListIterator iter = list.iterator();
    while (iter.hasNext()) {
        Comparable listItem = (Comparable)iter.next();
        if (item.equals(listItem)) {
            return true;
        }

        if (item.compareTo(listItem) < 0) {
            // we went past what we're looking for
            return false;
        }
    }

    return false;
}
```

Because the iterator has a reference to the underlying linked list, we can use it to iterate over the internals of the `LLList` *as if* we had that direct access!

In other words, with an iterator, we can perform an $O(n)$ search on a List implemented with *any* underlying data structure.

Stacks

- A *stack* is a collection that follows the last-in, first-out (LIFO) rule.
 - `push(item)` is used to insert a new item at the top
 - `pop()` is the only way to remove an item, and it removes the topmost item
 - `peek()` is used to access the topmost item without popping it

Stacks

- A *stack* is a collection that follows the last-in, first-out (LIFO) rule.
 - `push(item)` is used to insert a new item at the top
 - `pop()` is the only way to remove an item, and it removes the topmost item
 - `peek()` is used to access the topmost item without popping it
- In lecture, we covered two implementations of the `Stack` interface: `ArrayStack` and `LLStack`.
 - For both implementations, what is the time complexity of `push()`, `pop()`, and `peek()`?
 - How do both implementations use memory?

Stacks

- Suppose there are a number of `push()` and `pop()` calls to a stack. The numbers 1 through 8 will be pushed onto the stack in order, with zero or more pops after each push. When a number is popped, it is immediately printed. Which of the following outputs are impossible?
 - 1 2 3 4 8 7 6 5

Balancing braces

- In lecture, we briefly mentioned an application for stacks that involves processing a sequence of delimiters (parentheses, braces and brackets) to ensure they are balanced.
- A sequence of delimiters is balanced when every left delimiter has a matching right delimiter in the correct order. For example:
 - `{{{}}}` is balanced because every opening delimiter has a closing delimiter and there are no extra closing delimiters
 - `{{{}}{}{}}` is balanced (open brackets can come after closing brackets as long as they are eventually closed)
 - `{([)]}` is not balanced (mismatching delimiter types)
 - `((((` is not balanced (missing closing delimiters)
 - `))))` is not balanced (missing opening delimiters)
- Let's write a method `isBalanced()` that takes a string containing delimiters and returns `true` if the delimiters are balanced, and `false` otherwise.