CSCI 136: Data Structures and Advanced Programming
Lecture 11
Lists
Instructor: Dan Barowy
Williams

Topics
• Mathematical induction
  Vectors—why add is “always O(1)”
• Linked lists

Your to-dos
1. Read before Wed: Bailey, Ch 9.4–9.5.
2. Lab 3, due Tuesday 10/4 by 10pm.
3. Quiz 4, due Saturday by noon.

Announcements
• Colloquium: What I Did Last Summer (Industry), 2:35pm in Wege Auditorium with cookies.
Quiz 3

Which of the following completions of the expression calculating midIndex results in the recursive function correctly calculating the smallest element of the array?

(a) int midIndex = (startIndex + endIndex) / 2;
(b) int midIndex = (endIndex - startIndex) / 2;
(c) int midIndex = endIndex / 2;
(d) int midIndex = (endIndex + 1) / 2;

Well done!

Quiz 3

Assuming this Java method is completed correctly, what is the Big-O running time of this algorithm assuming the length of the array is n?

```java
public static int findSmallest(int[] array, int startIndex, int endIndex)
{
    if(startIndex == endIndex)
        return array[startIndex];
    else
    {
        int midIndex = _______; // Select code to complete
        int firstHalf = findSmallest(array, startIndex, midIndex);
        int secondHalf = findSmallest(array, 1 + midIndex, endIndex);
        return Math.min(firstHalf, secondHalf);
    }
}
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Correct Answers</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(1)</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>O(n/2)</td>
<td>18 respondents</td>
<td>31%</td>
</tr>
<tr>
<td>O(log n)</td>
<td>14 respondents</td>
<td>24%</td>
</tr>
<tr>
<td>O(n)</td>
<td>26 respondents</td>
<td>45%</td>
</tr>
</tbody>
</table>

Let’s discuss this problem...

Quiz 3

Assuming this Java method is completed correctly, what is the Big-O running time of this algorithm assuming the length of the array is n?

```java
public static int findSmallest(int[] array, int startIndex, int endIndex)
{
    if(startIndex == endIndex)
        return array[startIndex];
    else
    {
        int midIndex = _______; // Select code to complete
        int firstHalf = findSmallest(array, startIndex, midIndex);
        int secondHalf = findSmallest(array, 1 + midIndex, endIndex);
        return Math.min(firstHalf, secondHalf);
    }
}
```

Mathematical Induction

\[ T(n) = (c_1 \times n) + (c_2 \times (\log_2(n) - 1)) = O(n) \]
Principle of Mathematical Induction

Hypothesis: \( P(n) \) is true for all integers \( n \geq a \),

1. Base case: \( P(a) \) is true.
2. Inductive step:
   For all integers \( k \geq a \), if \( P(k) \) is true then \( P(k+1) \) is true.

To be clear:

If you want to prove that \( P(n) \) is true for all integers \( n \geq a \),

1. You must first prove that \( P(a) \) is true.
2. Then suppose \( P(k) \) is true and prove that \( P(k+1) \) is true.

Expanding vectors: why double?

Why is the array doubling strategy for Vector better than expanding the array one element at a time?

One-at-a-time expansion

Initial array.

```

```

Insert element.

```

```

New array; copy previous; insert element.

```

New array; copy previous; insert element.

```

New array; copy previous; insert element.

```

...
**Insertion into an array**

How much does **array insertion** cost?

```
| 'a' | 'b' | 'c' | 'd' |
```

It costs $O(1)$.

In fact, lookup and insertion both cost $O(1)$.

**Tradeoff:** arrays are fixed size.

---

**Copying an array**

How much does an **array copy** cost?

```
| 'a' | 'b' | 'c' | 'd'|   |
```

It costs $O(1) \times m$, where $m$ is the size of the original array.

$\approx O(m)$

---

**How many copies?**

# of copies for one-at-a-time expansion:

```
1 + 2 + 3 + ... + (n-1)
```

Recall theorem: $1 + 2 + 3 + ... + k = k(k+1)/2$

Sub $n-1$ for $k$: $\frac{(n-1)((n-1)+1)}{2} = \frac{n(n-1)}{2}$

$= \frac{n^2-n}{2}$

One-at-a-time expansion costs $\approx O(n^2)$

---

**How many copies?**

# of copies for doubling expansion:

```
1 + 2 + 4 + ... + (n/2)
```

Neat theorem: $1 + 2 + 4 + ... + 2^{k-1} = 2^k - 1$

Suppose $n = 2^k$.

Then $1 + ... + n/2 = 1 + ... + 2^{k/2}$

$= 1 + ... + 2^{k-1} = 2^{k-1} = n-1$

Doubling expansion costs $\approx O(n)$
Which is faster?

One-at-a-time expansion costs $\approx O(n^2)$

Doubling expansion costs $\approx O(n)$

Doubling is Vin Diesel-approved.

A good practice induction problem

Prove: $n$ cents can be obtained by using only 3-cent and 8-cent coins, for all $n \geq 15$.

Linked Lists

Linked List

A linked list is a recursive data structure. A linked list is composed of simple pieces called list nodes. A list node contains data (of generic type $T$) and a reference (a “link”) to either another list node or null.
The empty list is defined as \textit{null}.

Every other list has at least one list node.

A list node stores data of type $T$. Here, $T$ is \textit{Integer}.

The \textit{next} field stores a reference ("link") to the next node. If the node is the last node, the next node is \textit{null}. 
Linked List

If the next node is not null, it is, recursively, a list node.
The last node in the list must always point to null.

Linked List

A list has parts.

Linked List

When we add data to a list, we always append to the head.

Linked List

To find a value, we must always traverse the list starting from the head.
E.g., looking for 2...
To find a value, we must always traverse the list starting from the **head**.

E.g., looking for **2**…

---

**Recap & Next Class**

**Today:**

- Why Vector should double
- Lists

**Next class:**

- ADTs
- More lists