CS 367 - Introduction to Data Structures
Thursday, October 6th, 2016

Homework 4 (quiz) due 10 pm Friday, October 7th
(one attempt -- 2 hrs to complete -- available by 10am Wednesday morning)

Program 1 Peer Reviews: each student has been assigned two submissions to review. Please comment on the readability of the code assigned to you. Did the submission follow the style and commenting guidelines? Rate the overall work on a scale of 1-10 and share your comments and suggestions with the student who submitted the work. Complete Peer Review before Oct 11th.

Program 2 due 10 pm Sunday, October 23rd - GET STARTED NOW!
Late work is not Accepted

Last Time
Iterable and For-Each Loops
More Linked List Variations
- double linking
- circular linking
Intro to Complexity
- concept
- big-O notation

Today
Complexity
- best/worst cases
- analyzing Java code (from last time)
- practice analyzing Java code (from last time)
- significance of scaling
- caveats
Comparing ArrayList vs LinkedList

Next Time
Read: start Stacks and Queues
Shadow Array - improving array resizing
Stack ADT
- concept
- array implementations
- chain of nodes implementations
Queue ADT
- concept
- chain of nodes implementations
Number Guessing Game

Picker picks a number (positive integer)
Repeat until number is guessed:
  Guesser guesses a number
  Picker answers "correct", "higher", or "lower"

  problem size:
  dominant operation:

→ What is the complexity of each algorithm below that the guesser uses to decide the sequence of numbers to give as guesses?

Algorithm 1:
  guess = 1
  repeat
    If guess incorrect, increment guess by 1
  until correct

Algorithm 2:
  guess = /2
  step = /4
  repeat
    If guess is too small, increase guess by step
    otherwise decrease guess by step
    step = step/2 (alternate rounding up/down)
  until correct
# The Significance of Scaling

<table>
<thead>
<tr>
<th>N</th>
<th>N log(N)</th>
<th>N²</th>
<th>2^N</th>
<th>N!</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.0</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>8.0</td>
<td>16</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>15.5</td>
<td>36</td>
<td>64</td>
<td>720</td>
</tr>
<tr>
<td>8</td>
<td>24.0</td>
<td>64</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>33.2</td>
<td>100</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>58.6</td>
<td>225</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>86.4</td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>664.4</td>
<td>10,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>9965.8</td>
<td>1,000,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complexity Caveats

Small Problem Size

Same Complexity
# Comparing ListADT Implementations

## Time Requirements

Problem size \( N \) is number of items

<table>
<thead>
<tr>
<th></th>
<th>constructor</th>
<th>add (E) “at end”</th>
<th>add (int,E) “at pos”</th>
<th>contains (E)</th>
<th>size</th>
<th>Is Empty</th>
<th>get (int)</th>
<th>remove (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Singly-Linked List (SLL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Circular SLL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doubly-LL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CircularD LL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparing ListADT Implementations

Space Requirements

→ Problem size N is?

Array:

Singly-Linked List:

Circular Singly-Linked List:

Doubly-Linked List:

Circular Doubly-Linked List:
Comparing ListADT Implementations

Ease of Implementation

Array:

Singly-Linked List:

Circular Singly-Linked List:

Doubly-Linked List:

Circular Doubly-Linked List: