CS 367 - Introduction to Data Structures
Week 5, 2017

Homework 4 due 10 pm Friday, February 17th
Program 0 grades available
Program 1 quiz and survey
Program 2 due 10 pm Sunday, March 5th
Team Building 102

Last Week
Listnode class, chain of nodes, LinkedList Class
Linked List Variations: header node, tail reference
LinkedListIterator Class

This Week
Read: Complexity
Iterable and For-Each Loops (from last lecture’s outline)
More Linked List Variations
• double linking
• circular linking
Complexity
• concept, big-O notation
• analyzing algorithms practice
• analyzing Java code
• practice analyzing Java code
• best/worst cases
• significance of scaling
• complexity caveats
Comparing Complexity Analysis of ArrayList vs LinkedList

Next Week
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
Double and Circular Linking

Doubly-Linked Chains of Nodes

Circular Singly-Linked Chains of Nodes

Circular Doubly-Linked Chains of Nodes
Analyzing Algorithm Efficiency

Complexity

If problem size doubles and the number of operations:
Example: Complexity Analysis of Giving a Toast
N vs. Nlog(N) vs. N²

Complexity Analysis:

- 

-
Big-O Notation

Concept

some growth rate functions:

Simplifying Equations

Formal Definition
Complexity of Java Code

Basic operations

Sequence of statements

statement1;
statement2;
...
statementk;

If-else

if (cond) {
   //if sequence of statements
}
else {
   //else sequence of statements
}
Complexity of Java Code (cont.)

Basic loops
→ What is the problem size based on?

```java
for (i = 0; i < j; i++) {
    //sequence of statements
}
```

Nested loops
→ What is the problem size based on?

```java
for (i = 0; i < N; i++) {
    for (j = 0; j < M; j++) {
        //sequence of statements
    }
}
```

Loops with nested method calls (assume problem size based on N)

```java
for (i = 0; i < N; i++) {
    f1(i);  //assume O(1)
}

for (i = 0; i < N; i++) {
    f2(N);  //assume O(N)
}

for (i = 0; i < N; i++) {
    f3(i);  //assume O(i)
}
```
Practice - Complexity of Java Code

method1

→ What is the problem size based on?

```java
public void method1(int[] A) {
    for (int i = 0; i < A.length - 1; i++)
        method2(A, i);
}
```

method2

```java
public void method2(int[] B, int s) {
    for (int i = s; i < B.length - 1; i++)
        if (B[i] > B[i+1])
            method3(B, i, i+1);
}
```

method3

```java
public void method3(int[] C, int x, int y) {
    int temp = C[x];
    C[x] = A[y];
    C[y] = temp;
}
```
Practice - Complexity of Java Code

method4
⇒ What is the problem size based on?

```java
public void method4(int Q) {
    int sum = 0, R = 1000;

    for (int i = Q; i >= 1; i--)
        for (int j = 0; j < R; j++)
            sum += j;
}
```

method5
⇒ What is the problem size based on?

```java
public void method5(int X) {
    int tmp, arr[];

    arr = new int[X];
    for (int i = 0; i < X; i++)
        arr[i] = X - i;

    for (int i = 0; i < X - 1; i++) {
        for (int j = i; j < X - 2; j++) {
            if (arr[j] > arr[j+1]) {
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
```
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>N^2</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>construct</th>
<th>add (E) “at end”</th>
<th>add (int,E) “at pos”</th>
<th>contains (E)</th>
<th>size</th>
<th>isEmpty</th>
<th>get (int)</th>
<th>remove (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Array</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Singly-Linked List (SLL)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Circular SLL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Doubly-LL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CircularD LL</strong></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: