Week 15 - Monday

COMP 2100

Last time

- What did we talk about last time?
- Finished tries
- Substring searching

Questions?

Project 4

Review

Final Exam Format

- Roughly half short answer questions
- Roughly half programming
- Designed to take 90 minutes (50% longer than the previous exams)
  - But, you will have the full 120 minute time period
- The focus will be on the second half of the semester
- Look for things that were not covered on previous exams
- **Place:** Point 113
- **Time:** 10:15 - 12:15 p.m., Friday, 12/13/2024

Review up to Exam 1

Week 1

- Programming model
- Java
  - OOP
  - Interfaces
  - Exceptions
- Java Collections Framework

Week 2

- Big Oh Notation
  - Formal definition: $f(n)$ is $O(g(n))$ if and only if
    - $f(n) \leq c \cdot g(n)$ for all $n > N$
    - for **some** positive real numbers c and N
  - Worst-case, asymptotic, upper bound of running time
  - Ignore lower-order terms and constants
- Big Omega and Big Theta
- Abstract Data Types
- Array-backed list

Week 3

- Stacks
  - FILO data structure
  - Operations: push, pop, top, empty
  - Dynamic array implementation
- Queues
  - FIFO data structure
  - Operations: enqueue, dequeue, front, empty
  - Circular (dynamic) array implementation
- JCF implementations: **`Deque<T>`** interface
  - **`ArrayDeque<T>`**
  - **`LinkedList<T>`**

Week 4

- Linked lists
  - Performance issues
  - Single vs. double
  - Insert, delete, find times
- Special lists
  - Circular
  - Skip
  - Self-organizing
- Linked list implementation of stacks
- Linked list implementation of queues

Sample Problems

Running time

- What's the running time of the following code?

```
int count = 0;
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
        for (int k = 1; k <= n; k += j) {
            count++;
        }
    }
}
```

Array list class

```
public class ArrayList {
    private String[] array = new String[10];
    private int size = 0;
    …
}
```

Complete the following a method to insert a value in an arbitrary index in the list. You may have to resize the list if it doesn't have enough space.

```
public void insert(String value, int index)
```

BST and linked list classes

```
public class Tree {
  private static class Node
  {
    public String key;
    public Node left;
    public Node right;
  }

  private Node root = null;

  …
}
```

```
public class List {
  private static class
  Node {
      public String value;
      public Node next;
  }

  private Node head =
  null;
      …
}
```

Remove alternate nodes

- Write a method in the **`List`** class that will remove every other node (the nodes with even indexes) from a linked list

```
public void removeAlternateNodes()
```

Tree to Linked List

- Write a method that takes a binary search tree and returns an ordered linked list
  - Write the method in the **`Tree`** class
  - Assume you are given a linked list with an **`add()`** method that can add to the **front** of the list
- Hint: Use a reverse inorder traversal

Recursive method:

```
private static void toList(List list, Node node)
```

Proxy method:

```
public List toList() {
      List list = new List();
      toList(list, root);
      return list;
}
```

Upcoming

Next time…

- Review up to Exam 2
- Recursion
- Binary trees
- 2-3 and red-black trees
- Hash tables
- Graph basics
- Review Chapters 3 and 4

Reminders

- Bring a question to class Wednesday!
  - Any question about any material in the course
- **Fill out course evaluations!**
- **Keep working on Project 4**
  - **Due Friday**
- **Study for final exam**
  - **Friday, 12/13/2024 from 10:15 a.m. - 12:15 p.m.**