Data Structures and Algorithms - Lecture 1

February 2, 2019

Course Information

Essentials

- Course Website - atu-se.github.io/courses/dsa
- Primary Textbook - Introduction to Java Programming, 10th Edition (Liang)

Meeting Times

- Sunday @ 10:30 a.m.
- Tuesday @ 8:30 a.m.
- Wednesday @ 10:30 a.m.

How to Succeed

- Attend all lectures
- Take notes (not all materials will be distributed as slides)
- Ready and study your textbook
- Do all assignments
- Ask questions

Academic Integrity

- Read the academic integrity policy in the syllabus
- You are encouraged to discuss the topics among yourselves.
- Unless otherwise noted, your work should be your own and you should not share your work with others
- Copying or cheating on homework, exams, etc. may result in failing grades

Lecture 1: Recursion

Key Terms

- Recursion is a technique that leads to elegant solutions to problems that are difficult to program using simple loops.
- A recursive method is one that invokes itself.

Recursion Example: Factorial

Example: Factorial

$$
\begin{aligned}
0! &= 1 \\
n! &= n \cdot (n-1)! \qquad \text{when} \geq 1
\end{aligned}
$$

Example: Factorial

$2! = 2 \times 1! = 2 \times 1 = 2$

$3! = 3 \times 2! = 3 \times 2 \times 1! = 3 \times 2 \times 1 = 6$

Example: Factorial

```
import java.util.Scanner;

public class ComputeFactorial {
    /** Main method */
    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a non-negative integer:");
        int n = input.nextInt();
        // Display factorial
        System.out.println("Factorial of " + n + " is "
        + factorial(n));
    }
```

...

Example: Factorial

```
...
  /** Return the factorial for a specified number */
  public static long factorial(int n) {
      if (n == 0) // Base case
          return 1;
      else
          return n * factorial(n - 1); // Recursive call
  }
}
```

What-If

What if our factorial function was like this?

```
public static long factorial(int n) {
  return n * factorial(n - 1);
}
```

Example: Fibonacci

Example: Fibonacci

```
import java.util.Scanner;
public class ComputeFibonacci {
  /** Main method */
  public static void main(String args[]) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);
    System.out.print("Enter an index for the Fibonacci number: ");
    int index = input.nextInt();
    // Find and display the Fibonacci number
    System.out.println(
      "Fibonacci number at index " + index + " is " + fib(index));
  }
```

Example: Fibonacci 2

```
/** The method for finding the Fibonacci number */
public static long fib(long index) {
  System.out.println("Called fib");
  if (index == 0) // Base case
    return 0;
  else if (index == 1) // Base case
    return 1;
  else  // Reduction and recursive calls
    return fib(index - 1) + fib(index - 2);
  }
}
```

Example: Recursive Palindrome

Example: Recursive Palindrome

```
public class RecursivePalindrome {
  public static boolean isPalindrome(String s) {
    return isPalindrome(s, 0, s.length() - 1);
  }

  public static boolean isPalindrome(String s, int low, int high) {
    if (high <= low) // Base case
      return true;
    else if (s.charAt(low) != s.charAt(high)) // Base case
      return false;
    else
      return isPalindrome(s, low + 1, high - 1);
  }
```

Example: Recursive Palindrome 2

```
  public static void main(String[] args) {
    System.out.println("Is moon a palindrome? "
      + isPalindrome("moon"));
    System.out.println("Is noon a palindrome? "
      + isPalindrome("noon"));
    System.out.println("Is a a palindrome? " + isPalindrome("a"));
    System.out.println("Is aba a palindrome? " +
      isPalindrome("aba"));
    System.out.println("Is ab a palindrome? " + isPalindrome("ab"));
  }
}
```

Example: Recursive Selection Sort

Example: Selection Sort

- Find the smallest element in the list and swap it with the first element.
- Ignore the first element and sort the remaining smaller list recursively.

Example: Selection Sort

```
public class RecursiveSelectionSort {
  public static void sort(double[] list) {
    sort(list, 0, list.length - 1); // Sort the entire list
  }
```

Example: Selection Sort

```
public static void sort(double[] list, int low, int high) {
  if (low < high) {
    // Find the smallest number and its index in list(low .. high)
    int indexOfMin = low;
    double min = list[low];
    for (int i = low + 1; i <= high; i++) {
      if (list[i] < min) {
        min = list[i];
        indexOfMin = i;
      }
    }
```

Example: Selection Sort

```
    // Swap the smallest in list(low .. high) with list(low)
    list[indexOfMin] = list[low];
    list[low] = min;

    // Sort the remaining list(low+1 .. high)
    sort(list, low + 1, high);
  }
}
```

Example: Selection Sort

```
  public static void main(String[] args) {
    double[] list = {2, 1, 3, 1, 2, 5, 2, -1, 0};
    sort(list);
    for (int i = 0; i < list.length; i++)
      System.out.print(list[i] + " ");
  }
}
```