CS310: Advanced Data Structures and Algorithms

Fall 2021 Assignment 1

Due: Friday, Sep. 23, 2022 on Gradescope

Instructions

1. **Goal of this assignment — review of runtime analysis, recursion and Java.** I strongly encourage you to submit a printed solution to Gradescope. Handwritten solution will only be accepted if clearly legible!
2. Review of Java includes q. 7-9. You don't need to try to make the programs in questions 7-9 work online, but it would be a good idea to compose and run some test Java programs to make sure you have the right programming environment set up. Make sure it is compatible with the current runtime environment. I will be running the programming assignments with openJDK 11 or higher (I will not test the code for this assignment). OpenJDK 11 is compatible with Java 8.

Questions

1. You should learn to recognize and sum a geometric series. Try these:
   Note: You don't necessarily have to know exactly how to solve these equations. It's enough if you look it up in your calculus text or online. The important thing for me is that you know how to look these things up and understand how they generalize.

   (a) $\sum_{i=1}^{n=10} 2^i$.

   $$\sum_{i=1}^{n=10} 2^i = \sum_{i=0}^{n=10} 2^i - 1 = 2^{n+1} - 2 = 2^{11} - 2 = 2046$$

   (b) $\sum_{i=1}^{\infty}(2/3)^i$.

   $$\sum_{i=1}^{\infty}(2/3)^i = \frac{1}{1-\frac{2}{3}} = 3$$

2. How many binary digits are there in the numbers 2^{100}, 5^{100} and 10^{100} ? How are the answers to these three questions related? (**Hint:** this is a question about logarithms and change of base.)
3. (a) Show that $\log_a(x) = c * \log_b(x)$ for some constant c (expressed only in terms of the constants a and b). **Hint:** This is probably easier than you think... It follows quite directly from the log properties. You should, though, prove that it's true for any a and b and not prove by an example.

   (b) Calculate the ratio between the number of digits required to write a number in base 10 and the number of digits required to write the same number in base 2. Notice that this question relates to question 2 and 3a above.
4. (a) Order the following functions by growth rates:
   N, $\sqrt{N}$, $N^{1.5}$, N^2, $N \log N$, $N \log \log N$, $N \log^2 N$, $N \log(N^2)$, $2/N$, 2^N, $2^{N/2}$, 37, N^3, $N^2 \log N$.
   Indicate which functions grow at the same rate.

(b) Rank the following three functions: $\log N$, $\log(N^2)$, $\log^2 N$. Explain.

You should find all the mathematics you need in the class notes and in Kleinberg and Tardos, chapter 2. You may find it useful to remember that one way to compare the relative growth rates of $f(n)$ and $g(n)$ is to look at the ratio $f(n)/g(n)$ as $n \to \infty$. If that ratio approaches 0, then g grows faster than f: $f(n) = O(g(n))$. If it approaches infinity then f grows faster than g. If the ratio approaches a constant different from both 0 and ∞ than f and g grow at the same rate.

5. (a) Find a big-O estimate for the running time (in terms of n) of the following function (with explanation)

```
int mysterySum(int n) {
  int i, j, s=0;
  for(i=0; i < n; i++) {
    for(j=0; j < i; j++) {
      s += i*i;
    }
  }
}
```

(b) Is this version of mysterySum faster? Is the big-O analysis different?

```
int mysterySum1(int n) {
  int i, j, s=0;
  for(i=0; i < n; i++) {
    int i2 = i*i;
    for(j=0; j < i; j++) {
      s += i2;
    }
  }
}
```

This method does the same thing as the previous method with the only difference being that the mathematical operation is done separately to the increment of the sum. This operation doesn't affect runtime so it must be the same as the previous method.

(c) Replace the inner loop in mysterySum by an O(1) expression and compute the running time of the new program.

(d) Find a single O(1) expression giving the same result. **Hint:** Evaluate the function by hand (or compile and run the code) for a view values of n and try to see the pattern
Notice: You have to find the mathematical formula, not a piece of code. Start with the expression you derived in part c above (hint: It is a series) and find the sum of the series any way you want (including looking it up).

6. The following program computes 2^n:

```
int power2(int n) {
  if (n==0) return 1;
  return power2(n-1)+power2(n-1);
}
```

(a) Find a recurrence formula as we learned in class. Find the runtime. What is the big problem with this function? (hint: We discussed something similar in class).

(b) Introduce a small modification that makes the function run in linear time. Show why the runtime is linear.

(c) (bonus) The following function also calculates 2^n:

```
int power2New(int n) {
  if (n==0) return 1;
  if (n % 2 == 0) {
    int result = power2New(n/2);
    return result*result;
  }
  else
    return 2*power2New(n-1);
}
```

Explanation: If n is even, then $2^n = (2^{\frac{n}{2}})^2$, so we can calculate $2^{\frac{n}{2}}$ recursively and square, cutting half of n in one move. Otherwise, we resort to the previous method. Show that the runtime of power2New is logarithmic in n. Hint: It's easy to show it when n is even, but sometimes n is odd... The trick is to show that the entire function is logarithmic nonetheless.

7. **The stable matching algorithm:** Given the following rankings of students and hospitals like the one given in class:

Atlanta	Xavier	Yolanda	Zeus
Boston	Yolanda	Xavier	Zeus
Chicago	Xavier	Yolanda	Zeus

Xavier	Boston	Atlanta	Chicago
Yolanda	Atlanta	Boston	Chicago
Zeus	Atlanta	Boston	Chicago

   (a) Show that any stable match will match Zeus and Chicago (very easy. Almost copy-paste from the notes).

   (b) Show that it is true in any setting for any number of students/hospitals. In other words, if a student s is last on all of the hospital's list, and a hospital h is last on every student's list, they will end up matched to each other. This is also quite simple: Show that any other match would result in instability. Note that the proof should be for **any match**, not just the one produced by Gale-Shapley's algorithm.

8. **Designing a Java class:** A combination lock has the following basic properties: the combination (a sequence of three numbers) is hidden; the lock can be opened by providing the combination; and the combination can be changed, but only by someone who knows the current combination. Design a class with public methods `open` and `changeCombo` and private data fields that store the combination. The combination should be set in the constructor. Provide the java code as part of your submission, not as a separate file (just copy-paste the code to your text file). You may compile and run your code for testing but I will not- I will just look for overall design and understanding of the concept.

CombinationLock.java

```
public class CombinationLock{
    private int combination; // The lock's combination
    private boolean isOpen; // The lock's lock status

    // Constructor
    public CombinationLock(int combination) {
        this.combination = combination;
        this.isOpen = false;
    }
```

```
    // Opens the lock if the attempted combination matches the lock's combination
    public void open(int attempt) {
        if (attempt == combination) {
            isOpen = true;
        } else {
            System.out.printf("The combination %d is incorrect.\n", attempt);
        }
    }

    // Locks the combination lock
    public void lock() {
        this.isOpen = false;
    }

    // Changes the lock's combination
    public void changeCombo(int newCombination) {
        this.combination = newCombination;
        this.isOpen = false;
    }
}
```

9. **Designing an encapsulated Java class:** In the example `FrequencyCounter.java` (https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/FrequencyCounter.java), the word with highest frequency is found, along with its count. This result could be packaged up in one object with two instance variables "word" and "count". For example if "the" showed up 2034 times in the text, this object would have word "the" and count 2034. Create such a class, named WordUsage, with a constructor taking both the word and the count, and another constructor taking only the word and using count 1. Make the instance variables private for proper encapsulation of the objects. Give the class getters for word and count (i.e. getWord and getCount) and a setter for count but not word (setCount) and a mutator method named increment that adds one to the count. Don't implement equals or other Object methods here: this is a simple object meant for just carrying data from one place to another in our code. Note the strong Java convention that class names are capitalized, but method names and variables start with lower case. See pp. 84-85 of S&W for discussion of a similar class implementation. Here we can also mark the String instance variable as final because we can't change it once the object is created, since there is no setter for the word, and the instance variables are private, protected against direct access.

WordUsage.java

```
public class WordUsage {
    private final String word; // the current word being tracked
    private int count; // the number of times this word has been used

    // Constructors
    public WordUsage(String word, int count) {
        this.word = word;
        this.count = count;
    }

    public WordUsage(String word) {
        this.word = word;
        this.count = 1;
    }

    // Returns this tracked word
    public String getWord() {
```

```
        return this.word;
    }

    // Returns this word's count
    public int getCount() {
        return this.count;
    }

    // Sets this word's count
    public void setCount(int count) {
        this.count = count;
    }

    // Increments the count for this word
    public void increment() {
        this.count++;
    }
}
```

10. **Java questions:**

    (a) What is the difference between a final class and other classes? Why are final classes used?

    Final classes are a way for preventing the possibility of inheritance on that class. Meaning that no other classes can extend the final class. Final classes can be used to ensure that the program doesn't tamper with the object.

    (b) What is an interface? How does the interface differ from an abstract class? What members may be in an interface?

    An interface can be thought of an outline that other classes can implement. Interfaces can method signatures with no bodies (also known as abstract methods). Abstract classes can have abstract methods of any type as well as declare fields. You can extends multiple interfaces but one one abstract class.