CSCI 2235 – Solution/Grading Key
Programming Assignment 01 - Algorithm Analysis and Binary Search

Solution Key

Assigned: September 02, 2019
Due: September 13, 2019 @ 23:00h

1. Each student should have submitted a zip file which contains at least their src directory, the build.gradle file, and a PDF file containing the answers to part 1 and the results writeup for part 2 in MLA format.
2. If the above files are not found the following deductions can be made:
   - If there is no src directory, or they simply provided their .java files (-25 points)
   - If there is no PDF file (-25 points)

1 Part 1 - Algorithm Analysis

(2 Points each) Using the definition of big-oh notation, prove or disprove the following assertions (i.e., you must provide the $c > 0$ and $n^0 \geq 1$ that fulfills the definition, or give a formal argument for why the assertion is false).

1. $2n^3 - 7n^2 + 100n - 36$ is in $O(n^3)$

   for all $n \geq 1$

   $$\begin{aligned} 3n^3 - 7n^2 + 100n - 36 &\leq cn^3 \\ 3n^3 - 7n^2 + 100n - 36 &\leq 3n^3 + 100n \\ 3n^3 - 7n^2 + 100n - 36 &\leq 3n^3 + 100n^3 \text{ Since } n^3 > n \\ 3n^3 - 7n^2 + 100n - 36 &\leq 103n^3 \end{aligned}$$

   Therefore $2n^3 - 7n^2 + 100n - 36$ is in $O\left(n^3\right)$

2. $10n + 3\log(n)$ is in $O(n)$

   for all $n \geq 1$

   $$\begin{aligned} 10n + 3\log\left(n\right) &\leq cn \\ 10n + 3\log\left(n\right) &\leq 10n + 3\log\left(n\right) \\ 10n + 3\log\left(n\right) &\leq 10n + 3\log\left(2^n\right), \text{ Since } 2^n > n \text{ and log is base 2} \\ 10n + 3\log\left(n\right) &\leq 10n + 3n \\ 10n + 3\log\left(n\right) &\leq 13n \end{aligned}$$

   Therefore $10n + 3\log\left(n\right)$ is in $O\left(n\right)$

3. $n/1000$ is in $O(1)$

   for all $n \geq 1$

$$\begin{aligned} n/1000 &\leq c \\ n &\leq 1000 \end{aligned}$$

   n approaches infinity we cannot identify a constanc c which is always greater than n, therefore $n/1000$ is not in $O(1)$

4. $\log(n)^2 + \log(n)/30$ is in $O(\log(n)^2)$

   for all $n \geq 1$

$$\begin{aligned} \log(n)^2 + \log(n)/30 &\leq c\log(n)^2 \\ \log(n)^2 + \log(n)/30 &\leq \log(n)^2 + \log(n)/30 \\ \log(n)^2 &\leq \log(n)^2 \end{aligned}$$

   Therefore, $\log(n)^2 + \log(n)/30$ is in $O\left(\log(n)^2\right)$

5. $n^2/\log(n) + 3n$ is in $O(n^2)$

   for all $n \geq 2$

$$\begin{aligned} n^2/\log(n) + 3n &\leq cn^2 \\ n^2/\log(n) + 3n &\leq n^2 + 3n \text{ no division since log is base 2 and for any } n >= 2 \text{ division will only re} \\ n^2/\log(n) + 3n &\leq n^2 + 3n^2 \text{ Since } n^2 > n \\ n^2/\log(n) + 3n &\leq 4n^2 \end{aligned}$$

   Therefore $n^2/\log(n) + 3n$ is in $O(n^2)$.

(2 Points each) For each function below, provide the "tightest" big-oh bound. You can do this from the definition of big-oh if you are unsure of the answer, or you can use shortcuts and just provide the big-oh value.

6. $36n$ is in $O(n)$
7. $n^2/2 + 15n$ is in $O(n^2)$
8. $(n^2/4)(8/n)$ is in $O(n)$
9. $n + 10\log(n)$ is in $O(n)$
10. 87262 is in $O(1)$

(4 Points each) For each method below, provide the tightest big-oh running time.

```
public int m1FindLargest(int[] array) {
    if (array.length != 0) {
        int value = array[0];
        for (int i = 1; i < array.length; i++) {
            if (array[i] > value) {
                value = array[i];
            }
        }
        return value;
    }
    return -1;
}
```

$f(n) = 2n + 1$, for all $n > 1$:

$$\begin{aligned} 2n+1 &\leq cn \\ 2n+1 &\leq 2n+1 \\ 2n+1 &\leq 2n+n \\ 2n+1 &\leq 3n \end{aligned}$$

Therefore this method is bounded by $O(n)$.

```
public void m2PrintTriangle(int size) {
    for (int i = 1; i <= size; i++) {
        for (int j = 1; j <= 1; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

$f(n) = \frac{n(n+1}{2}$

$$\begin{aligned} \frac{n(n+1)}{2} &\leq cn^2 \\ \frac{n(n+1)}{2} &\leq \frac{n^2}{2} + \frac{n}{2} \\ \frac{n(n+1)}{2} &\leq \frac{n^2}{2} + \frac{n^2}{2} \\ \frac{n(n+1)}{2} &\leq n^2 \end{aligned}$$

$$\begin{aligned} \frac{n(n+1)}{2} &\leq cn \\ \frac{n^2}{2} + \frac{n}{2} &\leq n \\ \frac{n^2}{2} &\leq \frac{n}{2} \\ n &\leq 1 \end{aligned}$$

Therefore $f(n)$ is not bounded by $O(n)$ since as n approaches infinity we cannot find a constant that will always be larger, but we can see that $f(n)$ is bounded by $O(n^2)$ for all $n \geq 1$ with a constant, $c = 1$. There this method is bounded by $O\left(\frac{n(n+1)}{2}\right) = O\left(n^2\right)$

```
public void m3PrintBooks(String books[], int[] stars) {
    if (books.length == stars.length) {
        for (int i = 0; i < books.length; i++) {
            System.out.print(books[i] + "'s stars: ");
            for (int j = 0; j < stars[i]; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

$f(n) = n(c + s + 1)$, where s is the maximum number of stars, here we assume 5. Thus, $f(n) = (c + 6)n = cn$. Therefore this method is bounded by $O(n)$.

1.1 Grading

- The max value associated with this section is 25 points, but there are 32 points total available.
  - How this works is that you are to count only the first 25 points received. I would start by grading 6 through 13.
  - The code evaluations need only have the correct Big-O value, rather than a full proof. I have provided the proof in case anyone was unclear about the answer.
  - 6 - 10 need only provide the tightest Big-O bound. Any other answer is a loss of 2 points
  - 1 - 5 need the proof, my thought on this is that if the proof looks reasonable and does not violate any basic mathematics and their reasoning and final answer is sound then given them the 2 points. If it is a little off subtract 1, else give a 0.

2 Part 2

1. You will need gradle installed to evaluate their projects.
2. The algorithms are evaluated by executing the following command in the root directory of their project: `$ gradle test`
   - For each test that fails subtract 2.5 points
3. The remainder of this section is based on their writeup and an evaluation of their code.
   - 10 points: Their code in package `edu.isu.cs2235.algorithms.impl` is commented (2 points per file which has some comments at the top indicating their name and the function of the class). There is an additional 2 points for commenting their `edu.isu.cs2235.Driver` class.
   - 15 points: Their writeup is well written (contains paragraphs, minimal grammatical mistakes, and generally readable) presentation of a logical and sound description of the results of executing the algorithms. They should have included a chart depicting the results of their experiment and an analysis of these results discussing what they found and comparing their results of the algorithm.
     - 5 Points for the readability of the writeup
     - 5 Points for the writeup of the experiment
     - 5 Points for the discussion of the results
   - Note if they do not put their name at the top of the writeup, they lose 15 points for the writeup.