

Problem 1. Java Review

At this point, most of you should be comfortable enough to work with Java. Let's take some time to review a few concepts in Java so that we can limit our Java-related issues and, hence, focus on the algorithms when solving future Problem Sets.

- (a) What is the difference between a class and an object? Illustrate with an example.
- (b) Why does the `main` method come with a `static` modifier?
- (c) Give an example class (or classes) that uses the modifier `private` incorrectly (i.e., the program will not compile as is, but would compile if `private` were changed to `public`).
- (d) The following question is about Interfaces.
 - (d)(i) Why do we use interfaces?
 - (d)(ii) Give an example of using an interface.
 - (d)(iii) Can a method return an interface?
- (e) Refer to `IntegerExamination.java` in Coursemology. Without running the code, predict the output of the `main` method. Can you explain the outputs?
- (f) Can a variable in a parameter list for a method have the same name as a member (or static) variable in the class? If yes, how is the conflict of names resolved?

Problem 2. Asymptotic Analysis

This is a good time for a quick review of asymptotic big-O notation. For each of the expressions below, what is the best (i.e. tightest) asymptotic upper bound (in terms of n)?

$$f_1(n) = 7.2 + 34n^3 + 3254n$$

$$f_3(n) = 2^{4\log n} + 5n^5$$

$$f_2(n) = n^2 \log n + 25n \log^2 n$$

$$f_4(n) = 2^{2n^2+4n+7}$$

Problem 3. More Asymptotic Analysis!

Let f and g be functions of n where $f(n) = O(n)$ and $g(n) = O(\log n)$. Find the best asymptotic bound (*if possible*) of the following functions.

- (a) $h_1(n) = f(n) + g(n)$
- (b) $h_2(n) = f(n) \times g(n)$
- (c) $h_3(n) = \max(f(n), g(n))$
- (d) $h_4(n) = f(g(n))$
- (e) $h_5(n) = f(n)^{g(n)}$

Problem 4. Time complexity analysis

Analyse the following code snippets and find the best asymptotic bound for the time complexity of the following functions with respect to n .

- (a)

```
public int niceFunction(int n) {
    for (int i = 0; i < n; i++) {
        System.out.println("I am nice!");
    }
    return 42;
}
```
- (b)

```
public int meanFunction(int n) {
    if (n == 0) return 0;
    return 2 * meanFunction(n / 2) + niceFunction(n);
}
```
- (c)

```
public int strangerFunction(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            System.out.println("Execute order?");
        }
    }
    return 66;
}
```
- (d)

```
public int suspiciousFunction(int n) {
    if (n == 0) return 2040;

    int a = suspiciousFunction(n / 2);
    int b = suspiciousFunction(n / 2);
    return a + b + niceFunction(n);
}
```

```

(e) public int badFunction(int n) {
    if (n <= 0) return 2040;
    if (n == 1) return 2040;
    return badFunction(n - 1) + badFunction(n - 2) + 0;
}

(f) public int metalGearFunction(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < i; j *= 2) {
            System.out.println("!");
        }
    }
    return 0;
}

```

Problem 5. Another Application of Binary Search (Optional) Given a sorted array of $n - 1$ unique elements in the range $[1, n]$, find the missing element? Discuss possible naive solutions and possibly faster solutions.