CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 3

For: January 24–January 28

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 it is, but would compile if `private` was 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`, which can be found in the same folder as this PDF. 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)?

(a) $f_1(n) = 7.2 + 4n^3 + 3254n$

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

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

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

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;
}
```
n

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

(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;
}
```
n^2

(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);
}
```
nlogn

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

(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;
}
```
nlogn

(g)
```
public String simpleFunction(int n) {
    String s = "";
    for (int i = 0; i < n; i++) {
        s += "?";
    }
    return s;
}
```
n^2

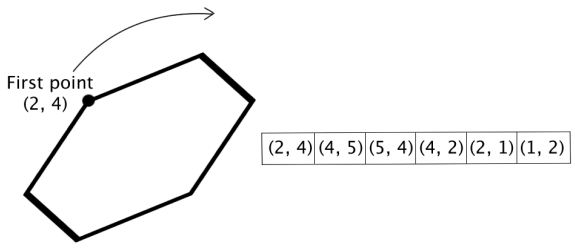
Problem 5. Another Application of Binary Search

Given a sorted array of $n-1$ unique elements in the range $[1, n]$, how would you find the missing element? Discuss possible naive solutions and possibly faster solutions.

Problem 6. Yet Another Application of Binary Search

(Optional) Given an array of n x and y-coordinates of an n-sided convex polygon in clockwise order, find a bounding box around the polygon. Discuss possible naive solutions and possibly faster solutions. A convex polygon is a polygon where all interior angles are less than 180 degrees.

An example of such an array is shown below:



Starting tactic is just to go the full loop and keep track of max. But I think you can jump by n/4 in any direction and find the gradients around it (max point will have gradients of different signs). If both the current location and n/4 location hav same nature of gradients around them (both double positive / both double negative), then keep going in that direction. Otherwise, if one is both positive and the other is both negative, a maxmin point has to be in between.