CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 3

For: January 27–January 31

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 object with an interface type?

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

(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. Application of Binary Search

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

Problem 5. Another Application of Binary Search

You have n piles of homework and the i^{th} pile has $piles[i]$ pieces of homework. Unfortunately, you realised you have h hours left before all your homework is due. In a moment of panic, you try to figure out the rate k (that is, the *"pieces of homework"-per-hour*) at which you need to do your homework at in order to finish everything on time. Assume we only want integer rates. ($k \in \mathbb{Z}$, nothing fractional.)

Here's how you plan to go about it: At every hour, you choose a pile of homework and start clearing pieces of homework from that pile. If the pile has less than k pieces of homework remaining, you decide to just finish that pile itself, and not start on the next pile yet during the same hour. We all need to take breaks after all, *right?*

To maintain your sanity, you want to minimise the number of pieces of homework you do per hour, i.e. k, while still finishing all piles of homework in time. You can assume that there exists a k that allows you to complete all piles of homework within h hours given the above constraints. Discuss how you can find the minimum integer k such that you can finish all your homework within h hours.

Here's an example: Let's say you had 10 hours, the piles are of size $[1, 7, 2, 3, 5]$. If we set $k = 7$, then we would take 5 hours. Each pile takes 1 hour to finish (since we finish 7 pieces per hour and there are 5 piles). 5 is less than our hour limit of 10. So $k = 7$ is viable.

If we set $k = 1$, then it would take $1 + 7 + 2 + 3 + 5 = 18$ hours (the first pile takes an hour, but the second pile takes 7, and so on), which is more than our allowance of 5 hours.

If we set $k = 3$, it would take $1 + 3 + 1 + 1 + 2 = 8$ hours. (1 hour to solve the first pile, 3 hours to solve the second pile, and so on)

If we set $k = 2$, it would take $1 + 4 + 1 + 2 + 3 = 11$ hours. (1 hour to solve the first pile, 2 hours to solve the second pile, and so on)

So minimum possible integer k in this example is 3, not 2.

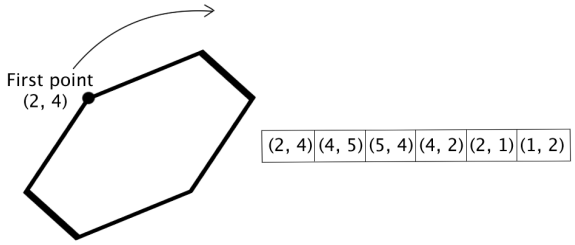
A naive solution would be to try values k from 1 onwards, and checking if for that value k whether it is viable or not. If it is, we stop. Otherwise, we try for k being one larger.

Question: Is there a more efficient way to find k? What is the algorithm?

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:



A *bounding box* is an axis-aligned rectangle (defined by 4 corner points) such that the entire polygon is contained within the box.

For the example above, the bounding box is defined by the 4 corner points: $(1, 1)$, $(1, 5)$, $(5, 1)$ $(5, 5)$.

Can you come up with a solution that takes $\Theta(n)$ time? Can you come up with something faster?