CS3000: Algorithms & Data — Spring 2019 — Paul Hand

Homework 3
Due Wednesday 2/6/2019 at 2:50pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Wednesday 2/6/2019 at 2:50pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden. You may use the internet for general research and learning pertaining to class material.

Problem 1. *Finding repeats*

Given a sorted list of numbers A, your task is to find the number of occurrences of a provided number t. Develop an algorithm to do so that runs in $O(\log n)$ time, where n is the length of A. Write your algorithm using pseudo code and establish its run time.

Solution:

Problem 2. *Computer Graphics*

Suppose you have a finite collection of vertical line segments that all have different x coordinates. Your goal is to design an algorithm that determines which parts of which line segments need to be rendered for a viewer looking from far away on the right. Ignore aspects of perspective; that is, a line is visible if some point is such that no point on any other line is directly to its right. The input data is provided as a list with elements of the form $(\text{index}, x, y_{\text{min}}, y_{\text{max}})$. Here, "index" refers to a unique identifier for each line from the input. The output of your algorithm is also a list with elements of the form $(\text{index}, x, y_{\text{min}}, y_{\text{max}})$, where any index may be repeated multiple times.

1. Suppose you have two lists of $O(n)$ visible line segments of the form $(\text{index}, x, y_{\text{min}}, y_{\text{max}})$, where each list is sorted in y (meaning that y_{max} of the ith entry is at least y_{min} of the ith entry, which is at least y_{max} of the $i-1$st entry, for all i). Suppose that the x-coordinates of one list are all less than the x-coordinates of the second list. Show how to form a new sorted list of all visible line segments in $O(n)$ time. **Write your algorithm using pseudocode and establish its run time**. Hint: you may need to break up a line segment into multiple parts when another line segment is blocking part of it.

   Solution:

2. Design a divide and conquer algorithm to determine which parts of which line segments need to be rendered for a viewer looking from the right. Your algorithm should have time complexity $\Theta(n \log n)$. **Write your algorithm using pseudocode and establish its run time.**

   Solution:

Problem 3. *Variant of the Master Theorem for Recurrences*

Suppose you have a recurrence of the form $T(n) = aT(n/b) + n^d(1 + \log n)$, where a, b, d are positive integers.

1. Determine the asymptotic scaling of T in the case of $a/b^d < 1$. Express your answer as Θ of some function of n.

   Solution:

2. Consider the case of $a/b^d = 1$. Show that $T(n) = O(n^d \log^2 n)$.

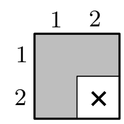
   Solution:

Problem 4. *Tiling*

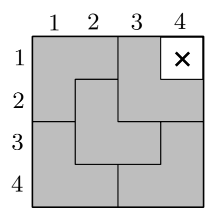
We are retiling the floor of ISEC, which has dimension $2^n \times 2^n$, and one square is reserved to be occupied by the statue of a wealthy donor. The location of the statue can be at any location (a, b) with $1 \leq a, b \leq 2^n$ that the donor tells us. The rest of the squares are tiled by L-shaped pieces, each covering three squares.

Here are two examples:

- If $n = 1$, and $(a, b) = (2, 2)$, we could cover the floor using a single L-shaped piece:



- If $n = 2$, and $(a, b) = (1, 4)$, we could cover the floor using five L-shaped tiles:



(a) Design a divide-and-conquer algorithm that takes as input the values n, a, b and outputs a list of the locations and orientations of $(4^n - 1)/3$ tiles.[1] Give pseudocode for your algorithm.

Solution:

(b) Write a recurrence for the running time and solve it using any method that we've covered in class.[2]

Solution:

[1] **Hint:** divide into four quadrants, each of dimension $2^{n-1} \times 2^{n-1}$, place a single tile so that each quadrant has one square covered, and then recurse.

[2] **Hint:** You may find it helpful to introduce a new variable $m = 2^n$ and solve for the running time as a function of m, then substitute to get the running time as a function of n.