Foundations of Algorithms Homework 1 SU24

Strategies for solving ***Collaborative Problem*** problems are to be formed by your assigned group, with each member *participating and contributing fully*. All answers must be your own work, in your own words, even on collaborative problems—collaborate on strategies only. Individual participation will be evaluated through collaborative assessments, submitted when the homework is due.

1. [15 points total] Describe the time complexity of the following linear search algorithm. Choose the tightest asymptotic representation, from Θ, O, or Ω, and argue why that is the tightest bound.

Algorithm 1. Linear Search

Input: sorted array A (indexed from 1), search item x
Output: index into A of item x if found, zero otherwise

```
function LINEAR-SEARCH($x$,$A$)
    $i = 1$                                  ▷ Arrays typically start at index 1
    $n =$len$(A)$
    while $i \leq n$ and $x \neq A[i]$ do
        $i = i + 1$
    if $i \leq n$ then
        $loc = i$
    else
        $loc = 0$
    return $loc$
```

2. [25 points total] This problem was posed by Professor Hamid Sarbazi-Azad, who invented an algorithm that he called *stupid sort.* For this problem, we assume it sorts integers in order from least to greatest. Here's a version of it:

Algorithm 2. Stupid Sort

Input:array A of n integers (indexed from $0..n-1$)

```
function STUPID-SORT($A$)
    $i = 0$
    while $i < n - 1$ do
        if $A[i] > A[i + 1]$ then
            swap $A[i]$, $A[i + 1]$
            if $i > 0$ then
                $i = i - 1$                  ▷ move backward
        else
            $i = i + 1$                      ▷ move forward
```

The running time of this algorithm depends on its input. Keep that fact in mind for your answers.

(a) [10 points] In terms of input length n, use **asymptotic notation** to express the best case running time of stupid sort, **and** give an example input of 5 integers that would cause this case to happen.

(b) [10 points] Give the *worst* case running time of stupid sort in asymptotic notation, **and** give an example input of 5 integers that would cause this case to happen.

(c) [5 points] Using Θ, Ω, or big-O notation, how would you express the general running time of this algorithm? Briefly explain why. (HINT: if it takes the same time to run everywhere, then it's a Θ relationship. Otherwise, it's either O if there only is a worst case that bounds the behavior from above, or Ω if there only is a best case that bounds the behavior from below.)[1]

[1] We try to sneak a little learning into these homework problems. Asymptotic notation is the most important thing to take away from this course.

3. [10 points total] This problem was posed by a student in a group discussion. Often when writing code, we see a double-nested loop pattern. In some cases, the inner loop runs a constant number of times. For example:

Algorithm 3. Double Loop

```
Input: array $A$ of $n$ items, some constant $k$
function DOUBLE-LOOP($A$)
    for $i = 1$ to $n$ do
        for $j = 1$ to $k$ do
            some costly calculation involving only $A_i$
            some calculation involving $A_i$ and $A_j$
    return something
```

(a) [5 points] Using the tightest asymptotic representation, from Θ, O, or Ω, derive the asymptotic time complexity of the loop above.

(b) [5 points] Suppose you are running this loop on a low-powered system, e.g., an Arduino board. What change could you make to the logic above to speed up the operation, and what overall improvement would you gain in running time?

4. [10 points total]

Accepted Master Method

For all problems on the Master Theorem, you are to use the method as given in the textbook, and you must show all justifications for the case (1, 2, or 3) that you've chosen; if something says "for any constant ϵ," show us actual values. If something says "$f(n) < O(n^2)$," show us the definition of $O(n)$ and give us values for each constant that satisfy the definition.

Use the Master Theorem to find the asymptotic bounds of $T(n) = 4T(n/4) + n^2$.

5. [10 points total] The Fibonacci sequence is one where the ith value F_i is computed as the sum of the values F_{i-1} and F_{i-2}, where the initial two values F_1 and F_2 are 1 and 1, respectively. Suppose someone used different initial values F_1 and F_2, and told you the 5th number is 20. They also told you that the two values are both positive integers, and that $F_2 \geq F_1$. What initial values did they use, and what is the next number in this *Fink-onacci*[2] series? Remember to show your work.

Programming Assignment Algorithm

The following question asks you to write an algorithm for an upcoming programming assignment. We want you to think ahead about how to solve the problem. You will implement this algorithm (with any corrections) in that upcoming programming assignment.

6. [30 points total] ***Collaborative Problem:*** In Programming Assignment 1 (PA1), you will implement a way of finding the closest pair of points in an xy (Euclidean) plane. For this problem, work as a group to come up with two algorithms—a *simple* algorithm that runs in $\theta(n^2)$ time, and *improvements* to that algorithm. Like any collaborative problem, you may confer on strategies, but your writeup, including your pseudocode, talkthrough, and explanation, must be your own. You will be expected to implement your submitted algorithm in Python for the programming assignment.

If you need a template for writing algorithms, check this out at Overleaf.com, an online LaTeX editor.[3] (Even if you're using Word, you can format at Overleaf and then paste a screenshot.)

[2]Prof. Fink wrote this one.
[3]Full URL: https://www.overleaf.com/read/gsjhctdjbbsw

(a) [8 points] Consider a set, P, of n points $(x_1, y_1), \ldots, (x_n, y_n)$, in a two dimensional plane. Write a simple algorithm that runs in $\theta(n^2)$ time, to find the closest pair of points in the xy plane among these n points. Your output should be the closest pair $(x_i, y_i), (x_j, y_j)$ and the associated Euclidean distance. The Euclidean distance between two points (x_i, y_i) and (x_j, y_j) is defined as $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Write your algorithm in proper pseudocode, and include a small narrative of your algorithm. The algorithm should reference certain line numbers as necessary, and derive the runtime analysis showing why it is $\theta(n^2)$.

(b) [8 points] Describe at least two improvements to the above algorithm that will make it run faster. Certain improvements can cut down overall work factor, and others can reduce the number of calls to the dreadfully slow square root function. (An English description is fine - don't have to change the pseudocode.) For each improvement, give the runtime after having applied the improvement; e.g., if something cuts it in half, then explain that it now runs in $n^2/2$ steps (but still remains $\theta(n^2)$, just with a different constant c) and so on. If there is an asymptotic change (e.g., θ to O), state that also and explain.

(c) [14 points] The CLRS textbook (3rd ed - see materials in the assignment page) sketches out an algorithm that runs in $O(n \log n)$ time, in section 33.4. It is a raw sketch, not a formal algorithm, so for this problem, complete the algorithm filling in all missing details. We expect to see a fully developed algorithm, with a couple permitted shortcuts:

- If you need a sort function, just assume one exists, e.g., $X_L \leftarrow \mathrm{SORTED}(\mathrm{P_L}, \ldots)$ where you show me the sorting key in some way. I don't want to see pseudocode for MergeSort.
- Don't show the logic for any minimum function, just assume it exists. Again, $\delta \leftarrow \min(\delta_L, \delta_R)$ is fine.

As an aid to understanding this algorithm, see this video from former student Samuel Nathanson[4] that visually explains how it works.

[4]Full URL: https://www.youtube.com/watch?v=-RWvTzJlE_I