Northeastern University
CS5800 – Algorithms
Spring 2024 Semester, Jonathan Mwaura

Problem Set #1

This Problem Set is due at **11:30PM on Monday, Jan** 22^{nd}, 2023, and will be submitted on GRADESCOPE.

This Problem Set will be marked out of 50. There are five problems, each worth ten marks.

Please type (or neatly handwrite) your solutions on standard 8.5×11 paper, with your name at the top of each solution. Ensure that you submit you solutions in one file PDF file on Gradescope. **each problem sets solution should be on in its own individual page, Gradescope will help ensure you submit each solution under its correct problem number**

While a solution must be absolutely perfect to receive full marks, I will be generous in awarding partial marks for incomplete solutions that demonstrate progress.

So that there is no ambiguity, there are two non-negotiable rules. A violation of either rule constitutes plagiarism and will result in you receiving an F for this course.

(a) If you meet with a classmate to discuss any of the Individual Problems, your submission must be an individual activity, done in your own words, away from others. The process of finding a solution might take 3 - 5 iterations or even more BUT you learn from all these attempts and your confidence grows with each iteration.

(b) These problem sets might seem hard on a first look. They are designed to be so. We learn by attempting problems, struggling through them and coming on top. I encourage you to make this learning exercise worth your while. What do I mean? Open the problem sets as early as you get them, then do not look at hints or answers any where (including on the internet and consulting other students for direct answers), give it the best shot you can. If you get stuck come to Professor or TA's office hour and we shall be glad to listen to your rationale and work with you till you are able to tackle the problem sets.

Problem #1

Let $f(n)$ and $g(n)$ be two functions, defined for each positive integer n.

By definition, $f(n) = O(g(n))$ if there exist positive constants c and n_0 for which $0 \leq f(n) \leq cg(n)$ for all integers $n \geq n_0$.

To prove that $f(n) \neq O(g(n))$ one must prove that no such constants c and n_0 exist.

(a) Let $f(n) = n^2 + 2n + 3$. Prove that $f(n) = O(n^2)$ and $f(n) \neq O(n)$.

(b) Let $f(n) = n \log n + 100n$. Prove that $f(n) = O(n \log n)$ and $f(n) \neq O(n)$.

NOTE: in this course, we will assume $\log n = \log_2 n = \lg n$.

(c) Let $f(n) = 2n^2 + 4$ and $g(n) = 4n^2 + 2$. Prove that $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

(d) Let $f(n)$ and $g(n)$ be any two functions for which $f(n)$ and $g(n)$ are positive numbers, for each integer $n \geq 1$.

Prove or disprove: at least one of these two statements *must* be true:

$$f(n) = O(g(n)), \quad g(n) = O(f(n)).$$

Clearly and carefully justify your answer.

Problem #2

Let $f(n)$ and $g(n)$ be two functions, defined for each positive integer n.
To prove that $f(n) \neq O(g(n))$ one must prove that no such constants c and n_0 exist.

For each of these questions, show your step by step work

(a) Prove that $2^{n+1} = O(2^n)$.

(b) Prove or disprove: $2^{2n} = O(2^n)$?.

(c) Let $f(n) = lg(lg^k n)$ and $g(n) = lg^k(lgn)$. Prove which one is asymptotically larger.

(c) Let $f(n) = lg_3 n$ and $g(n) = lg_9 n$. Prove the relationship between $f(n)$ and $g(n)$ in terms of upper bound (big O), lower bound (Ω) and tight bound (Θ)

Problem #3

Consider an array, A, whose contents is integer values. Given a particular threshold value t, an event E between indices $i < j$ is a critical event if $a_i > t * a_j$.

In this problem, write a full program that outputs the number of critical events for an arbitrary array of integers and any arbitrary threshold value t.

(a) Submit the code of your working program (You can use any programming language). Note, you need to upload your source file (TA's will download and run the code- the code can be upload in Canvas)

(b) Submit a screen capture showing that your program outputs correct values. You only need to repeat over 2 different arrays running with 2 different threshold.

(c) Perform an analysis of your algorithm and report its time complexity.

Problem #4

LeetCode (https://www.leetcode.com) is a popular website for Northeastern MSCS students, especially when preparing for job interviews.

There are over a thousand ”coding challenges” from which students can practice and improve their skills in Algorithm Analysis and Design, and the website supports numerous programming languages, including C, Java, and Python.

There are two parts to this weeks programming problem, see below:

(a) if you do not have an account with Leetcode, please create one. Please submit an image here showing your account.

(b) For this part, search for Leetcode Problem 35-Search Insert Position, you can also access the problem by visiting this page directly: https://leetcode.com/problems/search-insert-position. Solve this problem and then:

  (i) Provide a screen capture image that shows your has been accepted

  (ii) Provide a written run time analysis of your method (note according to this problem, your solution must run in $O(logn)$, also provide your proof of correctness for the solution based on Loop invariants.

  (iii) Submit you source code in Canvas (it should be as .py, .java, .cpp etc)

Problem #5

Let $\{a_1, a_2, \ldots, a_n\}$ be an unsorted list of n numbers.

Bubble Sort is a well-known sorting algorithm that works as follows.

Iteration #1: Compare the first two numbers. If the first number is bigger than the second, then swap the two numbers. Compare the second and third numbers, and swap them if necessary. Keep doing this until we have compared the final two numbers. (After this iteration, can you see why the largest number is guaranteed to be at the end?)

Iteration #2: Start from the beginning, comparing the first two numbers, and repeating the same process as above. But this time we only look at the first $n-1$ numbers, since we know the final number is already in the right position. (After this iteration, the two largest numbers are guaranteed to be at the end.)

We keep proceeding until the entire list has been sorted. Here is the pseudocode of Bubblesort.

```
for i = n down to 1
    for j = 1 to i-1
        if a[j]>a[j+1]
            swap(a[j],a[j+1])
```

This sorting algorithm is known as Bubble Sort, because after each complete iteration the largest unsorted number "bubbles" to the end of the list. For example, if the initial list is $\{1, 7, 4, 5, 2\}$, we have $\{1, 4, 5, 2, \mathbf{7}\}$ after the first iteration, $\{1, 4, 2, \mathbf{5}, \mathbf{7}\}$ after the second iteration, $\{1, 2, \mathbf{4}, \mathbf{5}, \mathbf{7}\}$ after the third iteration, and $\{1, \mathbf{2}, \mathbf{4}, \mathbf{5}, \mathbf{7}\}$ after the fourth and final iteration. We now know that the list is sorted.

(a) Demonstrate the Bubble Sort algorithm on the input list $\{4, 3, 2, 1, 5\}$. Clearly show your steps.

(b) Let $C(n)$ and $S(n)$ be the total number of comparisons and swaps required by Bubble Sort when the input list has n numbers. For example, in the list $\{1, 7, 4, 5, 2\}$ above, Bubble Sort requires 10 comparisons and 5 swaps.

Suppose the input list is $\{n, n-1, n-2, \ldots, 3, 2, 1\}$, where the numbers appear in reverse order. In this worst-case scenario, determine the exact formulas for $C(n)$ and $S(n)$. Clearly show all of the steps in your proof.

(c) Determine a precise "loop invariant" for Bubble Sort, clearly stating your Initialization, Maintenance, and Termination statements. Prove that your loop invariant holds, clearly and carefully justifying each step in your proof.

(d) Consider a random permutation of $\{1, 2, 3, \ldots, n\}$. Determine an exact formula for the average expected number of swaps required by Bubble Sort. Clearly and carefully justify your answer.