Problem Set 1

Data Structures and Algorithms, Fall 2022

Due: September 15 23:59:59 (UTC+8), mail to `ai-dsalg-ps@chaodong.me`.

Problem 1

Recall the INSERTIONSORT algorithm we discussed in class. We have argued why it always terminates on any input, we have also informally argued why it always returns the correct answer on any input. In this exercise, you are asked to *formally* prove that the INSERTIONSORT algorithm always correctly sorts the input. *(Hint: You can prove the correctness of loop invariant we discussed in class via induction. But in the inductive step, you may need to come up with another loop invariant to show the effectiveness of the **while** loop. That is, to prove the inductive step, you need to state and prove another loop invariant.)*

Problem 2

The *greatest common divisor* of positive integers x and y is the largest integer d such that d divides x and d divides y. Suppose $\mathrm{GCD}(x, y)$ returns the greatest common divisor of x and y, where $x, y \in \mathbb{N}^+$ and $x > y$. Then, the following equality is true:

$$\mathrm{GCD}(x, y) = \mathrm{GCD}(y, x \bmod y)$$

Effectively, the above equality suggests that we can reduce the problem of finding the greatest common divisor of x and y to another "smaller" problem: find the greatest common divisor of y and $(x \bmod y)$. Now, build upon this observation to derive an algorithm that can efficiently compute $\mathrm{GCD}(x, y)$. You must prove the correctness of your algorithm. (That is, it always terminates within finite time, and always returns correct answer upon termination.)

Problem 3

(a) Prove or give a counterexample: for every positive constant c and every function f from non-negative integers to non-negative reals, $f(cn) \in \Theta(f(n))$.

(b) Prove or give a counterexample: for every function f from non-negative integers to non-negative reals, $o(f) = O(f) - \Theta(f)$. Here "$-$" denotes the set minus operation.

Problem 4

Sort the following functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You do *not* need to prove your answer. To simplify notation, write $f(n) \ll g(n)$ to denote $f(n) \in o(g(n))$ and $f(n) = g(n)$ to denote $f(n) \in \Theta(g(n))$.[1]

[1] Read Section 3.2 of CLRS for the definition of function $\lg^* n$. Also, in this exercise, $\lg(n)$ means $\log_2(n)$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{3})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(9/8)^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg \lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

Problem 5

Explain how to implement a FIFO queue using two stacks. You should give a brief overview of your implementation, then provide pseudocode for ENQUEUE and DEQUEUE operations. Assuming PUSH and POP each takes $\Theta(1)$ time, analyze the running time of ENQUEUE and DEQUEUE.

Problem 6

Design a MINSTACK data structure that can store comparable elements and supports stack operations PUSH(x), POP$()$, as well as the MIN$()$ operation, which returns the minimum value currently stored in the data structure. All operations should run in $O(1)$ time in your implementation. (You may assume there exists a STACK data structure which supports PUSH and POP, and both operations run in $\Theta(1)$ time.) You should give a brief overview of your MINSTACK data structure, then provide pseudocode for each of the three operations. You should also discuss the space complexity of your implementation.