Problem Set 1

Data Structures and Algorithms, Fall 2020

Due: September 17, in class.

Problem 1

Consider the following algorithm to sort n numbers stored in an array A. First find the smallest element of A and swap it with the element stored in $A[1]$. Then find the second smallest element of A and swap it with the element stored in $A[2]$. Continue in this manner for $n-1$ times and we are done.

(a) Give the pseudocode of this algorithm.

(b) Give the best-case and worst-case running times of this algorithm in Θ-notation.

(c) What loop invariant does this algorithm maintain? State it, prove it, and then use it to show the correctness of the algorithm.

Problem 2

Given $c_0, c_1, \cdots, c_n$ and a value for x, we can evaluate a polynomial

$$P(x) = \sum_{k=0}^{n} c_k x^k = c_0 + x(c_1 + x(c_2 + \cdots + x(c_{n-1} + xc_n)\cdots))$$

using the following algorithm:

`PolyEval`$(x, c_0, c_1, \cdots, c_n)$

1: $y \leftarrow 0$
2: **for** $(i = n$ **downto** $0)$ **do**
3: $\quad y \leftarrow c_i + xy$

(a) Give the runtime of `PolyEval` in Θ-notation.

(b) What loop invariant does `PolyEval` maintain? State it, prove it, and then use it to show the correctness of `PolyEval`.

Problem 3

(a) Let $f(n) : \mathbb{N}^+ \mapsto \mathbb{R}^+$ and $g(n) : \mathbb{N}^+ \mapsto \mathbb{R}^+$ be two functions, prove that $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$.

(b) Let $a \in \mathbb{R}$ and $b \in \mathbb{R}^+$ be two constants, prove that $(n+a)^b = \Theta(n^b)$.

(c) Among asymptotic notations $o, O, \omega, \Omega, \Theta$, which can be used to describe the relationship between $\sqrt{n}$ and $n^{\sin(n)}$? You do not need to prove your answer.

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 mean $f(n) = o(g(n))$ and $f(n) = g(n)$ to mean $f(n) = \Theta(g(n))$.

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(3/2)^n$	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	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	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.

Problem 7

Design a RANDOMQUEUE data structure. This is an implementation of the QUEUE interface in which the `remove` operation removes an element that is chosen uniformly at random among all the elements currently in the queue. You may assume you have access to a `random` function which takes a positive integer as input: `random(x)` returns a uniformly chosen random integer in $[1, x]$, in $O(1)$ time. You should give a brief overview of your data structure, then provide pseudocode for `add` and `remove` operations. You should also discuss the time complexity of `add` and `remove` operations. (To get full credit, `add` and `remove` operations should run in $O(1)$ time in your implementation. You may assume the number of elements in the queue never exceeds N.)

Problem 8

You are given an infix expression in which each operator is in $\{+, \times, !\}$ and each operand is a single digit positive integer. Write an algorithm to convert it to postfix. (For example, given "$2 + 3 \times 3!$", whose value is 20, your algorithm should output "$233! \times +$".) You should give a brief overview of your algorithm, then provide pseudocode, and finally discuss its time complexity. (To get full credit, you your algorithm need to have $O(n)$ time complexity.)