University of Waterloo
CS240E, Winter 2024
Written Assignment 1

Due Date: Tuesday, January 23, 2023 at 5pm

Be sure to read the assignment guideliness (`https://student.cs.uwaterloo.ca/~cs240e/w24/assignments.phtml#guidelines`). Submit your solutions electronically to MarkUs as **individual** PDF files named a1q1.pdf, a1q2.pdf, … (one per question).

Ensure you have read, signed, and submitted the **Academic Integrity Declaration** AID01.TXT.

Grace period: submissions made before 11:59PM on Jan. 23, will be accepted without penalty. Please note that submissions made after 11:59PM **will not be graded** and may only be reviewed for feedback.

Question 1 [5 marks]

There are two different definitions of 'little-omega' in the literature (to distinguish them, we will call them ω_0 and ω_1 here). Fix two functions $f(x), g(x)$. We say that

- $f(x) \in \omega_0(g(x))$ if for all $c > 0$ there exists $n_0 > 0$ such that $|f(x)| \geq c|g(x)|$ for all $x \geq n_0$, and
- $f(x) \in \omega_1(g(x))$ if $g(x) \in o(f(x))$.

Show that these two definitions are equivalent, i.e., $f(x) \in \omega_0(g(x))$ if and only if $f(x) \in \omega_1(g(x))$. Your proof must be from first principle, i.e., directly using the definitions (do not use the limit-rule). Note that $f(x), g(x)$ are not necessarily positive.

Question 2 [3+3+3=9 marks]

Consider the following (rather strange) code-fragment:

Algorithm 1: mystery (int n)

Input: $n \geq 2$

$L \leftarrow \lfloor \log(\log(n)) \rfloor$

print all subsets of $\{1, \ldots, 2^L\}$

For example, for $n = 17$, we have $\log 17 \approx 4.08$ and $\log(4.08) \approx 2.02$, so $\log\log(17) \approx 2.02$ and $L = 2$ (and we print the 16 subsets of $\{1, \ldots, 4\}$). This question is really asking about the run-time of `mystery`, but to avoid having to deal with constants, define $f(n)$ to be the number of subsets that we are printing when calling `mystery` with parameter n.

(a) Show that $f(n) \in O(n)$.

(b) Show that $f(n) \in \Omega(\sqrt{n})$.

(c) Prof. Conn Fused thinks that $f(n) \in \Theta(n^d)$ for some constant d. (By the previous two parts, necessarily $\frac{1}{2} \le d \le 1$.) Show that Prof. Fused is wrong, or in other words, for any $\frac{1}{2} \le d \le 1$ we have $f(n) \notin \Theta(n^d)$.

Question 3 [2+3+7+2(+1+1)=14(+2)]

We define the Fibonacci sequence $\{t_n\}$ by

$$t_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ t_{n-1} + t_{n-2}, & \text{if } n \ge 2 \end{cases}$$

(a) Show that $t_n \ge \left(\sqrt{2}\right)^n$ for $n \ge 8$.

(b) Find a constant $k < 1$ such that $t_n \le 2^{kn}$ for $n \ge 0$. Justify that the inequality holds for your choice of k.

(c) One way to compute t_n uses matrix exponentiation. We can express the linear system

$$\begin{cases} t_1 = t_1 \\ t_2 = t_0 + t_1 \end{cases}$$

in matrix notation:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \end{bmatrix}.$$

In general,

$$\begin{bmatrix} t_n \\ t_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \cdot \begin{bmatrix} t_0 \\ t_1 \end{bmatrix}.$$

Give an algorithm $\mathcal{A}$ to compute t_n that uses $O(\log n)$ 2×2 **matrix multiplications**.

(d) Argue that 4 additions and 8 multiplications (of integers) suffice to compute the product of two 2×2 matrices with integer entries (hence the runtime of your algorithm $\mathcal{A}$ in (c) is $O(\log n)$).

(e) (Bonus) Another algorithm to compute t_n is

Explain why the $O(\log n)$ algorithm $\mathcal{A}$ is likely to be slower than the $\Omega(n)$ algorithm $\mathcal{B}$ for small values of n when implemented on an actual machine.

(f) (Bonus) Is there an easy way to improve algorithm $\mathcal{B}$?

Algorithm 2: $\mathcal{B}$ (int n)

Input: $n \geq 0$

if $n = 0$ **then** return 0
if $n = 1$ **then** return 1
create array of integers $T[0..n]$
$T[0] \leftarrow 0; T[1] \leftarrow 1$
for $i \leftarrow 2$ *to* n **do**
  $T[i] \leftarrow T[i-1] + T[i-2]$
return $T[n]$

Question 4 [2+6+4=12 marks]

To reduce the height of the heap one could use a d-way heap. This is a tree where each node contains up to d children, all except the bottommost level are completely filled, and the bottommost level is filled from the left. It also satisfies that the key at a parent is no smaller than the keys at all its children.

a) Explain how to store a d-way heap in an array A of size $O(n)$ such that the root is at $A[0]$. Also state how you find parents and children of the node stored at $A[i]$. You need not justify your answer.

b) What is the height of a d-ary heap on n nodes? Give a tight asymptotic bound that depends on d and n. You may assume that n and d are sufficiently big (e.g. $d \geq 3$ and $n \geq 10$). Note that d is not necessarily a constant.

c) Assume that $n \geq 4$ is a perfect square. What is the height of a d-ary heap for $d = \sqrt{n}$? Give an exact bound (i.e., not asymptotic).

Question 5 [9 marks]

Consider a (max-oriented) meldable heap H that holds n integers. Describe an algorithm that is given H and an integer x, and that finds all items in H for which the priority is at least x. (Note that x may or may not be in H.) Your algorithm should have $O(1 + s)$ worst-case run-time, where s is the number of items that were found.