National Taiwan University
Department of Electrical Engineering
Algorithms, Spring 2024

Classes:
Prof. Shao-Hua Sun &
Prof. Hui-Ru Chiang

Homework #1

(on-line submission due: 9:10 AM, 14 March 2024)

TA: Hsin-Tzu Chang sheena9101019@gmail.com and Yun-Yao Tien erictien02@gmail.com

Collaboration policy: You can discuss the problems with other students, but you must write the final answers by yourself. Please specify all of your collaborators (names and student id's) or resources (websites) for each problem. If you solve some problems by yourself, please also specify "no collaborators". Homework without collaborator specification will be penalized.

1. (10 pts) Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first n1 elements of A. Write pseudocode for this algorithm, which is known as **selection sort**. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n-1$ elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in Θ-notation.

2. (10 pts) Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2, \\ 2T\left(\frac{n}{2}\right) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

   is $T(n) = n \log n$.

3. (10 pts) **Correctness of Horner's rule**
   The following code fragment implements Horner's rule for evaluating a polynomial

$$P(x) = \sum_{k=0}^{n} a_k x^k = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots)),$$

   given the coefficients $a_0, a_1, \ldots, a_n$, and a value for x:

```
1   y = 0
2   for i = n downto 0
3       y = a_i + x \cdot y
```

   a. In terms of Θ-notation, what is the running time of this code fragment for Horner's rule?

   b. Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare to Horner's rule?

   c. Consider the following loop invariant:
   At the start of each iteration of the `for` loop of lines 2–3,

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k.$$

   Interpret a summation with no terms as equaling 0. Following the structure of the loop invariant proof presented in this chapter, use this loop invariant to show that, at termination, $y = \sum_{k=0}^{n} a_k x^k$.

   d. Conclude by arguing that the given code fragment correctly evaluates a polynomial characterized by the coefficients $a_0, a_1, \ldots, a_n$.

4. (10 pts) Prove or disprove $f(n) + g(n) = \Theta(\max(f(n), g(n)))$.

5. (10 pts) Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{30}$ of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\ldots$, $g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. Please **explain** your answer.

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

6. (10 pts) Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem. Start at the left end of the array, and progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray of $A[1 \ldots j]$, extend the answer to find a maximum subarray ending at index $j+1$ by using the following observation: a maximum subarray of $A[1 \ldots j+1]$ is either a maximum subarray of $A[1 \ldots j]$ or a subarray $A[i \ldots j+1]$, for some $1 \le i \le j+1$. Determine a maximum subarray of the form $A[i \ldots j+1]$ in constant time based on knowing a maximum subarray ending at index j. Please write your **pseudo code**.

7. (10 pts) What is the largest k such that if you can multiply 3×3 matrices using k multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\log_2 7})$? What would the running time of this algorithm be?

8. (10 pts) Using the master method in Section 4.5, you can show that the solution to the recurrence $T(n) = 4T\left(\frac{n}{3}\right) + n$ is $T(n) = \Theta(n^{\log_3 4})$. Show that a substitution proof with the assumption $T(n) \le cn^{\log_3 4}$ fails. Then show how to subtract off a lower-order term to make a substitution proof work.

9. (10 pts) Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.

10. (10 pts) Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers. If you use the master theorem, please specify the case number.

    a. $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg n}$.

    b. $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$.

    c. $T(n) = T(n-1) + \lg n$.

    d. $T(n) = \sqrt{n}T(\sqrt{n}) + n$.