National Taiwan University
Department of Electrical Engineering
Algorithms, Fall 2016

Handout #2
September 22, 2016
Yao-Wen Chang

Homework #1 (due in-class, October 13, 2016)

1. Work on (a) Exercise 2.1-1 (page 22) and (b) Exercise 2.3-1 (page 37) based on the string (array of 14 characters): "*ALGORITHMNTUEE*". Please mark the two T's as T_1 and T_2, and the two E's as E_1 and E_2 according to their order in the input, and **show their positions during the processing**.

2. Exercise 2.2-2 (page 29).

3. Exercise 2.3-7 (page 39). (Hint: This is a problem of sorting and searching.)

4. Problem 2-3 (page 41).

5. Consider the following algorithm for computing the square root of a number:

```
Square-Root(x)
1 for i = 1, ..., x/2
2     if i^2 = x
3         output i
```

   Does this algorithm run in polynomial time? Justify your answer.

6. Let $f(n)$ and $g(n)$ be asymptotically positive, prove that $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$ by showing both the O and Ω relations.

7. Exercise 3.1-2 (page 52). **(Hint: Notice that the binomial theorem is true only when b is an integer. For this problem, you shall first find the relations O and Ω.)**

8. Problem 3-3(a) (pages 61–62). Please work only on the 20 functions in the 1st, 3rd, 4th, and 6th **columns**. (Hint: You may find the Stirling's approximation of $n!$ in Page 57 of the text useful for part of this problem. Also, recall the technique of taking the logarithm mentioned in class.)

9. Problem 3-4 (b), (d), (g), and (h) (page 62).

10. Exercise 4.1-5 (page 75).

11. Strassen's algorithm relies on the fact that two 2×2 matrices can be multiplied using only 7 multiplications and 18 additions/subtractions. Suppose you found a way to multiply two 2×2 matrices using 3 multiplications and 10 additions/subtractions. Based on this, you can devise an algorithm similar to Strassen's algorithm for multiplying two $n \times n$ matrices. What would the running time of this algorithm be?

12. (a) Exercise 4.3-7 (page 87). (b) Exercise 4.3-9 (page 88).

13. Exercise 4.4-9 (page 93).

14. Exercise 4.5-4 (page 97).

15. Problem 4-1 (d), (e), and (f) (page 107).

16. Problem 4-3 (a), (e), and (f) (page 108).

17. Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A. Inversions are often used for *collaborative filtering* to match your preferences (for books, movies, music, etc.) with other people on the internet. Once the web site has identified people with similar tastes to yours—based on your rating or search of things and the inversion computations—it can recommend you new things that these other people have liked.

    (a) Give the four inversion pairs of the array $< 2, 4, 3, 1 >$.

    (b) Give an efficient, subquadratic (little-oh $o(n^2)$) algorithm that determines the number of inversions in any permutation on n elements by modifying the Merge procedure in MergeSort discussed in class. Instead of giving a detailed pseudo code, it suffices to describe your procedure. See below for the MergeSort pseudo code from class. **(Hint: Think about how you count the number of inversions while merging.)**

    (c) First give the recurrence of the running time for your overall algorithm that counts the number of inversions and then derive the Θ notation for its time complexity.

18. (DIY Problem) For this problem, you are asked to design a problem *set* related to Chapter(s) 1, 2, 3, and/or 4 and give a sample solution to your problem set. Grading on this problem will be based upon the *quality* of the designed problem as well as the *correctness* of your sample solution.