Recitation HW#1

EE4033 Algorithms, Fall 2019
Instructor: Yao-Wen Chang, James Chien-Mo Li, and Hui-Ru Iris Jiang

Presenter: Shang-Chien Lin, Yu-Sheng Lu

HW1

- Deadline: 10/1 (Tue.) 9:00 AM
  - Submit your solution 10 minutes before the class begins

- Collaboration policy
  - Please specify all of your collaborators (name and student id) for each problem. If you solve some problems by yourself, please also specify “no collaborators”. Problems without collaborator specification will not be graded.

1. (a) Exercise 2.1-1 and (b) Exercise 2.3-1

Work on (a) Exercise 2.1-1 and (b) Exercise 2.3-1 based on the string (array of 14 characters): "BESTALGORITHMS". Please mark the two T's as T_1 and T_2, and the two S's as S_1 and S_2 according to their order in the input, and **show their positions during the processing**.

- Input is "BESTALGORITHMS"
- Insertion sort and merge sort
- Show your steps
- Stable or unstable sort

2. Exercise 2.2-2

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 $n-1$ 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.

- Pseudocode, loop invariant, the first *n-1* elements, runtime analysis (best-case and worst-case)

- Make sure your loop invariant fulfills the three necessary properties

3. Exercise 2.3-7

Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

- Please explain why the runtime of your algorithm is *Θ(nlgn)*

4. Problem 2-4 (a), (b), and (d)

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.

a. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

- Please explain why the runtime of your algorithm is *Θ(nlgn)*

5 and 6

3.1-1

Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

3-3 Ordering by asymptotic growth rates

a. 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 $f(n) = \Theta(g(n))$.

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

7. Exercise 4.2-1

Use Strassen's algorithm to compute the matrix product

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}.$$

Show your work.

- Show your work

都要列出來

8. Exercise 4.3-7

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

- Two sub-problems 前面失敗後面成功都要寫

9 and 10

4.4-9

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.

4.5-4

Can the master method be applied to the recurrence $T(n) = 4T(n/2) + n^2 \lg n$? Why or why not? Give an asymptotic upper bound for this recurrence.

- Please answer why or why not
- Give an asymptotic upper bound

11 and 12

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^4$.

c. $T(n) = 16T(n/4) + n^2$.

e. $T(n) = 7T(n/2) + n^2$.

a. $T(n) = 4T(n/3) + n \lg n$.

c. $T(n) = 4T(n/2) + n^2\sqrt{n}$.

e. $T(n) = 2T(n/2) + n/\lg n$.

- Give asymptotic upper and lower bounds
  - Justify your answers
- Make your bounds as tight as possible

13

These songs tend to last a long time, despite having relatively short scripts. In particular, you can convey the words plus instructions for one of these songs by specifying just the new line that is added in each verse without having to write out all the previous lines each time. (So the phrase "five golden rings" only has to be written once, even though it will appear in verses five and onward.)

There's something asymptotic that can be analyzed here. Suppose, for concreteness, that each line has a length that is bounded by a constant c, and suppose that the song, when sung out loud, runs for n words total. Show how to encode such a song using a script that has length $f(n)$, for a function $f(n)$ that grows as slowly as possible.

- Total *n* words
- Suppose the song is composed of *L* different lines
  - What is the relationship between *L* and *n?*
  - What is the relationship between *L* and the script length *f(n)?*

如何快速的print出歌詞？

14. DIY Problem

(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.

- Design a creative problem (set) and **solution** based on specified chapter range
  - Full credit: innovative, novel, no one see it before!
  - Half credit: reasonable, incremental extension from existing problem
  - Quarter/No credit: no novelty, copy & paste from other source
- Makeup story is useless ☹
- Great DIY will be demonstrated after each HW deadline

Chill Time!

- Sorting song
  - https://www.youtube.com/watch?v=kPRA0W1kECg
  - https://www.youtube.com/watch?v=y9Ecb43qw98

What is “Stable”?

- Numerical stability
  - Primarily means stability for solving numerical linear algebra and differential equations
  - https://en.wikipedia.org/wiki/Numerical_stability

- Sorting stability
  - The relative order of records with equal keys remain the same
  - Stable sort: merge sort, insertion sort, bubble sort
  - Unstable sort: quick sort, heap sort

What is “Optimal”?

- For a given problem, a solution is **optimal** means that no solution in the solution space is greater/smaller w.r.t the given objective function
- For a given problem, an algorithm is **exact** means that it can generate an optimal solution
- For a given problem, an algorithm is **optimal** means that it can generate an optimal solution within the lowest time complexity
  - Need to prove the lowest bound, and the algorithm can reach it

What is “Polynomial-Time” Function?

- **Polynomial-time complexity**
  - $O(p(n))$, where n is the **input size** and $p(n)$ is a polynomial function of n **($p(n) = n^{O(1)}$)**

- **Input size**
  - The asymptotic storage space needed in the computer w.r.t the input

- An algorithm is a **polynomial-time** one iff
  1. All variables in the asymptotic bound are **inputs**
  2. All functions are polynomial w.r.t their **input size**