Homework 3

2019.03.14

Note: When the exercise asks you to "design an algorithm for…," it always means that "designs an EFFICIENT algorithm for … and ANALYZES your algorithm". You should keep this in mind when writing solutions.

1. **EXT 3-1**
   在課本 4.1 節中(p.68)作者將 stock buying problem 轉成 maximum-subarray problem 再用 Divide-and-conquer 的概念來解此問題，請給一個 $O(n)$執行時間的 Pseudo code algorithm 直接解 stock buying problem，毋需再做任何 transformation。

2. **Problem 2-4: Inversions**
   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 〈2, 3, 8, 6, 1〉.
   b. What array with elements from the set {1, 2, ⋯, n} has the most inversions? How many does it have?
   c. What is the relationship between the running time of the insertion sort and the number of inversions in the input array? Justify your answer.
   d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\theta(nlgn)$ worst-case time. (Hint: Modify merge sort.)

3. Analyze best-case, average-case, and worst-case performance of the following pseudocode which describes a sorting algorithm and determine whether it is in place and whether it is stable. Append your analyzing process or reasons

```
sort(first, last) # pass by address
   swap first and last if last.value < first.value
   size = last - first + 1
   if size >= 3
       offset = size / 3
       sort(first, last - offset)
       sort(first + offset, last)
       sort(first, last - offset)
```

4. **Problem 6-3: Young tableaus**

   An $m \times n$ ***Young tableau*** is an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries of each column are in sorted order from top to bottom. Some of the entries of a Young tableau may be ∞, which we treat as nonexistent elements. Thus, a Young tableau can be used to hold $r \le mn$ finite numbers.

   a. Draw a 4×4 Young tableau containing the elements {9, 16, 3, 2, 4, 8, 5, 14, 12}.
   b. Argue that an $m \times n$ Young tableau Y is empty if $Y[1, 1] = \infty$. Argue that Y is full (contains mn elements) if $Y[m, n] < \infty$.
   c. Give an algorithm to implement EXTRACT-MIN on a nonempty $m \times n$ Young tableau that runs in $O(m + n)$ time. Your algorithm should use a recursive subroutine that solves an $m \times n$ problem by recursively solving either an $(m - 1) \times n$ or an $m \times (n - 1)$ subproblem. (*Hint:* Think about MAX-HEAPIFY.) Define $T(p)$, where $p = m + n$, to be the maximum running time of EXTRACT-MIN on any $m \times n$ Young tableau. Give and solve a recurrence for $T(p)$ that yields the $O(m + n)$ time bound.
   d. Show how to insert a new element into a nonfull $m \times n$ Young tableau in $O(m + n)$ time.
   e. Using no other sorting method as a subroutine, show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time.
   f. Give an $O(m+n)$-time algorithm to determine whether a given number is stored in a given $m \times n$ Young tableau.

5. **Exercise 4.2-3**

   How would you modify Strassen's algorithm to multiply n×n matrices in which n is not an exact power of 2? Show that the resulting algorithm runs in time $\theta(n^{lg7})$.

6. **Exercise 4.2-4**

   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×n matrices in time $o(n^{lg7})$? What would the running time of this algorithm be?

7. **Exercise 4.2-5**

   V. Pan has discovered a way of multiplying 68×68 matrices using 132,464 multiplications, a way of multiplying 70×70 matrices using 143,640 multiplications, and a way of multiplying 72×72 matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

8. A rod-cutting problem is said that, given a rod and a table of prices. How to cut a steel rod of length ***n*** into pieces in order to maximize the revenue ***r*** ? Suppose the rod lengths are always an integer. Given a rod of length ***10***, how to cut it to get the maximum revenue? You need to show your steps and find the maximum revenue. (Hint: You can use the recursive function in PPT Unit 5.)

length i	1	2	3	4	5	6	7	8	9	10
price p_i	2	7	9	11	13	15	17	17	19	20