Algorithm HW3

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.

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.

Professor Howard have proposed the following "elegant sorting" algorithm:

```
STOOGE SORT(A, i, j)
{
   if A[ i ] > A[ j ]
      then   exchange A[ i ] <-> A[ j ]
if i + 1 >=  j
      then   return
k <- ⌊(j − i + 1) / 3⌋
STOOGE SORT(A, i, j - k)
STOOGE SORT(A, i + k, j)
STOOGE SORT(A, i, j - k)
}
```

a. Determine whether the sorting algorithm is in place and whether it is stable.

b. Argue that, if *n* = length[*A*], then *STOOGE SORT(A, 1, length[A])* correctly sort the input array *A*[1…*n*]

c. Give a recurrence for the worst-case running time of STOOGE SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

d. Compare the worst-case running time of *STOOGE SORT* with insertion sort, merge sort, heapsort, and quicksort. Do the professor deserve tenure?

4.

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 $\theta(n^{lg7})$.

5.

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*?

6.

How quickly can you multiply a *kn×n* matrix by an *n×kn* matrix, using *Strassen's algorithm* as a subroutine? Answer the same question with the order of the input matrices reversed.

7.

Young tableaus

An *m × n* ***Young tableau*** is an *m × 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 ≤ 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 × n* Young tableau *Y* is empty if *Y*[1, 1] = ∞. Argue that *Y* is full (contains *mn* elements) if *Y*[*m*, *n*] < ∞.

c. Give an algorithm to implement **EXTRACT-MIN** on a nonempty *m × n* Young tableau that runs in *O*(*m + n*) time. Your algorithm should use a recursive subroutine that solves an *m × n* problem by recursively solving either an *(m - 1) × n* or an *m × (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 × 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 non-full *m × n* Young tableau in *O*(*m + n*) time.

e. Using no other sorting method as a subroutine, show how to use an *n × 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.