Algorithm Quiz 1

1. (10 %) Please prove that $\lg(n!) = \Theta(n \lg n)$.

Ans：

We take two parts. One for O, and another for Ω.

(a) Upper bound(O):

$$\begin{aligned}\lg n! &= \lg n + \lg(n-1) + \cdots + \lg 1 \\ &\leq \lg n + \lg n + \cdots \lg n \\ &= n \lg n\end{aligned}$$

Therefore, $\lg n! = O(n \lg n)$

(b) Lower bound(Ω):

$$\lg n! = \lg n + \cdots + \lg\left(\frac{n}{2}\right) + \lg\left(\frac{n}{2} - 1\right) + \cdots + \lg 1$$

$$\geq \lg n + \cdots + \lg\left(\frac{n}{2}\right) + 0 + \cdots + 0$$

$$\geq \lg\left(\frac{n}{2}\right) + \cdots + \lg\left(\frac{n}{2}\right)$$

$$= \left(\frac{n}{2}\right)\lg\left(\frac{n}{2}\right) = \left(\frac{n}{2}\right)\lg(n) - \left(\frac{n}{2}\right)$$

Therefore, $\lg n! = \Omega(n \lg n)$

By (a)(b), $T(n) = \theta(n \lg n)$

2. (10 %) Give asymptotic tight bound (Θ) for $T(n)$ where $T(n) = T(n-1) + n^2$. (Assume that $T(n)$ is a constant for sufficiently small n.)

Ans：$\theta(n^3)$

$$T(n) = T(n-1) + n^2$$
$$= T(n-2) + (n-1)^2 + n^2$$
$$= T(0) + 1^2 + 2^2 + \cdots + (n-1)^2 + n^2$$
$$= c + 1^2 + 2^2 + \cdots + (n-1)^2 + n^2$$
$$= c + \frac{1}{6}n(n+1)(2n+1)$$
$$= c + \frac{2n^3+3n^2+n}{6}$$

Therefore, $T(n) = \theta(n^3)$

3. (10 %) Give asymptotic tight bound (Θ) for $T(n)$ where $T(n) = T(\sqrt{n}) + \lg n$. (Assume that $T(n)$ is a constant for sufficiently small n.)

Ans：$\theta(lgn)$

Sol 1:

Let $n = 2^m$

$$T(n) = T(\sqrt{n}) + \lg n$$

⇨ $T(2^m) = T(2^{\frac{m}{2}}) + \lg 2^m$

$$= T\left(2^{\frac{m}{2}}\right) + m$$
$$= T\left(2^{\frac{m}{4}}\right) + \frac{m}{2} + m$$
$$= T\left(2^{\frac{m}{8}}\right) + \frac{m}{4} + \frac{m}{2} + m$$
$$= T(2^1) + 2 + \cdots + \frac{m}{4} + \frac{m}{2} + m$$
$$= c + \frac{2(2^{lgm} - 1)}{2-1}$$
$$= c + 2m - 2$$

Since $n = 2^m, m = lgn$. Therefore,

$$T(2^m) = c + 2m - 2$$

⇨ $T(n) = c + 2lgn - 2 = \theta(lgn)$

Sol 2:

Let $m = lgn$ and $S(m) = T(2^m)$

$T(2^m) = T(2^{\frac{m}{2}}) + \lg 2^m$

⇨ $S(m) = S(\frac{m}{2}) + m$

Using the master theorem, $f(m) = m = \Omega(m^{\frac{1}{2}+\epsilon})$,

$\epsilon = \frac{1}{2} > 0$ and $f\left(\frac{m}{2}\right) \leq cf(m)$, $c = \frac{1}{2} < 1$, case 3 applies.

Therefore, $S(m) = \theta\big(f(m)\big) = \theta(m)$.

⇨ $T(n) = \theta(lgn)$

4. (10 %) Compare the following time complexity and arrange them in increasing order.

(a) $n^{\frac{1}{\lg n}}$

(b) $(\lg n)^{\lg n}$

(c) $\lg n!$

(d) $n!$

(e) $4^{\lg n}$

(f) 2^n

Ans：a<c<e<b<f<d

(a) $n^{\frac{\lg 2}{\lg n}}$ → $n^{\lg_n 2}$ → $2^{\log_n n}$ → 2 → O(1)

(b) $n^{\lg\lg n}$

(c) $\text{nlg}\, n$

(d) n!

(e) $n^{\lg_2 4}$ → n^2

(f) 2^n

5. (10 %) Radix sort: Given 1243, 854, 618, 56, 75, 81, 994, 324, please add the first three element after finishing the second digit arrange. (*Hint*: zero padding to a certain length.)

Ans：2185

First round

81, 1243, 854, 994, 324, 75, 56, 618

Second round

618, 324, 1243, 854, 56, 75, 81, 994

618+324+1243=2185

6. (10 %) Please fill the blanks in MAX-HEAPIFY Pseudocode below.

```
MAXHEAPIFY(A, i, n)
1  l ← LEFT(i)                          // l 代表以 i 為 root 的左子樹
2  r ← RIGHT(i)                         // r 代表以 i 為 root 的右子樹
3  if l ≤ n and A[l] > A[i] then        // l ≤ n 確認左子樹存在
4      largest ← l                      // 如果左子樹的值較大，就將 largest 設成左子樹的 [位置]
5  else
6      largest ← i                      // 如果 A[i]的值較大，就將 largest 設成 root 的 [位置]
7  if      (a)      then                // r ≤ n 確認右子樹存在
8      largest ← r                      // 如果右子樹的值較大，就將 largest 設成右子樹的 [位置]
9  if largest ≠ i then                  // 如果 largest 不是 root 的位置，就把左右子樹中比較
10     (b)                              //   大的數值跟 root 的數值做交換
11     MAXHEAPIFY(A, largest, n)
```

Ans：

(a) $r \leq n \; and \; A[r] > A[largest]$
(b) $exchange \; A[i] \leftrightarrow A[largest]$

7. (10 %) What are the minimum and maximum numbers of elements in a heap of height h?

Ans：min：2^h; max：$2^{h+1} - 1$

A heap with height h is an almost-complete binary tree (complete at all levels except possibly the lowest)

- Maximum: at most $2^{h+1} - 1$ elements (if it is complete)
- minimum: at least $2^h - 1 + 1 = 2^h$ elements (if the lowest level has just 1 element and the other levels are complete).

8. (10 %) When does the worst case of Quicksort occur?

(a) The input array A contains distinct elements and is sorted in increasing order.

(b) The input array A contains distinct elements and is sorted in decreasing order.

(c) All elements of the input array are the same.

(d) The PARTITION always produces a 9-to-1 split.

(e) None of above.

Ans：abc

9. (10 %) Show that any comparison sort algorithm requires $\Omega(n \lg n)$ comparisons in the worst case.

Ans：

Proof

- $l \geq n!$ because each of the $n!$ permutations of the input appears as some leaf.
- By lemma, $n! \leq l \leq 2^h$ or $2^h \geq n!$
- Take logs: $h \geq \lg(n!)$
- Use Stirling's approximation: $n! > (n/e)^n$ (by equation (3.16))

$h \geq \lg(n/e)^n$ (e = Euler's number ≈ 2.71828)

$= n \lg(n/e)$

$= n \lg n - n \lg e$

$= \Omega(n \lg n)$

■

10. (10 %) Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? Give a simple scheme that makes any comparison sort stable.

Ans：

Stable: insertion sort, merge sort

Use an extra array to store the indices of the array elements.

After any sorting algorithm is completed, sort the same

numbers in the input array according to the index array.