2015 Algorithm Midterm Solutions

指導教授 ： 謝孫源 教授

助教： 葉承翰、詹博丞、林帝丞

Question 1(10pts)

Solution:

$$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ s.t. } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \}$$

$$\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ s.t. } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \}$$

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ s.t. } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}$$

Question 1

Solution:

Yes $2^{n+1} = O(2^n)$ since $2^{n+1} = 2 \times 2^n \leq 2 \times 2^n$!
NO, by definition we have $2^{2n} = (2n)^2$ which for no constant c asymptotically may be less than or equal to $C \times 2^n$

配分:
各兩分
只寫 yes, no (1 分)

Question 2(10pts)

Solution:

- Suppose $n = 2^m$

$$T(2^m) = 2^{m/2} T(2^{m/2}) + 2^m$$

$$S(m) = 2^{m/2} S(m/2) + 2^m = 2^{m/2}(2^{m/4} S(m/4) + 2^{m/2}) + 2^m$$

$$= 2^{\frac{3}{4}m} S(m/4) + 2 \times 2^m = \cdots = 2^{\frac{m(m-1)}{m}} S(1) + 2^m \lg m$$

$$= \Theta(2 \lg m)$$

- So $T(n) = \Theta(n \lg\lg n)$

配分：

只寫答案 (3 分) , 依過程斟酌給分

Question 3(10pts)

Solution:

- We use RADIX-SORT , this makes RADIX-SORT to be

$$\Theta(d(n+k)) = \Theta(2(n+n)) = \Theta(4n) = \Theta(n)$$

suppose $k = n$ and $d = \log_k(n^2)$

RADIX-SORT(*A*, *d*)

for *i* ← 1 **to** *d*

do use a stable sort to sort array *A* on digit *I*

配分：

寫出利用$\Theta(n)$的演算法 (5 分)

分析時間複雜度 (5 分)

Question 4(10pts)

Solution:

Proof From the preceding discussion, it suffices to determine the height of a decision tree in which each permutation appears as a reachable leaf. Consider a decision tree of height h with l reachable leaves corresponding to a comparison sort on n elements. Because each of the $n!$ permutations of the input appears as some leaf, we have $n! \le l$. Since a binary tree of height h has no more than 2^h leaves, we have

By lemma, $n! \leqq l \leqq 2^h$ or $2^h \geqq n!$

Take logs: $h \geqq \lg(n!)$

Use Stirling's approximation: $n! > (n/e)^n$ (by equation (3.16))

$$
\begin{aligned}
h &\geqq \lg(n/e)^n \\
&= n \lg(n/e) \\
&= n \lg n - n \lg e \\
&= \Omega(n \lg n)
\end{aligned}
$$

(theorem) ■

配分：

依過程斟酌給分

Question 5(10pts)

Solution: Modify the PARTITION procedure of QUICKSORT to use the SELECT algorithm to choose the median of the input array as the pivot element. The worst-case running time of SELECT is linear, so we do not increase the time requirement of PARTITION, and selecting the median as pivot guarantees the input is split into two equal parts, so that we always have the best-case partitioning. The running time of the entire computation is then given by the recurrence $T(n) = 2T(n/2) + O(n) = O(n \lg n)$.

配分：

依過程與敘述給分

Question 6(10pts)

解答：

- The number of internal nodes a complete binary tree has is −1 where h is the height of the tree. A heap of height h has at least one additional node (otherwise it would be a heap of length h−1) and at most additional nodes (otherwise it would be a heap of length h+1), which is similar to the answer in exercise 6.1-1.

The number of internal nodes a complete binary tree has is $2^h - 1$ where h is the height of the tree. A heap of height h has at least one additional node (otherwise it would be a heap of length h−1) and at most 2^h additional nodes (otherwise it would be a heap of length h+1), which is similar to the answer in exercise 6.1-1.

- Thus, if n∈(,), then the height will be ⌊lgn⌋.
- Thus, if n∈$(2^h, 2^{h+1} - 1)$, then the height will be ⌊lgn⌋.

配分： 完整答案 10 分，斟酌扣分

Question 7(10pts)

解答：

If f(n)∈Θ(g(n)), then there are constants c1,c2>0 such that for large enough n, we have $c_1 g(n) \le f(n) \le c_2 g(n)$

But this implies $g(n) \le \frac{1}{c_1} f(n)$ as well as $\frac{1}{c_2} f(n) \le g(n)$ for large enough n.

$$\frac{1}{c_2} f(n) \le g(n) \le \frac{1}{c_1} f(n)$$

Therefore , g(n)∈Θ(f(n))

配分： 完整答案 10 分，斟酌扣分

Question 8(10pts)

解答：

(1). 不能用Master method -----------------------------(5分)

(2). 解釋 --(5分)

OR 寫出答案 ---------------------------------------(5分)

如果第一題寫**可以**的話，後面一定要寫出

Extension of Master method

如果寫不行

解釋:

因為跟 f(n) 差了 log n 倍

所以不符合 master method

Question 8(10pts)

解答：

In the Master Theorem, as given in the textbook and previous handout, there is a gap between cases (1) and (2), and a gap between cases (2) and (3).

For example, if $a = b = 2$ and $f(n) = n/\lg(n)$ or $f(n) = n\lg(n)$, none of the cases apply. The extension below partially fills these gaps.

THEOREM (*Extension of Master Theorem*) If a, b, $E \stackrel{\text{def}}{=} \log_b(a)$, and $f(n)$ are as in the Master Theorem, the recurrence

$$T(n) = aT(n/b) + f(n), \;\; T(1) = d,$$

has solution as follows:

1') If $f(n) = O(n^E(\log_b n)^\alpha)$ with $\alpha < -1$, then $T(n) = \Theta(n^E)$.

2') If $f(n) = \Theta(n^E(\log_b n)^{-1})$, then $\mathrm{T}(n) = \Theta(n^E \log_b \log_b(n))$.

3') If $f(n) = \Theta(n^E(\log_b n)^\alpha)$ with $\alpha > -1$, then $\mathrm{T}(n) = \Theta(n^E(\log_b n)^{\alpha+1})$.

4') [*same as in Master Theorem*] If $f(n) = \Omega(n^{E+\varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(f(n))$, provided there is a constant c with $c < 1$ such that

$$af(n/b) \le cf(n) \;\; \text{for all } n \text{ sufficiently large.}$$

Question 8(10pts)

解答：

Sol:

$$T(n) = 27T\left(\frac{n}{3}\right) + \Theta\left(\frac{n^3}{\lg n}\right)$$

$$= 27\left(27T\left(\frac{n}{3^2}\right) + \Theta\left(\frac{\frac{n^3}{3}}{\lg\frac{n}{3}}\right)\right) + \Theta\left(\frac{n^3}{\lg n}\right)$$

$$= 27^2T\left(\frac{n}{3^2}\right) + C_1\left(\frac{n^3}{\log_3 n - 1}\right) + C_0\left(\frac{n^3}{\log_3 n\, n}\right)$$

$$= \cdots$$

$$= 27^{\log_3 n}T(1) + C_{\log_3(n-1)}\left(\frac{n^3}{\lg_3 n - \lg_3 n - 1}\right) + \cdots + C_1\left(\frac{n^3}{\log_3 n - 1}\right) + C_0\left(\frac{n^3}{\log_3 n}\right)$$

$$= n^3 + C\,n^3(\lg\lg n)$$

$$= \Theta(n^3 \lg\lg n)$$

Question 9(10pts)

解答：

QuickSort	Best Case	Worst Case	Average case
	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n \log n)$

Quicksort

```
Quicksort(A,p,r) {
    if (p < r) {
        q <- Partition(A,p,r)
        Quicksort(A,p,q)
        Quicksort(A,q+1,r)
    }
}
```

```
Partition(A,p,r)
    x <- A[p]
    i <- p-1
    j <- r+1
    while (True) {
        repeat
            j <- j-1
        until (A[j] <= x)
        repeat
            i <- i+1
        until (A[i] >= x)
        if (i A[j]
        else
            return(j)
    }
}
```

配分： 時間複雜度各 2 分， Quicksort 2 分， partition 兩分

Question 10(10pts)

解答：

MergeSort	Best Case	Worst Case	Average case
Heapsort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$

```
Heapsort(A) {
   BuildHeap(A)
   for i <- length(A) downto 2 {
      exchange A[1] <-> A[i]
      heapsize <- heapsize -1
      Heapify(A, 1)
}
```

```
BuildHeap(A) {
   heapsize <- length(A)
   for i <- floor( length/2 ) downto 1
      Heapify(A, i)
}
```

```
Heapify(A, i) {
   le <- left(i)
   ri <- right(i)
   if (le<=heapsize) and (A[le]>A[i])
      largest <- le
   else
      largest <- i
   if (ri<=heapsize) and (A[ri]>A[largest])
      largest <- ri
   if (largest != i) {
      exchange A[i] <-> A[largest]
      Heapify(A, largest)
   }
}
```

配分： 時間複雜度各 2 分，heap sort 2 分

build max heap2 分 heapify 4 分