2014 Algorithm Midterm Solution

指導教授：謝孫源 教授

課程助教：陳紀廷 林箴諺 鐘煒康 梁文宣 黃信傑

實驗室：互聯網暨高效率演算法 65803 (新系館8F)

Question 1 - 1

- 解答:

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

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

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

Question 1 - 2

- 配分(10%)
  - $\theta(g(n))$ (4分)
  - $O(g(n))$ (3分)
  - $\Omega(g(n))$ (3分)
- 扣分方式
  - 每錯一部分 扣2分

Question 2

- 解答:
- Part(1) :
  - $a = 4, b = 2, f(n) = n^2 \log n\,, n^{\log_b a} = n^2$
  - The ratio of $f(n)/_{n^{\log_b a}} = \log n$. This implies $f(n)$ is asymptotically larger than $n^{\log_b a}$, but not polynomial larger .
  - Thus, the master method can not be applied to solve this recurrence.
- Part(2) :
  - Using the recursion tree method, the effort at each level is $\leq n^2 \log n$, and the effort for the leaf-level is n^2.
  - The total effort is $\leq (n^2 \log n)(\log n - 1) + n^2$.
  - This implies $n^2 log^2 n$ is an upper bound for this recurrence.

Question 2

- **配分方式(10%)**
  - Can or Can’t (2%)
  - Why or Why not (4%)
  - Upper bound (4%)

Question 3

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.

- 解答:利用mergesort和binary search方法解

  1.先將S集合以mergesort排序

  2.把排序好的S最後面的element取出(即最大element)並設為y

  3.若S非空，使用binary search 找尋 z= x-y

  4.如果在S中可以找到這樣的z ，即代表找到z和y使得其相加為x，則STOP，否則，重複STEP2~4直到找到

  5.若一直找直到S為空，則無兩數相加等於x，回傳NIL值

- Pseudo code為:

```
DETERMINE-x(S, x)
1   MERGE-SORT(S, 1, length(S))
2   y = REMOVE-MAX(S)
3   while (S)
4       do z = ITERATIVE-BINARY-SEARCH(S, x-y, 1, length(S))
5          if z + y = x
6             then return TURE
7             else y = REMOVE-MAX(S)
8   return NIL
```

- 第一行執行時間為: θ(nlgn)
- 第二行執行時間為: O(1)
- 第四行執行時間最多 lgn time
- 第三~七行最多執行n次，所以總執行時間為: θ(nlgn)

Question 3

- 配分(10%)
- 扣分方式
  - 本題無一定的標準答案，只要可以寫出能找到此兩數又可讓時間複雜度控制在 $\Theta(n \lg n)$即給分
  - 只寫出sort沒寫出如何找兩數給5分
  - 未分析時間複雜度扣2分

Question 4

Show that there are at most ⌈$n/2^{h+1}$⌉ nodes of height h in any n-element heap.

- 解答:

特別注意:此處的height為從leaf開始往上算，而depth才是從root開始算，所以leaf的height為0

證明: By induction on number of nodes at height h.

Base case : Number of leaves at height h=0=number of leaves in the n-element heap. Parent of the last node is the ⌊n/2⌋-node. This will be the last parent in the heap. All the nodes after this one will be leaves. Therefore the number of leaves is ⌈n/2⌉ . This is also true for the formula, therefore base case is true.

We assume that number of nodes at height h-1 is given by the formula, ⌈$n/2^h$⌉.

Prove height =h

Note that if we remove all the leaves from the heap, the nodes that were earlier at height 1 now become leaves in the new heap, i.e. they have height 0. Similarity, all the nodes in the new tree will have height that is one less than their old height. We shall use this to prove the induction.

Question 4

- 解答:

  Induction: Consider a new tree that is obtained by removing the leaves. This tree is a heap with n- [n/2] nodes in it. The number of nodes at height h in the old heap will be the same at the number of nodes at height h-1 in the new heap, which is $(n-[n/2])/2^h = [n/2^{h+1}]$

- 配分(10%)
- 扣分方式
  - 只畫圖舉例未實際證明只給2分

Question 5

- 解答

The running time of HEAPSORT on n array of length that is already sorted in increasing order is Θ(nlogn), because even though it is already sorted, it will be transformed back into a heap and sorted.

Decreasing order will be Θ(nlogn). This occurs because even though the heap will be built in linear time, every time the element is removed and HEAPIFY is called, it could cover the full height of the tree.

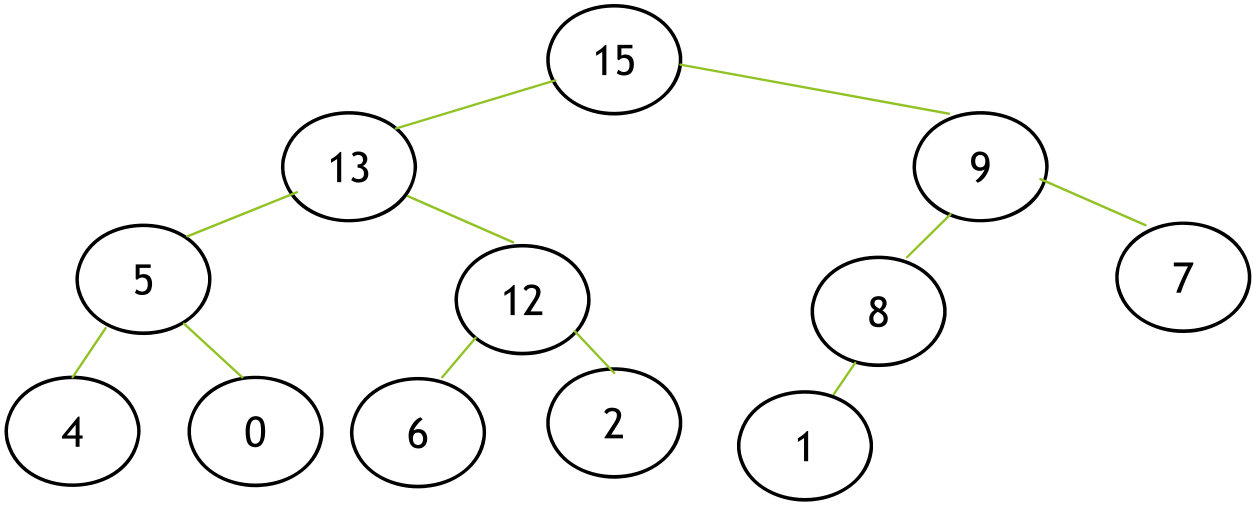
- 配分(10%)

Increasing(5%)-----複雜度(2%)說明(3%)

Decreasing(5%)-----同上

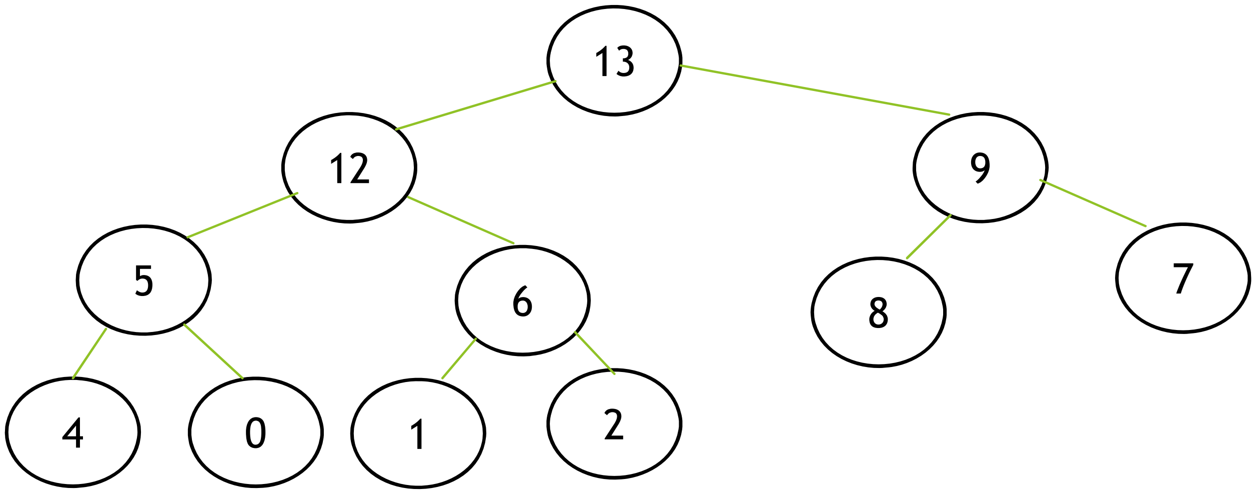
Q6(Mid)

- Illustrate the operation of HEAP-EXTRACT-MAX on the heap A = <15,13,9,5,12,8,7,4,0,6,2,1>
- Step1建樹



- 取root值為15

- step2



Question 7

- 解答

  全部數字依其數值大小，放到相符位置，接著由小到大讀取各個位置的數字。

```
counting_sort(int array[], int N) {  //這邊設定數字小於10000
        int count[10000] = {0};
        for (int i=0; i<N; ++i);
                count[ array[i] ]++;
        for (int i=0, k=0; k<N; ++i)
        while (count[i]--)
                array[k++] = i;
}
```

- 配分(10%)

  algorithm(7分)   Time Complexity(3分)

Q8(Mid)

- Present the merge sort algo and analyze the complexity

```
MERGE(A, p, q, r)
n1 = q - p + 1
n2 = r - q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

- $T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$

$\Theta(n \lg n)$

Question 9

- **解答:**
  - **Input:** range 0~k ,n個integers存入A陣列中
  - **Output:** B[k]={0, 0, 0,…., 0};

```
for( i = 0;  i< n; i++)
     B[A[i]]=B[A[i]]+1;
for( i = 0;  i< n; i++)
     B[i]= B[i]+B[i-1];
```

假使要詢問[a..b]有多少整數只需要return B[b]-B[a-1]

- **評分方式(10%)**
  - **描述演算法過程　　(6%)**
  - **證明時間複雜度　　(4%)**

Question 10 - 1

- 解答:

  Worst case number of comparisons performed corresponds to maximal height of decision tree ; Therefore lower bound on height ⇒ lower bound on sorting.

  Lemma-

  Any binary tree of height h has ≦ 2^h leaves.

  In other words:

  l = # of leaves,

  h = height,

  Then $l \leqq 2^h$.

- *Proof*:

  - Assume elements are the (distinct) numbers 1 through n
  - There must be n! leaves (one for each of the n! permutations of n elements)
  - Tree of height h has at most 2 leaves

Question 10 - 2

- *proof*:

(1) Using Stirling's approximation: $n! = \sqrt{2\pi n}(n/e)^n(1+\theta(1/n))$

$l \geqq n!$

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))

$h \geqq \lg(n/e)^n$

$= n \lg(n/e)$

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

$= \Omega(n \lg n)$

Question 10 - 3

- *proof*:

(2)

$$
\begin{aligned}
2^h \geq n! \Rightarrow h &\geq \log(n!) \\
&= \log(n(n-1)(n-2)\cdots(2)) \\
&= \log n + \log(n-1) + \log(n-2) + \cdots + \log 2 \\
&= \sum_{i=2}^{n} \log i \\
&= \sum_{i=2}^{n/2-1} \log i + \sum_{i=n/2}^{n} \log i \\
&\geq 0 + \sum_{i=n/2}^{n} \log \frac{n}{2} \\
&= \frac{n}{2} \cdot \log \frac{n}{2} \\
&= \Omega(n \log n)
\end{aligned}
$$

Question 10 - 4

- 配分(10%)
  - 說明Worst case number of comparisons performed corresponds to maximal height of decision tree. (2分)
  - 證明 (8分)
- 扣分方式
  - 證明每錯一部分 扣2分