x 41

# Algorithm Middle Exam.

107.11.16

/* 答案紙 請寫上：學號 姓名 ；試卷共 9 頁 總分 100

1. (6%) What is the definition of Algorithm? (三個條件)

有限的步驟，有效率的計算，有終止點

2. (4%) Design a Turing to verify "Can 30 be divided by 8?"(執行過程要寫出來！(2%))

100

3. (6%) The following statements are the merge-sort,

Merge-sort (A, p, r)

1. q <- $\lfloor (p+r)/2 \rfloor$  O(1)
2. Merge-sort (A, p, q)  O(1)
3. Merge-sort (A, q+1, r)  $T(\frac{n}{2})$
4. Merge (A, p, q, r) /* 將兩組排列好的序列合併成一排列好的序列

O(n)

(1) 改正程式錯誤

(2) Use the following sequence of 8 data: 3, 1, 7, 5, 6, 2, 4, 8 to illustrate each step of merge-sort. (正確的演算法)

(3) And analyze the time complex of the algorithm.

✓ 4. (6%) 解 T(n) = 2 T(n/2) + n lg n  $n=2^k$, $k=\lg n$.

5. (8%) 我們可以用 Divide-and-Conquer 方法，將兩個 n x n 矩陣相乘，分成 4 個 n/2 x n/2 小矩陣相乘後合併，如下：

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix},$$

so that we rewrite the equation $C = A \cdot B$ as

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

Equation (4.10) corresponds to the four equations

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21}, \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22}, \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21}, \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22}. \end{aligned}$$

將會有 8 個小矩陣相乘與 4 個 $n^2/4$ 加法。若採 Strassen's method 則拆成 7 個小矩陣相乘與 18 個 $n^2/4$ 加法，請問，優勢在哪？(列出分析過程才有分數)

6. (12%) (1) (4%)何謂 max-heap？此種資料結構有何優勢？可應用在哪裡？

(2) (4%) 在 BUILD-MAX-HEAP(A) 副程式中，為何第 2 行程是要為 for i <- ⌊length[A]/2⌋ downto 1？ 改為 for i <- 1 to ⌊length[A]/2⌋ 可以嗎？ 此副程式時間複雜度為多少？ 推導過程

O(lgn), O(n)

(3) (4%) Illustrate each step of the heap sort by using the following example A[6] = {3, 2, 9, 6, 4, 8}. 說明：1. 初始樹；2. 建好 heap tree ; 3. Heap sort <u>前兩筆資料</u>排序步驟

```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i ← n downto 2
    do exchange A[1] ↔ A[i]
        MAX-HEAPIFY(A, 1, i − 1)
```

```
BUILD-MAX-HEAP(A)
1  heap-size[A] ← length[A]
2  for i ← ⌊length[A]/2⌋ downto 1
3      do MAX-HEAPIFY(A, i)
```

O(1)

O(n)

```
MAX-HEAPIFY(A, i)
l ← LEFT(i)
r ← RIGHT(i)
if l ≤ heap-size[A] and A[l] > A[i]
   then largest ← l
   else largest ← i
if r ≤ heap-size[A] and A[r] > A[largest]
   then largest ← r
if largest ≠ i
   then exchange A[i] ↔ A[largest]
        MAX-HEAPIFY(A, largest)
```

$T(n) = T(\frac{n}{2}) + cn$

$T(\frac{n}{2})$ $= \left(T(\frac{n}{2^2}) + \frac{cn}{2}\right) + cn$

$= T(\frac{n}{2^2}) + \frac{1}{2}cn + cn$

$= \left(T(\frac{n}{2^3}) + \frac{cn}{2^2}\right) + \frac{1}{2}cn + cn$

$= T(\frac{n}{2^3}) + \frac{1}{2^2}cn + \frac{1}{2}cn + \frac{1}{2^0}cn$

$= T(\frac{n}{2^k}) + \left[\sum_{i=0}^{k-1} (\frac{1}{2^i})\right] cn$

$= T(\frac{n}{2^k}) + \left[\frac{1}{\frac{1}{2}}\right] cn$

$n = 2^k$

$k = \lg n$

$= T(1) + cn = O(n)$

k=1 k=2 k=3

$\frac{1}{2^0}$ $\frac{1}{2^0}+\frac{1}{2^1}$ $\frac{1}{2^0}+\frac{1}{2^1}+\frac{1}{2^2}$

7. (4%) 試寫一個產生亂數(隨機)序列演算法之虛擬碼 ?

```
RANDOMIZE-IN-PLACE(A)
1  n ← length[A]
2  i ← 1 to n
3     do swap A[i] ↔ A[RANDOM(i, n)]
```

8. (8%) The following algorithm is counting sort. If array A is following:

```
For i ← 0 to k
    Do C[i] = 0
For j ← 1 to n
    Do C[A[j]] ← C[A[j]] + 1
For i ← 1 to k
    Do C[i] = C[i] + C[i - 1]
For j ← n downto 1
    Do B[C[A[j]]] ← A[j]
       C[A[j]] ← C[A[j]] -1
```

若 A[] = {3, 2, 1, 5, 3, 2, 2, 1} B

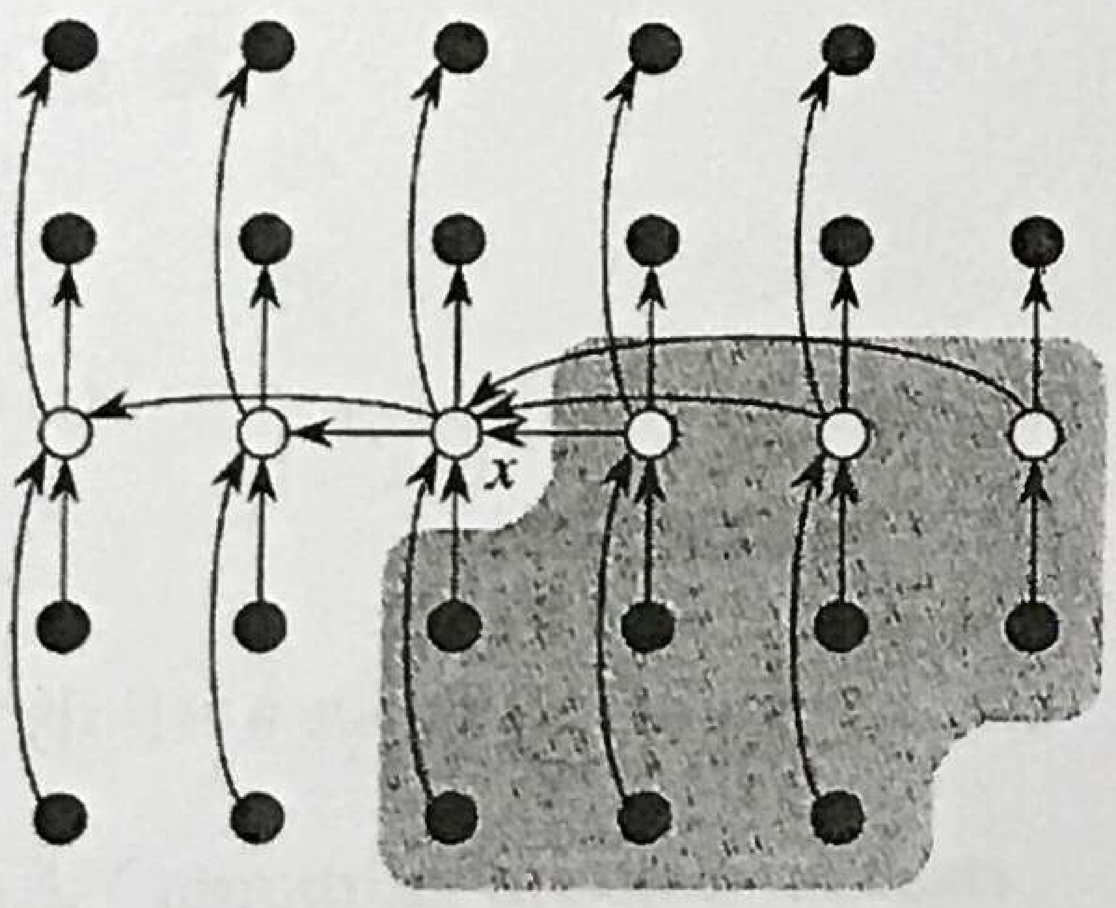
(1) (4%) Please write the initial state and each state of array A & B & C in the step of algorithm.

(2) (4%) 為何此種方法又稱為 stable sorting？是因為哪一行程式造成此現象？

9. (6%) 下列為一個 O(n)的 selection algorithm

**SELECT algorithm**:

1. Divide the ***n*** elements of the input array into $\lfloor n/5 \rfloor$ groups of **5** elements each and at most one group made up of the remaining ***n*** **mod 5** elements. O(n)
2. Find the **median** of each of the $\lceil n/5 \rceil$ **groups** by the first insertion sorting the elements of each group and then picking the median from the sorted list of group elements. O(n)
3. Use the **SELECT** recursively to find the median **x** of the $\lceil n/5 \rceil$ medians found in step 2. $T(\lceil \frac{n}{5} \rceil)$
4. Partition the input array around the median-of-medians **x** using the modified version of **PARTITION**. (以 **x** 來分(**x** 為第 **k** 小)，前段 **k-1** 個，後半段 **n-k** 個) O(n)
5. If $i = k$, then return $x$. Otherwise, use SELECT recursively to find the $i$th smallest on the low side if $i < k$, or $(i - k)$th smallest elements on the high side if $i > k$. $T(\frac{7}{10}n+6)$




(1) (2%) 請說明步驟 4 所選出 x，其大小排序位置範圍為？(以 n 筆資料中， x 介在：R1 ≦ x ≦ R2， R1, R2 =?)

(2) (4%) 設此方法時間為 T(n)，分析此方法，單獨每一步驟之時間複雜度？ 與總時間複雜度？

10. (20%) The matrix-chain multiplication problem is: given a chain <A1, A2, …, An> of $n$ matrices, where for $i$ =1, 2, …, $n$, fully parenthesize the product

A1, A2, …, An in a way that minimizes the number of scalar multiplications.

Use DP to solve this problem is as follows.

**Step 1:** The structure of an optimal parenthesization(括號)

Suppose that an optimal parenthesization of $\mathbf{A_i\ A_{i+1}\ \ldots\ A_j}$ splits the product between $\mathbf{A_k}$ and $\mathbf{A_{k+1}}$. Thus, we can build an optimal solution to an instance of the matrix-chain multiplication problem by splitting the problem into two subproblems, finding optimal solutions to subproblem instances, and then combineing these optimal subproblem solutions.

**Step 2: A recursive solution**

We define the cost of an optimal solution recursively in terms of the optimal solutions to subproblems. Let m[i, j] be the number of scalar multiplications needed to compute the matrix $A_{i..j}$; for the full problem, the cost of a cheapest way $A_{1..n}$ is m[1, n].

A dimension of matrix $A_i$ is $P_{i-1} \times P_i$ ; the matrix product $A_{i..k}\ A_{k+1..j}$ is $P_{i-1} \times P_k \times P_j$ .

Then,

$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j\ .$$

一 般 式：

$$m[i, j] = \begin{cases} 0 & \text{if } i = j\ , \\ \min\limits_{i \le k < j} \{m[i, k] + m[k+1, j] + p_{i-1} p_k p_j\} & \text{if } i < j\ . \end{cases}$$

Let s[i,j] = ***k*** such that $m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$ .

**Step 3. Computing the optimal costs**

We can use a tabular, bottom up approach, compute the optimal cost. For a matrix Ai has **dimension Pi-1 x Pi**. The input sequence <P0, P1, …, Pn>, where **length[P] = n+1**.

We use a **table m[1..n, 1..n]** for storing **m[i, j]** and a table **s[1..n, 1..n]** for recording the index ***k*** achieve the optimal cost in computing m[i, j].

The procedure could be:

$$
m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 & = 13000\,, \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 & = 7125\,, \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 & = 11375 \end{cases}
$$
$$= 7125\,.$$

```
PRINT-OPTIMAL-PARENS(s, i, j)
if i = j
    then print "A"_i
    else print "("
         PRINT-OPTIMAL-PARENS(s, i, s[i, j])
         PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
         print ")"
```

(1) (8 %) 此問題所有括號位置有幾種可能？(用 order 表示)。請說明 Step 3 演算法中，line 4, 5, 6, 8 之時間複雜度？

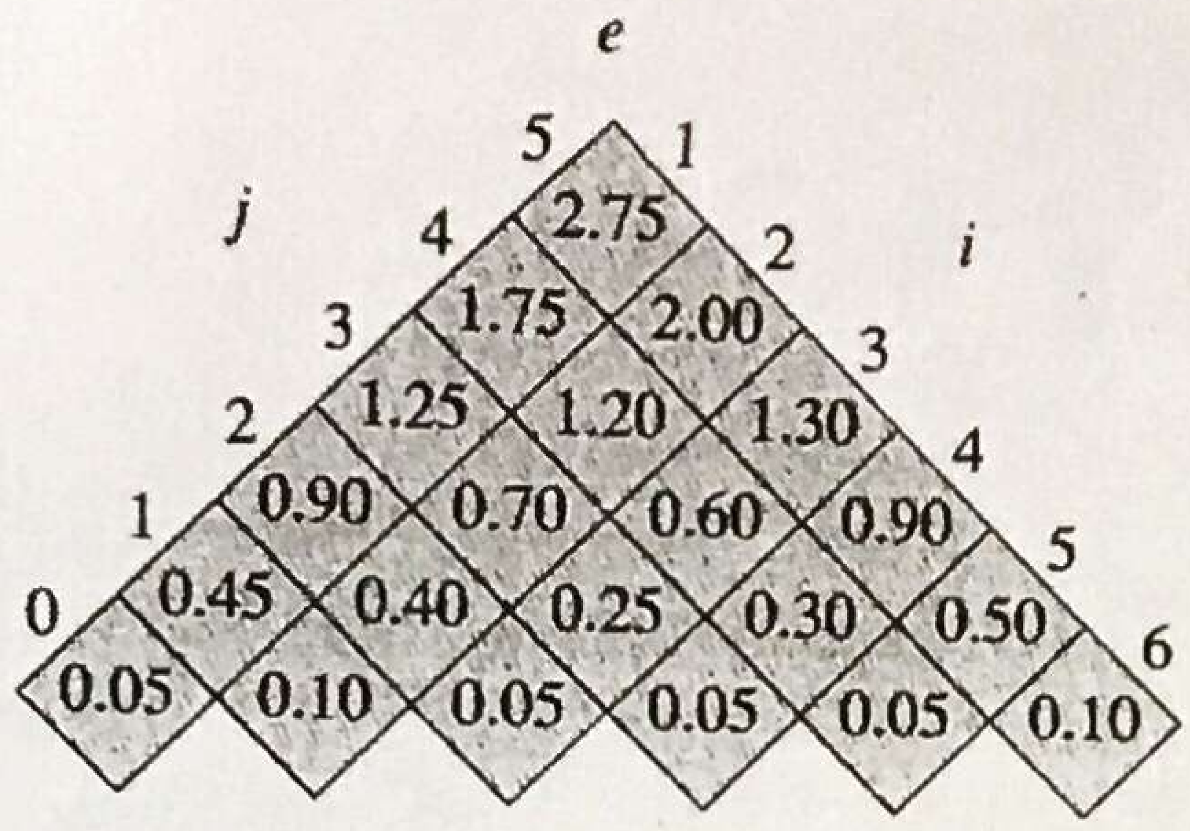
(2) (12%) 若 A1(10, 5), A2(5, 10), A3(10, 10), A4(10, 5), A5(5, 10)

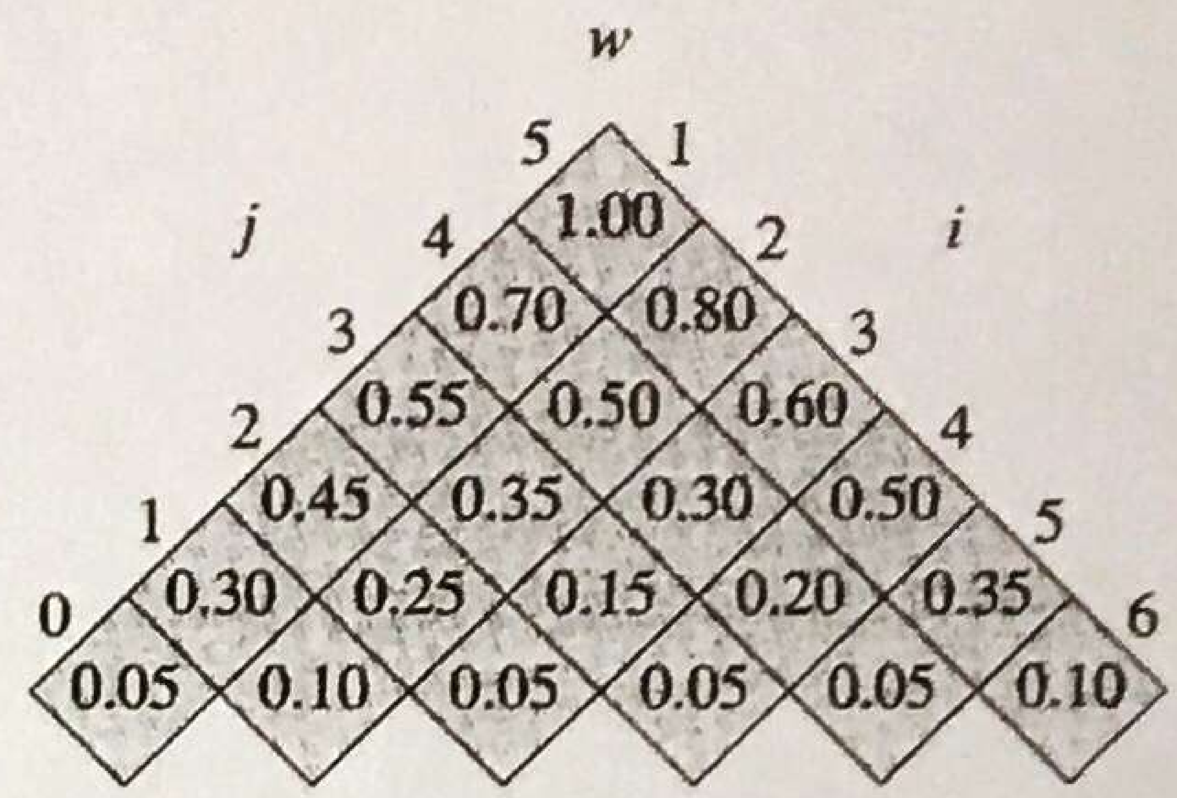
求 Matrix m(i,j) & s(i,j) & 最佳解 (如何括號乘？最少乘法次數？)為何 ？

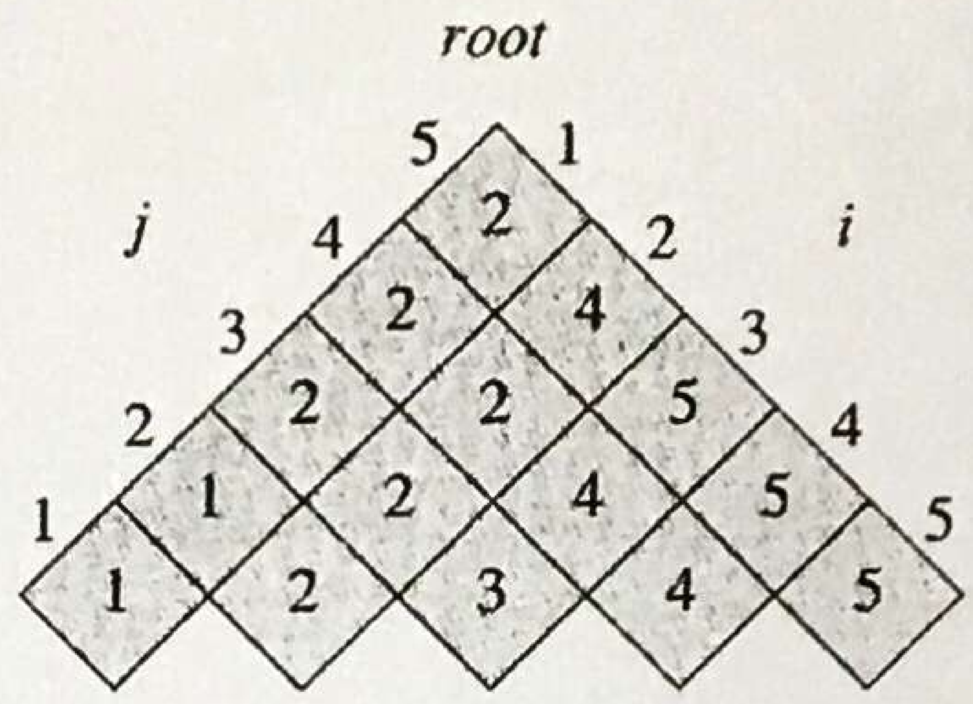
11. (20%) **We want to construct a binary search tree whose expected search cost is smallest. We call such a tree an *optimal binary search tree*.**

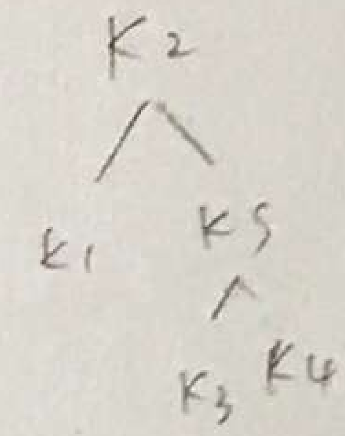
```
OPTIMAL-BST(p, q, n)
for i ← 1 to n + 1
    do e[i, i − 1] ← q_{i−1}
       w[i, i − 1] ← q_{i−1}
for l ← 1 to n
    do for i ← 1 to n − l + 1
           do j ← i + l − 1
              e[i, j] ← ∞
              w[i, j] ← w[i, j − 1] + p_j + q_j
              for r ← i to j
                  do t ← e[i, r − 1] + e[r + 1, j] + w[i, j]
                     if t < e[i, j]
                        then e[i, j] ← t
                             root[i, j] ← r
return e and root
```

注意虛擬碼，老師有可能改掉，
以考試的虛擬碼為主當答案




Print-optimal-BST(root, i, j)

1. if i ≧ j then print “root”, root(i,j)
2. print “left root”, Print-optimal-BST (i, root(i,j))
3. print “right root”, Print-optimal-BST (root(i,j)+1, j)
4. Else return.

現有 5 筆資料， 每筆資料被讀取機率為 pi，落在資料間隔機率為 qi，如下表：

| i | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| pi | | 0.1 | 0.15 | 0.15 | 0.1 | 0.05 |
| qi | 0.1 | 0.05 | 0.05 | 0.1 | 0.1 | 0.05 |

求 e(i,j), w(i,j), root(i,j)與 optimal binary search tree 為何 ？

國立臺南大學　學年度　學期　考試　答案卷

考試科目：演算法　　班別：數位三

學號：S10533020　　姓名：鄭家欣

總分 77

1、有限的步驟, 有效率的計算, 有終止點

2、

2|30 0　2|8 0
2|15 1　2|4 0
2|7 1　2|2 0
2|3 1　1 1
1 1

$Q=\{q_0, q_1, q_2, q_3, q_Y, q_N\}$.

$\Gamma=\{0,1,b\}$

$I \subseteq \Gamma$

b: blank, in $\Gamma - I$

$q_0$: initial state

$q_Y$: Yes state

$q_N$: No state

| | 0 | 1 | b |
|---|---|---|---|
| $q_0$ | $(q_0,0,R)$ | $(q_0,1,R)$ | $(q_1,b,L)$ |
| $q_1$ | $(q_2,0,R)$ L | $(q_N,1,S)$ | |
| $q_2$ | $(q_3,0,R)$ L | $(q_N,1,S)$ | $(q_N,b,S)$ |
| $q_3$ | $(q_Y,0,S)$ | $(q_N,1,S)$ | $(q_N,b,S)$ |

Ans: No, $q_0 \to q_1 \to q_2 \to q_N$
(執行過程)

∗ 30 can't be divided by 8. #

3. (1)

```
  Merge-sort(A, p, r)
1 if p<r
2    then q ← ⌊(p+r)/2⌋               O(1)
3        Merge-sort(A, p, q)          O(1)
4        Merge-sort(A, q+1, r)        T(n/2)
5        Merge(A, p, q, r)            O(n)
```

(3)

$$T(n)=T(\lceil \tfrac{n}{2} \rceil)+T(\lfloor \tfrac{n}{2} \rfloor)+cn,\ n \geq 2$$

$$=2T(\tfrac{n}{2})+cn$$

$$=2[2T(\tfrac{n}{2^2})+\tfrac{cn}{2}]+cn$$

$$=2^2T(\tfrac{n}{2^2})+2cn$$

$$=2^2[2T(\tfrac{n}{2^3})+\tfrac{cn}{2^2}]+2cn$$

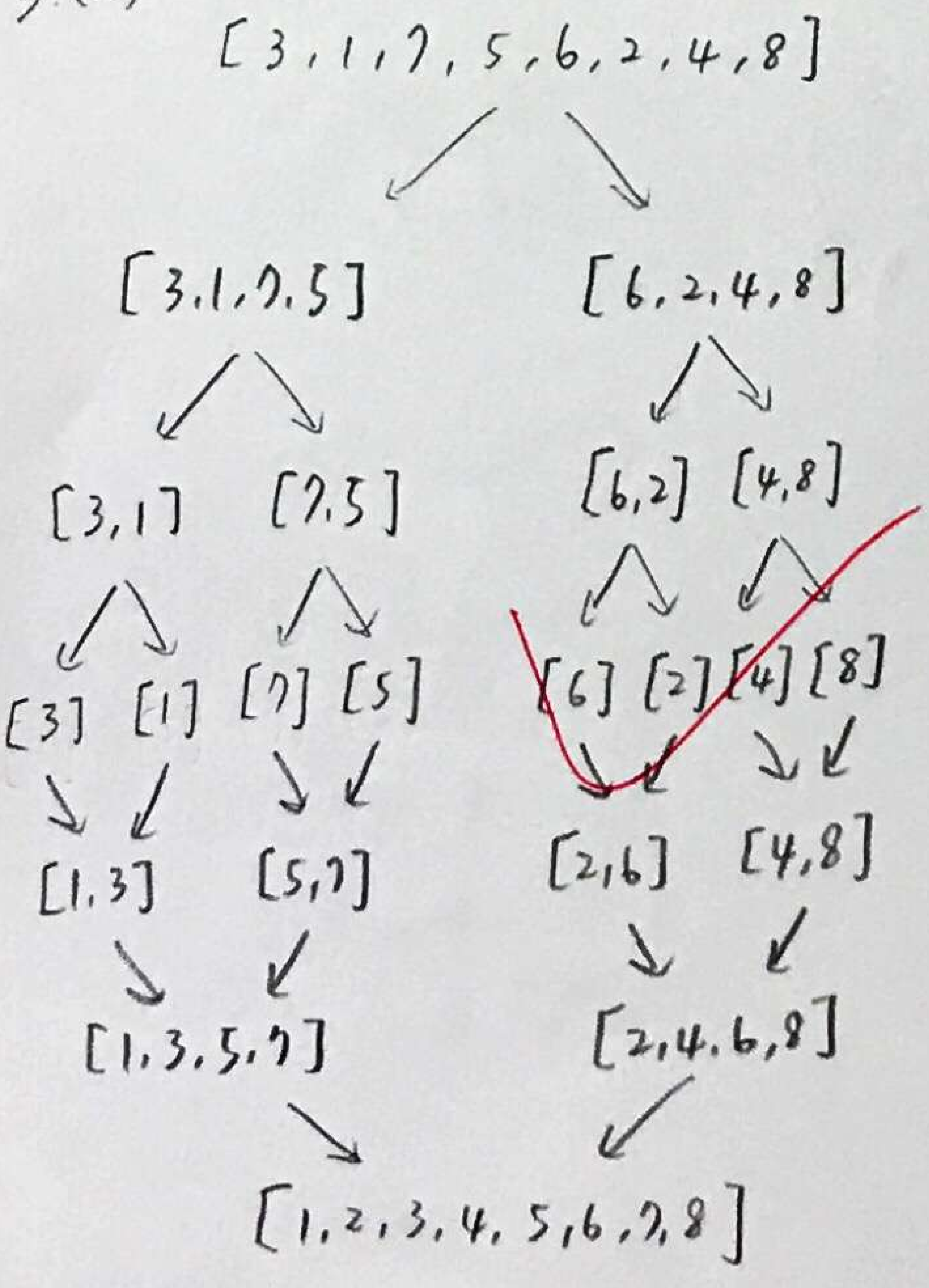
Let $n=2^k$, $k=\lg n$

$$=2^3T(\tfrac{n}{2^3})+3cn$$

$$\vdots$$

$$=2^kT(\tfrac{n}{2^k})+kcn = nT(1)+\lg n\cdot cn,\ \therefore T(n)=O(n\lg n)\ \#$$

3. (2)




4.

$$T(n)=2T(\tfrac{n}{2})+n\lg n$$

$$=2[2T(\tfrac{n}{2^2})+\tfrac{n}{2}\lg\tfrac{n}{2}]+n\lg n$$

$$=2^2T(\tfrac{n}{2^2})+n\lg n-n\lg 2+n\lg n = 2^2T(\tfrac{n}{2^2})+2n\lg n-n$$

$$=2^2[2T(\tfrac{n}{2^3})+\tfrac{n}{2^2}\lg\tfrac{n}{2^2}]+2n\lg n-n$$

$$=2^3T(\tfrac{n}{2^3})+n\lg n-n\lg 2^2+2n\lg n-n$$

Let $n=2^k$, $k=\lg n$

$$=2^3T(\tfrac{n}{2^3})+3n\lg n-3n$$

$$\vdots$$

$$=2^kT(\tfrac{n}{2^k})+kn\lg n-[\tfrac{k(k+1)}{2}]n$$

$$=nT(1)+\lg n\cdot n\lg n-[\tfrac{\lg^2 n+\lg n}{2}]n$$

$$=cn+n\lg^2 n-\tfrac{1}{2}n\lg^2 n+\tfrac{1}{2}n\lg n$$

$$=cn+\tfrac{1}{2}n\lg^2 n+\tfrac{1}{2}n\lg n$$

$$\therefore T(n)=O(n\lg^2 n)\ \#$$

k=1　k=2　k=3
0　1　1+2+3 … n

$\frac{n(n+1)}{2}$

5.
(8%) $T(n) = \Theta(1) + 8T(\frac{n}{2}) + \Theta(n^2)$

$= 8T(\frac{n}{2}) + \Theta(n^2)$

(8 個乘法，4 個 $\frac{n^2}{4}$ 加法 $= \Theta(n^2)$)

$T(n) = \Theta(n^3)$ (此時，divide-and-conquer 並未節省時間)

※ 改進：

Strassen's method

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ 7T(\frac{n}{2}) + \Theta(n^2) & \text{if } n>1 \end{cases}$$

(18 個加法)

$T(n) = \cdots \quad \Theta(n^{\lg 7}) = \Theta(n^{2.81})$

6.(1) heap tree：幾乎是完整二元樹(只有最後一層沒滿)，資料插入、取出最大值等，只需要 $O(\log n)$，可用於多工作業系統中，工作排程管理

6.(1) max-heap 是最大堆積，能將最大值堆積到 root，也就是 index 1 的位置

(2) ∵ 要從子樹開始檢查回 root 才能把最大值換到根，∴ 不行改為 for $i \leftarrow 1$ to $\lfloor length(A)/2 \rfloor$

(4%) ①② 若從 index 1 開始往下檢查，有可能最大值就不在 root，只能保証父節點都比子節點大

(2)③ 高度 $h$，個數為 $\frac{n}{(2^{h+1})}$，∴ 總花費時間為：

$$\sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right).$$

The last summation can be evaluated by substituting $x = \frac{1}{2}$ in the formula (A.8), which yields.

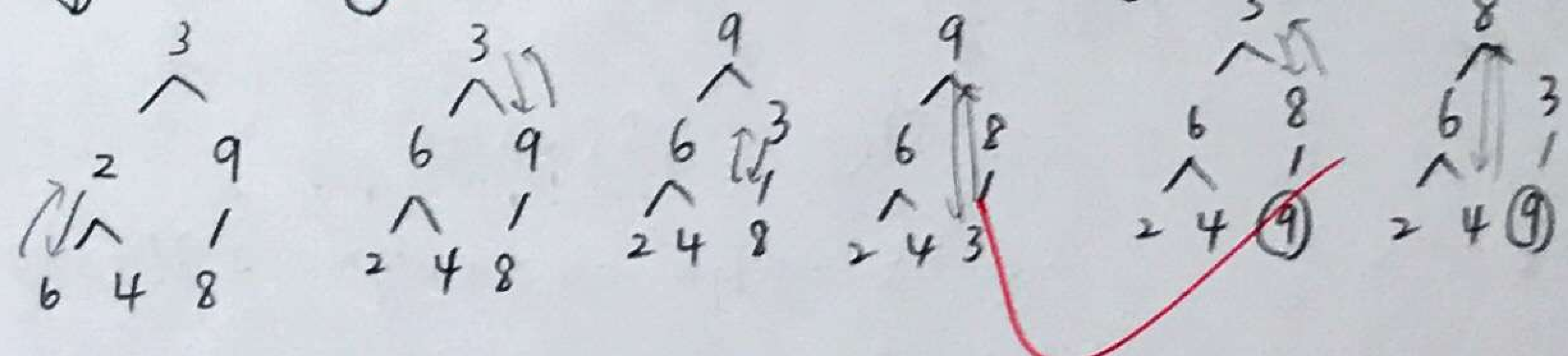
$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = 2$$

Thus, the running time of Build-Max-Heap can bounded as

$$O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) = O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right)$$

$= O(n)$

(3) ① ② ③



7.

```
RANDOMIZE-IN-PLACE(A)
n ← length[A]
i ← 1 to n
      do swap A[i] ↔ A[RANDOM(i,n)]
```

8. (1)

A: [3 | 2 | 1 | 5 | 3 | 2 | 2 | 1]

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | | | | | | | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 0 | 0 | 0 | 0 | 0 |

→ 統計

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 2 | 3 | 2 | 0 | 1 |

→ 累加

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 2 | 5 | 7 | 7 | 8 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 1 | | | | | | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 1 | 5 | 7 | 7 | 8 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 1 | | | 2 | | | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 1 | 4 | 7 | 7 | 8 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 1 | | 2 | 2 | | | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 1 | 3 | 7 | 7 | 8 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 1 | | 2 | 2 | | 3 | |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 1 | 3 | 6 | 7 | 8 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | | 1 | | 2 | 2 | | 3 | 5 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 1 | 3 | 6 | 7 | 7 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | 1 | 1 | | 2 | 2 | | 3 | 5 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 0 | 3 | 6 | 7 | 7 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | 1 | 1 | 2 | 2 | 2 | | 3 | 5 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 0 | 2 | 6 | 7 | 7 |

→

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| B | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 5 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 0 | 0 | 2 | 5 | 7 | 7 |

[1, 1, 2, 2, 2, 3, 3, 5] #

相同數值，先遇到，排前面；後遇到，排後面。

8(2) ∵ 數字由後往前放入B中，不会產生"先進後出"的問題

第7行 For $j \leftarrow n$ downto 1. 使順序由後往前放入B，故為 stable sorting.

9. (1)

(2%) ~~$(k-1) \le x \le (n-k)$~~

$R_1 = \frac{3}{10}n - 6$ , $R_2 = \frac{7}{10}n + 6$ #

(2)

1. $O(n)$
2. $O(n)$
3. $T(\lceil \frac{n}{5} \rceil)$
4. $O(n)$
5. $T(\frac{7}{10}n+6)$

$3(\frac{1}{2}\lceil\lceil\frac{n}{5}\rceil\rceil - 2) \le \frac{3}{10}n - 6$

$n - (\frac{3}{10}n - 6) = \frac{7}{10} + 6$

$$T(n) \le \begin{cases} O(1) & \text{if } n < 140 \\ T(\lceil \frac{n}{5} \rceil) + T(\frac{7}{10}n+6) + O(n) & \text{if } n \ge 140 \end{cases}$$

$$T(n) \le C(\lceil \frac{n}{5} \rceil) + T(\frac{7}{10}n+6) + an$$

$$= \frac{cn}{5} + c + \frac{7}{10}cn + 6c + an$$

$$= \frac{9}{10}cn + 7c + an$$

$$= cn + \underbrace{(-\frac{cn}{10} + 7c + an)}$$

at most $cn$, while $(-\frac{cn}{10} + 7c + an) \le 0$

$\therefore T(n) = O(n)$ #

10. (1)

(8%) ①

10(1) 括号數組:

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \ge 2 \end{cases}$$

The solution to the recurrence is $\Omega(2^n)$ #

②

line 4 $O(n)$

line 5 $O(n)$

line 6 $O(1)$

line 8 $O(n)$

10. (2)

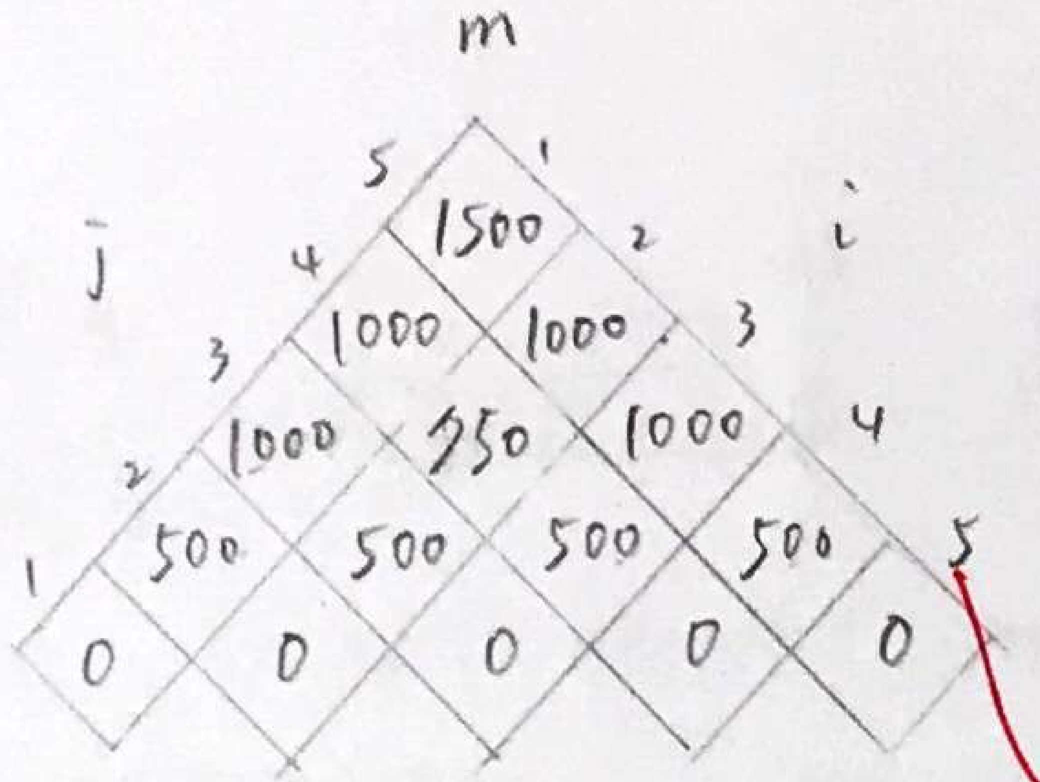
$A_1(10,5)$

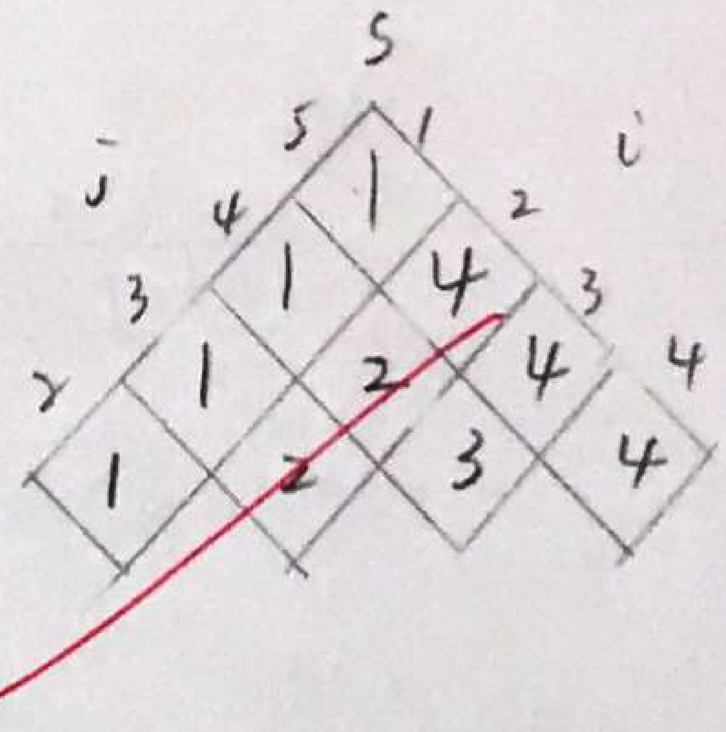
$A_2(5,10)$

$A_3(10,10)$

$A_4(10,5)$

$A_5(5,10)$





$m[1,2] = m[1,1] + m[2,2] + 10\times5\times10 = 500$

$m[2,3] = m[2,2] + m[3,3] + 5\times10\times10 = 500$

$m[3,4] = m[3,3] + m[4,4] + 10\times10\times5 = 500$

$m[4,5] = m[4,4] + m[5,5] + 10\times5\times10 = 500$

$m[1,3] = m[1,1] + m[2,3] + 10\times5\times10 = 0 + 500 + 500 = 1000$

$= m[1,2] + m[3,3] + 10\times10\times10 = 500 + 0 + 1000 = 1500$

$m[2,4] = m[2,2] + m[3,4] + 5\times10\times5 = 0 + 500 + 250 = 750$

$= m[2,3] + m[4,4] + 5\times10\times5 = 500 + 0 + 250 = 750.$

$m[3,5] = m[3,3] + m[4,5] + 10\times10\times10 = 0 + 500 + 1000 = 1500$

$= m[3,4] + m[5,5] + 10\times5\times10 = 500 + 0 + 500 = 1000$

$m[1,4] = m[1,1] + m[2,4] + 10\times5\times5 = 0 + 750 + 250 = 1000$

$= m[1,2] + m[3,4] + 10\times10\times5 = 500 + 500 + 500 = 1500$

$= m[1,3] + m[4,4] + 10\times10\times5 = 1000 + 0 + 500 = 1500$

$m[2,5] = m[2,2] + m[3,5] + 5\times10\times10 = 0 + 1000 + 500 = 1500$

$m[2,3] + m[4,5] + 5\times10\times10 = 500 + 500 + 500 = 1500$

$m[2,4] + m[5,5] + 5\times5\times10 = 750 + 0 + 250 = 1000$

$m[1,5] = m[1,1] + m[2,5] + 10\times5\times10 = 0 + 1000 + 500. = 1500$

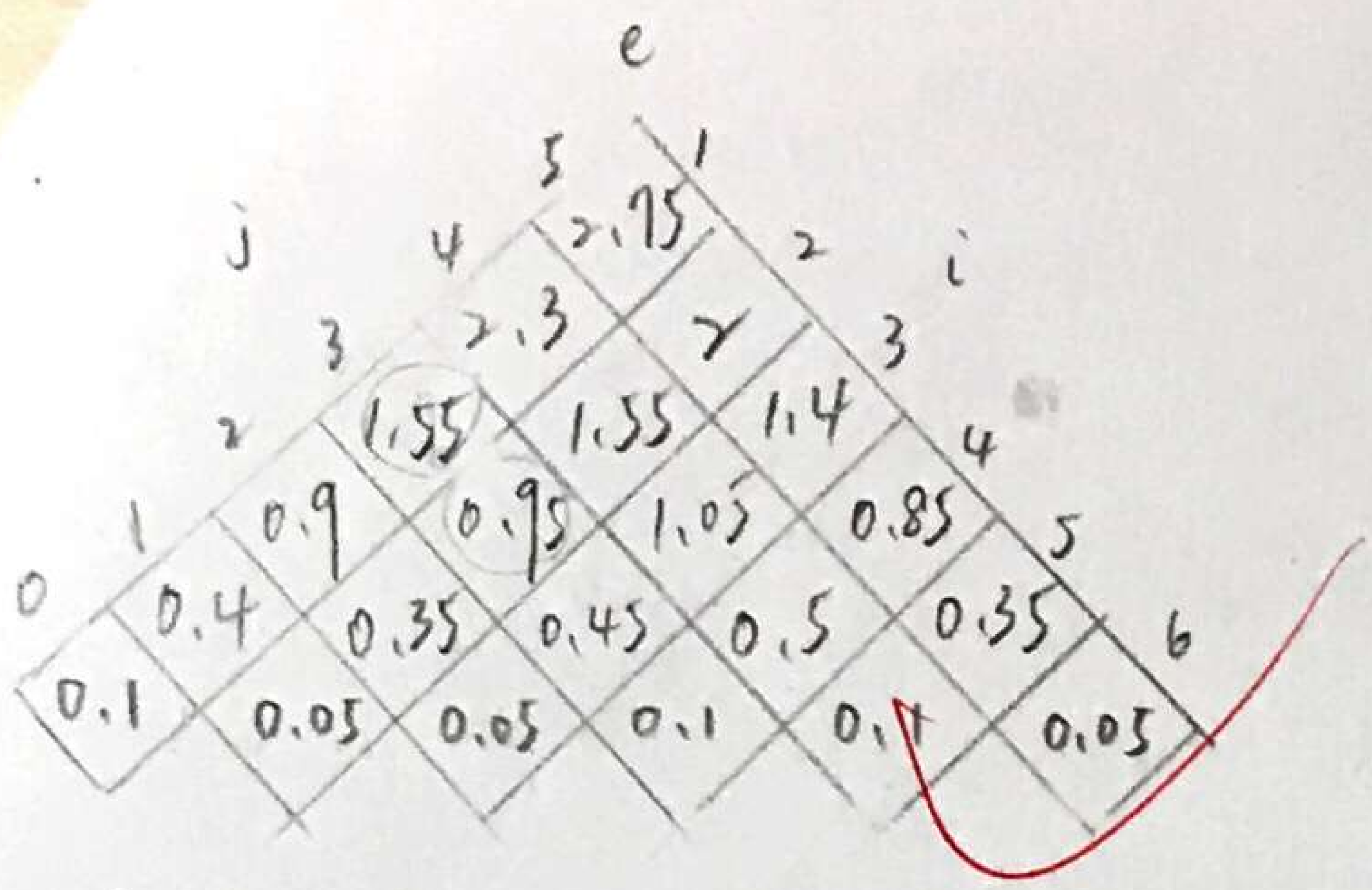
$= m[1,2] + m[3,5] + 10\times10\times10 = 500 + 1000 + 1000 = 2500$

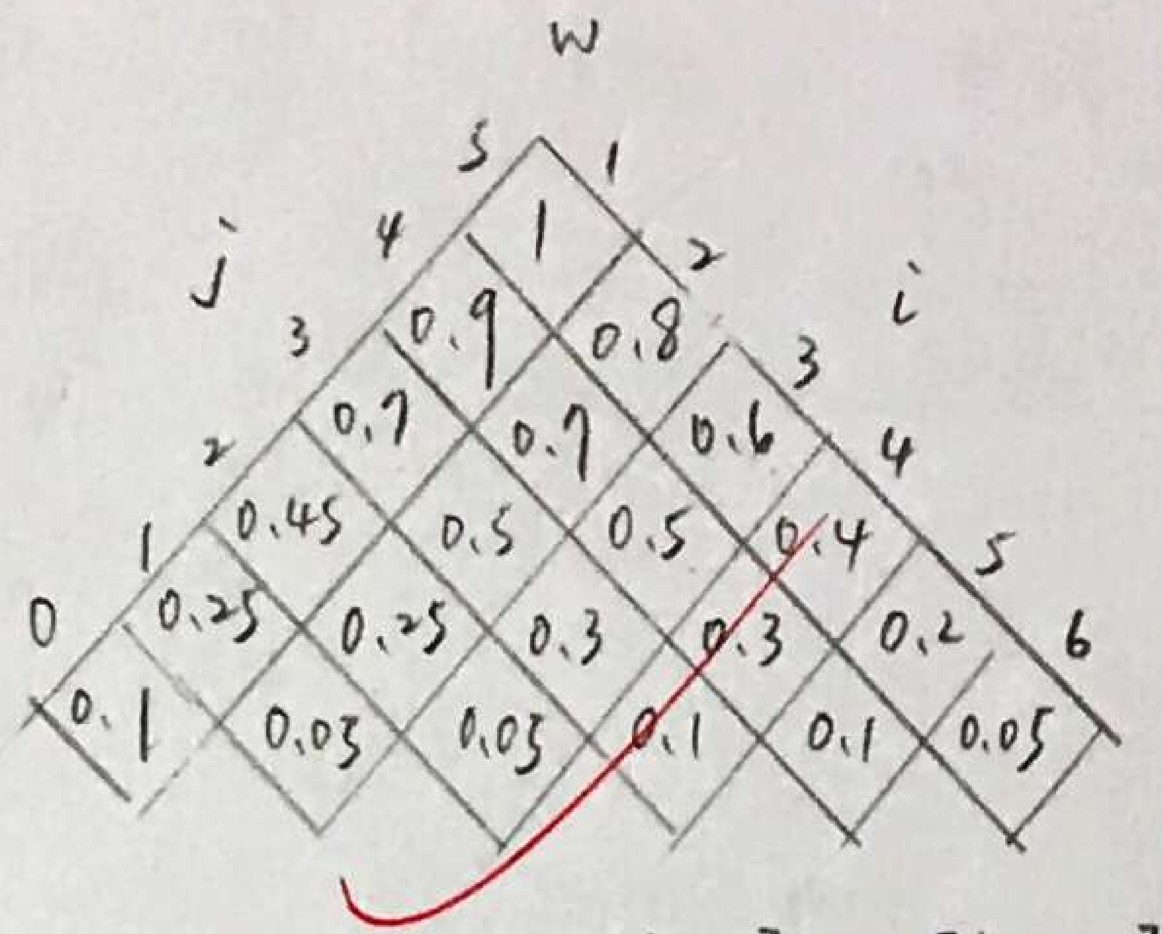
$= m[1,3] + m[4,5] + 10\times10\times10 = 1000 + 500 + 1000 = 2500$

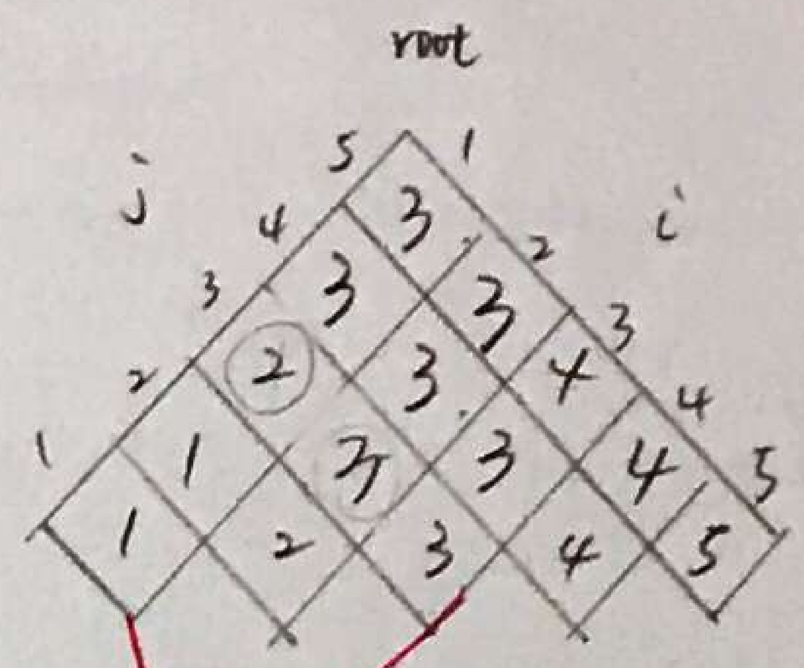
$= m[1,4] + m[5,5] + 10\times5\times10 = 1000 + 0 + 500 = 1500$

$(A_1((A_2(A_3A_4))A_5))$. 1500次

11、




$e[i,j] = e[i,r-1] + e[r+1,j] + w[i,j]$

$e[1,1] = e[\overset{0.1}{1,0}] + e[\overset{0.05}{2,1}] + w[\overset{0.25}{1,1}] = 0.4$

$e[2,2] = e[\overset{0.05}{2,1}] + e[\overset{0.05}{3,2}] + w[\overset{0.25}{2,2}] = 0.35$

$e[3,3] = e[\overset{0.05}{3,2}] + e[\overset{0.1}{4,3}] + w[\overset{0.3}{3,3}] = 0.45$

$e[4,4] = e[\overset{0.1}{4,3}] + e[\overset{0.1}{5,4}] + w[\overset{0.3}{4,4}] = 0.5$

$e[5,5] = e[\overset{0.1}{5,4}] + e[\overset{0.05}{6,5}] + w[\overset{0.2}{5,5}] = 0.35$

$e[1,2]$, $r=1$, $e[\overset{0.1}{1,0}] + e[\overset{0.35}{2,2}] + w[\overset{0.45}{1,2}] = 0.9$
$r=2$, $e[\overset{0.4}{1,1}] + e[\overset{0.05}{3,2}] + w[\overset{0.45}{1,2}] = 0.9$

$e[2,3]$, $r=2$, $e[\overset{0.05}{2,1}] + e[\overset{0.45}{3,3}] + w[\overset{0.5}{2,3}] = 1$
$r=3$, $e[\overset{0.35}{2,2}] + e[\overset{0.1}{4,3}] + w[\overset{0.5}{2,3}] = 0.95$ ✓

$e[3,4]$, $r=3$, $e[\overset{0.05}{3,2}] + e[\overset{0.5}{4,4}] + w[\overset{0.5}{3,4}] = 1.05$
$r=4$, $e[\overset{0.45}{3,3}] + e[\overset{0.1}{5,4}] + w[\overset{0.5}{3,4}] = 1.05$

$e[4,5]$, $r=4$, $e[\overset{0.1}{4,3}] + e[\overset{0.35}{5,5}] + w[\overset{0.4}{4,5}] = 0.85$
$r=5$, $e[\overset{0.5}{4,4}] + e[\overset{0.05}{6,5}] + w[\overset{0.4}{4,5}] = 0.95$

$e[1,3]$, $r=1$, $e[\overset{0.1}{1,0}] + e[\overset{0.95}{2,3}] + w[\overset{0.7}{1,3}] = 1.75$
$r=2$, $e[\overset{0.4}{1,1}] + e[\overset{0.45}{3,3}] + w[\overset{0.7}{1,3}] = 1.55$
$r=3$, $e[\overset{0.9}{1,2}] + e[\overset{0.1}{4,3}] + w[\overset{0.7}{1,3}] = 1.7$

$e[2,4]$, $r=2$, $e[\overset{0.05}{2,1}] + e[\overset{1.05}{3,4}] + w[\overset{0.7}{2,4}] = 1.8$
$r=3$, $e[\overset{0.35}{2,2}] + e[\overset{0.5}{4,4}] + w[\overset{0.7}{2,4}] = 1.55$
$r=4$, $e[\overset{0.95}{2,3}] + e[\overset{0.1}{5,4}] + w[\overset{0.7}{2,4}] = 1.75$

$e[3,5]$, $r=3$, $e[\overset{0.05}{3,2}] + e[\overset{0.85}{4,5}] + w[\overset{0.6}{3,5}] = 1.5$
$r=4$, $e[\overset{0.45}{3,3}] + e[\overset{0.35}{5,5}] + w[\overset{0.6}{3,5}] = 1.4$
$r=5$, $e[\overset{1.05}{3,4}] + e[\overset{0.05}{6,5}] + w[\overset{0.6}{3,5}] = 1.7$

$e[1,4]$, $r=1$, $e[\overset{0.1}{1,0}] + e[\overset{1.55}{2,4}] + w[\overset{0.9}{1,4}] = 2.55$
$r=2$, $e[\overset{0.4}{1,1}] + e[\overset{1.05}{3,4}] + w[\overset{0.9}{1,4}] = 2.35$
$r=3$, $e[\overset{0.9}{1,2}] + e[\overset{0.5}{4,4}] + w[\overset{0.9}{1,4}] = 2.3$
$r=4$, $e[\overset{1.55}{1,3}] + e[\overset{0.1}{5,4}] + w[\overset{0.9}{1,4}] = 2.55$

$w[i,j] = w[i,j-1] + p_j + q_j$

$w[1,1] = w[\overset{0.1}{1,0}] + 0.1 + 0.05 = 0.25$

$w[2,2] = 0.05 + 0.15 + 0.05 = 0.25$

$w[3,3] = 0.05 + 0.15 + 0.1 = 0.3$

$w[4,4] = 0.1 + 0.1 + 0.1 = 0.3$

$w[5,5] = 0.1 + 0.05 + 0.05 = 0.2$

$w[1,2] = 0.25 + 0.15 + 0.05 = 0.45$

$w[2,3] = 0.25 + 0.15 + 0.1 = 0.5$

$w[3,4] = 0.3 + 0.1 + 0.1 = 0.5$

$w[4,5] = 0.3 + 0.05 + 0.05 = 0.4$

$w[1,3] = 0.45 + 0.15 + 0.1 = 0.7$

$w[2,4] = 0.5 + 0.1 + 0.1 = 0.7$

$w[3,5] = 0.5 + 0.05 + 0.05 = 0.6$

$w[1,4] = 0.7 + 0.1 + 0.1 = 0.9$

$w[2,5] = 0.7 + 0.05 + 0.05 = 0.8$

$w[1,5] = 0.9 + 0.05 + 0.05 = 1$

$e[2,5]$, $r=2$, $e[\overset{0.05}{2,1}] + e[\overset{1.4}{3,5}] + w[\overset{0.8}{2,5}] = 2.25$
$r=3$, $e[\overset{0.35}{2,2}] + e[\overset{0.85}{4,5}] + w[\overset{0.8}{2,5}] = 2$
$r=4$, $e[\overset{0.95}{2,3}] + e[\overset{0.35}{5,5}] + w[\overset{0.8}{2,5}] = 2.1$
$r=5$, $e[\overset{1.55}{2,4}] + e[\overset{0.05}{6,5}] + w[\overset{0.8}{2,5}] = 2.4$

$e[1,5]$, $r=1$, $e[\overset{0.1}{1,0}] + e[\overset{2}{2,5}] + w[\overset{1}{1,5}] = 3.1$
$r=2$, $e[\overset{0.4}{1,1}] + e[\overset{1.4}{3,5}] + w[\overset{1}{1,5}] = 2.8$
$r=3$, $e[\overset{0.9}{1,2}] + e[\overset{0.85}{4,5}] + w[\overset{1}{1,5}] = 2.75$
$r=4$, $e[\overset{1.55}{1,3}] + e[\overset{0.35}{5,5}] + w[\overset{1}{1,5}] = 2.9$
$r=5$, $e[\overset{2.3}{1,4}] + e[\overset{0.05}{6,5}] + w[\overset{1}{1,5}] = 3.35$