# Department of Computer Science and Engineering
# National Sun Yat-sen University
# Design and Analysis of Algorithms - Final Exam., Jan. 13, 2015

1. Multiple choices (There may be zero or more correct answers. If there is no correct answer, you should write down "None".) (28%)

   (a) Which statement(s) is correct? (A) If problem *A* is NP-hard, then *A* is NP-complete. (B) If problem *A* is NP-complete, then *A* is NP-hard. (C) If problem *A* is NP-complete, then *A* has no polynomial time algorithm in the worst case. (D) If problem *A* is a P problem, then *A* is an NP problem.

   (b) Which statement(s) is correct? (A) If problem *A* is NP-complete and problem *B* is NP-hard, then *A* polynomially reduces to *B*. (B) If problem *A* is NP-complete and problem *B* is NP-hard, then *B* polynomially reduces to *A*. (C) If problem *A* is NP-complete and problem *B* is P, then *A* polynomially reduces to *B*. (D) If problem *A* is NP-complete and problem *B* is P, then *B* polynomially reduces to *A*.

   (c) Consider the time complexity in the worst case. Let $n$ denote the input size of the following given problem. Which statement(s) is correct? (A) The time complexity of quicksort is $O(n\log n)$. (B) The time complexity of insertion sort is $O(n^2)$. (C) The time complexity of binary search is $O(\log n)$. (D) The time complexity of FFT is $O(n\log n)$.

   (d) Which statement(s) is correct? (A) In a connected graph, for any two nodes *u* and *v*, there is at least one path connecting them. (B) The minimum spanning tree of a connected graph $G = (V,E)$, where $|V| = n$, has exactly $n-1$ edges. (C) In a minimum spanning tree of a connected graph $G = (V,E)$, where $|V| = n$, each tree edge has cost no more than each non-tree edge. (D) The minimum spanning tree of a connected graph is also connected.

   (e) Which statement(s) is correct for the searching strategy? (A) The queue data structure should be used in depth-first search for guiding the search. (B) The queue data structure should be used in breadth-first search for guiding the search. (C) The priority queue data structure should be used in best-first search for guiding the search. (D) The hill climbing method is a variation of depth-first search.

   (f) Which statement(s) is correct for an AVL tree? (A) The level difference of every two nodes is never more than 1. (B) The height difference of the two subtrees of each node is at most 1. (C) If a new node is inserted without any rotation, then the tree height will increase by 1. (D) When a new node is

added, it is always inserted as a leaf node.

(g) Which statement(s) is correct? (A) If $a$ mod $p = b$ mod $p$, then $(a-b)$ mod $p = 0$. (B) If $(a+x)$ mod $p = (b+y)$ mod $p$, and $(a-b)$ mod $p = 0$, then $x = y$. (C) 4 is a quadratic residue mod 9. (D) 7 is a quadratic residue mod 9.

2. (a) Explain the Eulerian cycle of a graph. (6%)

   (b) What is the necessary and sufficient condition for a graph having an Eulerian cycle? (6%)

3. Design an algorithm to find both the minimum and the maximum of $n$ elements with at most $\left\lfloor \frac{3n}{2} \right\rfloor$ comparisons. (12%)

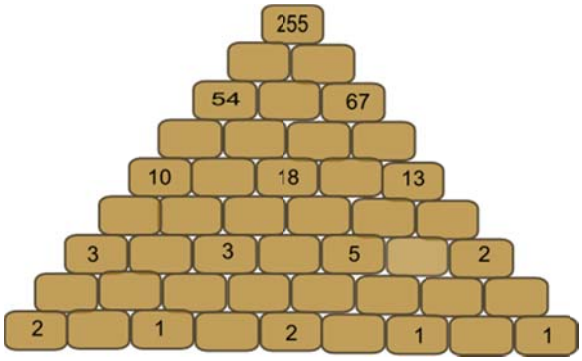
4. (a) Explain the longest common subsequence (LCS) problem. And, then give an example to illustrate your answer. Note that you should give both explanation and example. (6%)

   (b) Give the dynamic programming method for solving the LCS problem. (6%)

5. The first-fit algorithm is a simple approximation algorithm for solving the bin packing problem. The algorithm puts an object into the $i$th bin as long as it is available and tries the $(i+1)$th bin if otherwise. Show that the number of bins resulting from the first-fit algorithm is no more than twice the number of bins needed in an optimal solution. (12%)

6. Prove that the partition decision problem polynomially reduces to the bin packing decision problem. (12%)

7. There are 9 rows in the magic triangle, where row $i$ has $i$ cells, shown as follows. The number in each cell is the sum of the two cells below it. For example, 255 is the sum of the two cells in row 2, not shown in the figure. Please calculate all missing numbers and fill in the triangle. Your answer has been written in 9 rows, each row (from left to right) corresponds to one row of the triangle, from top to bottom. Note your answer must include the numbers already in the figure. (12%)



Answer:
1. BD, AD, BCD, ABD, BCD, BD, ACD

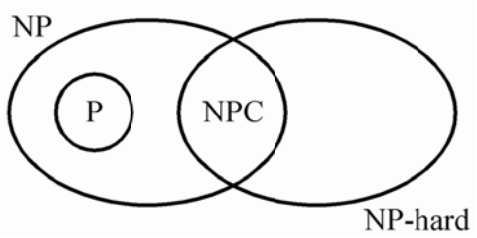
7.

255
121 134
54 67 67
23 31 36 31
10 13 18 18 13
5 5 8 10 8 5
3 2 3 5 5 3 2
2 1 1 2 3 2 1 1
2 0 1 0 2 1 1 0 1

**Design and Analysis of Algorithms**
**Final Exam., Jan. 13, 2015 參考解答**

**1. Multi choices**

**(a) Ans：B、D**




(A) 由上述圖形，可知 NP-complete 是 NP-hard 和 NP 的交集，所以若 A 是 NP-hard，並不一定是 NP-complete。

(B) NP-complete : the class of problems which are NP-hard and belong to NP。
由 NP-complete 的定義，可得知：若 A 是 NP-complete，那麼 A 也是 NP-hard。

(C) 目前沒有 polynomial time algorithm 來解決 NP-complete 的問題，但不代表未來沒有。

(D) P 問題可以用 polynomial 演算法來解決的問題，NP 可以用 non-deterministic polynomial 演算法來解決的問題。P 問題既然可以用 polynomial 演算法來解決，當然也可以用 non-deterministic polynomial 演算法來解決問題，因此所有的 P 問題都是 NP 問題。

**(b) Ans：A、D**

(A) NP-complete : the class of problems which are NP-hard and belong to NP。
NP-hard: the class of problems to which every NP problem reduces。
因爲 A 是 NP-complete，所以 A 也是 NP。而且 B 是 NP-hard，所以 A(NP)可以 reduces to B。

(B) 因爲 A 是 NP-complete，所以 A 也是 NP 跟 NP-hard。
但 B 是 NP-hard，所以 B 不一定可以 reduces to A。

(C)應是 B reduce to A，而非 A reduce to B

(D)因爲 B 是 P，所以 B 也是 NP，且所有 NP 可以 reduces to NP-hard，所以 B 可以 reduce to A。

**(c) Ans：B、C、D**

此題討論的是 worst case 下的時間複雜度:

(A) Quicksort 的 worst case 是 $O(n^2)$。
In each round, the number used to split is either the smallest or the largest.

$$n+(n-1)+\cdots+1=\frac{n(n+1)}{2}=O(n^2)$$

(B) insertion sort 的 worst case 是 $O(n^2)$。

Analysis of # of movements in worst case: reversely sorted

$d_1 = n - 1$

$d_2 = n - 2$

:

$d_i = n - i$

$d_n = 0$

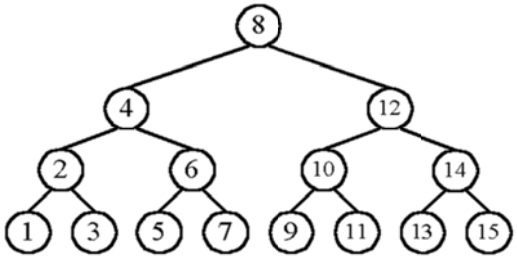
M: # of data movements in straight

$$M = \sum_{i=1}^{n-1}(2+d_i) = 2(n-1) + \frac{n(n-1)}{2} = O(n^2)$$

(C) binary search 的 worst case 是 O(logn)。

worst case(即找不到的情況):須由 root 搜尋到 leaf 共 $(\lfloor \log_2 n \rfloor + 1)$ steps = O(log n)

例:




共 $(\lfloor \log_2 15 \rfloor + 1)$ steps

(D) FFT 的 worst case 是 O(nlogn)。

FFT (Divide-and-conquer)

Input: $a_0, a_1, \ldots, a_{n-1}$, $n = 2k$

Output: $b_j$, $j=0, 1, 2, \ldots, n-1$

where $b_j = \sum_{0 \le k \le n-1} a_k w^{kj}$, where $w = e^{i2\pi/n}$

Step 1: If n=2, compute

$b_0 = a_0 + a_1$,

$b_1 = a_0 - a_1$, and return.

Step 2: Recursively find the Fourier transform of $\{a_0, a_2, a_4, \ldots, a_{n-2}\}$ and $\{a_1, a_3, a_5, \ldots, a_{n-1}\}$, whose results are denoted as $\{c_0, c_1, c_2, \ldots, c_{n/2-1}\}$ and $\{d_0, d_1, d_2, \ldots, d_{n/2-1}\}$.

Step 3: Compute $b_j$:

$b_j = c_j + w^j d_j$ for $0 \le j \le n/2 - 1$

$b_{j+n/2} = c_j - w^j d_j$ for $0 \le j \le n/2 - 1$.

Time complexity:

$T(n) = 2T(n/2) + O(n)$

$= O(n \log n)$

**(d) Ans：A、B、D**

(A)Connected graph: there is a path from any point to any other point in the graph.

根據 connected graph 定義明顯可得任兩點之間至少存在一條路徑。

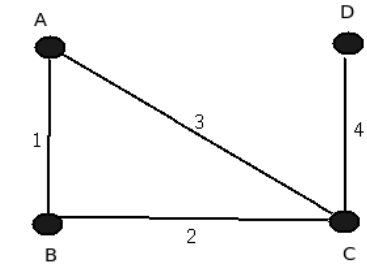
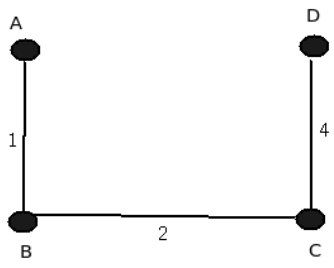
(B)Spanning tree:在 graph 中，是一個 subgraph 也是一個 tree，而且包含 graph 的所有 vertices 並彼此都相連。

Minimum spanning tree(MST): 在 graph 中，是一個 Spanning tree 且其中所有 edge 的 cost 總和是所有 MST 中最小的。

根據 MST 的定義可知 MST 是一個 tree 所以不會有 cycle，而且包含所有 vertices(v)因此其 edges 的數量等於|v|-1。

(C) minimum spanning tree 是 edge 的 cost 總和最小並不代表選的 edge 的 cost 一定比沒選的 edge 的 cost 大，若 edge 造成 cycle 則不會被選中。

例:

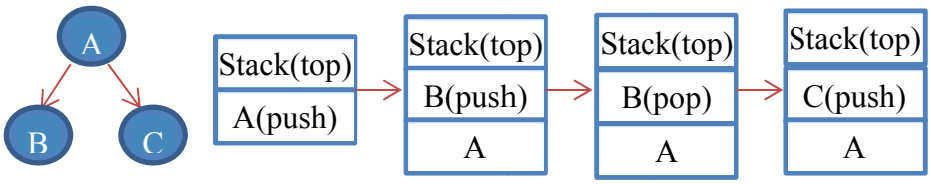
| Graph | MST |
|---|---|
|  |  |
| *CD* 爲 MST edge, *AC* 爲 non-tree edge，但是 *cost(C, D) > cost(A, C)* | |

(D)若原 graph 是 connected，則根據 MST 的定義，其所包含的 vertices 和原 graph 相同，明顯可得 MST 也是 connected。

**(e) Ans：B、C、D**

(A) DFS(depth-first search)：以某一節點爲出發點，不斷地前進拜訪未曾被拜訪過的節點，直到無路可走(到達 leaf node)或是所有相鄰的節點(brother node)都已經拜訪過爲止，然後再退回前一個節點(father node)，尋找沒有拜訪過的節點，直到所有相鄰的節點都已被拜訪過。正因爲在 DFS 中須退回到前一個節點，所以必須記錄經過的所有 node；而且每次取出都是前一個拜訪過的結點，所以 DFS 適合使用 stack (first in last out)而非 queue

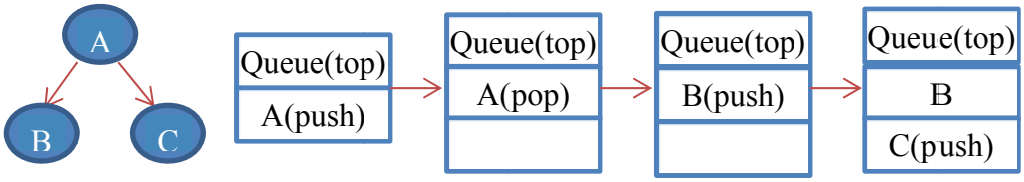
Example: 從 A 開始 DFS




(B) BFS(breadth-first search)：

在拜訪節點時，會將同一層的節點展開完畢後，才會再展開下一層。正因爲 BFS 會由上而下一曾一層的展開，因此對於該層的父節點不需要再度拜訪，所以 BFS 適合使用 queue(first in first out)，在展開下一層時，可以將該層的父節點給去除。

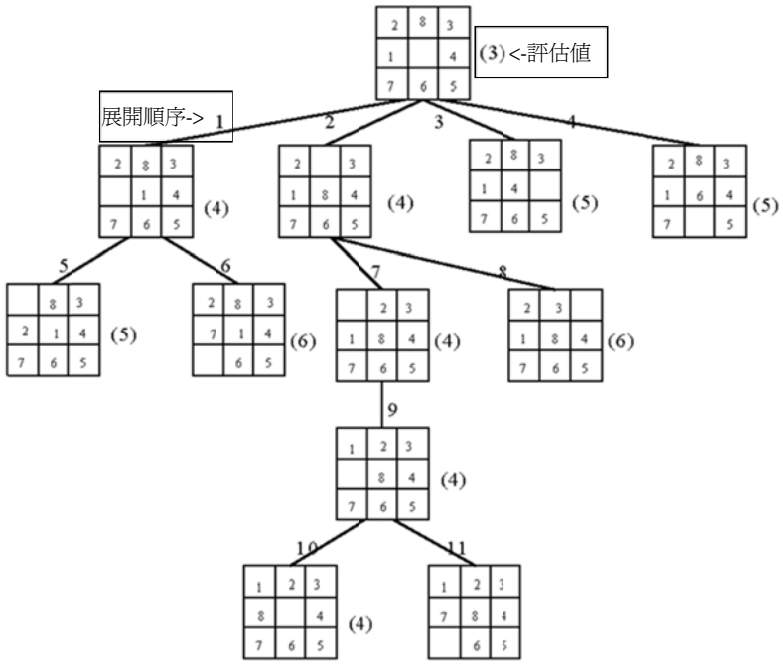
Example: 從 A 開始 BFS




**(C)** Best-first search(結合DFS&BFS)：

在拜訪節點時，檢查目前展開的所有節點，但其son 尚未展開者，選擇評估值最佳者，爲下一個欲展開的節點。

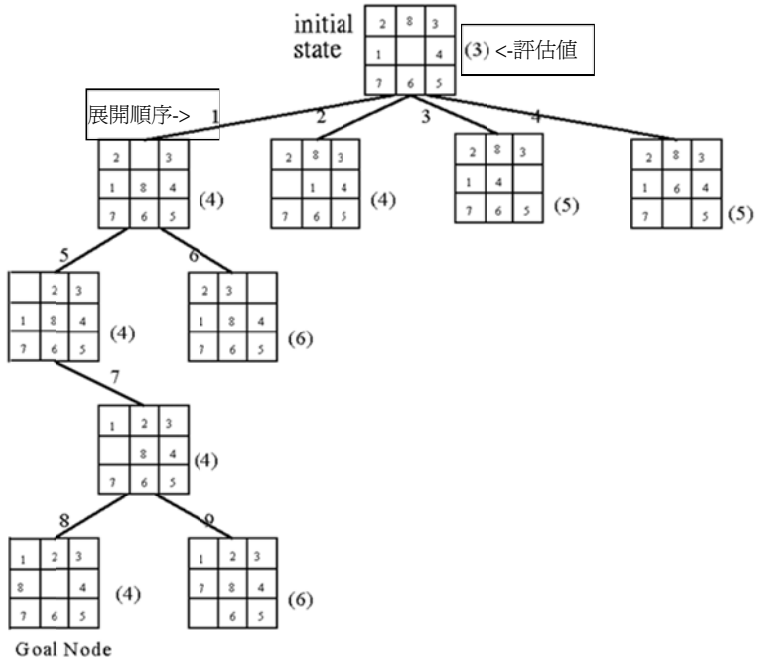
Example: An 8-puzzle problem




(D) Hill climbing：

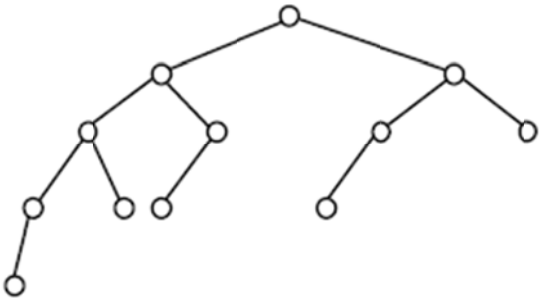
爲DFS 的變形，在展開node x 的children 之後，先給予每個children 一個評估值，以評估值最佳者爲root繼續recursively展開。

Example: An 8-puzzle problem




**(f) Ans：B、D**

(A)反例如下：此 AVL-tree，兩個 leaves 高度差可能超過 1。

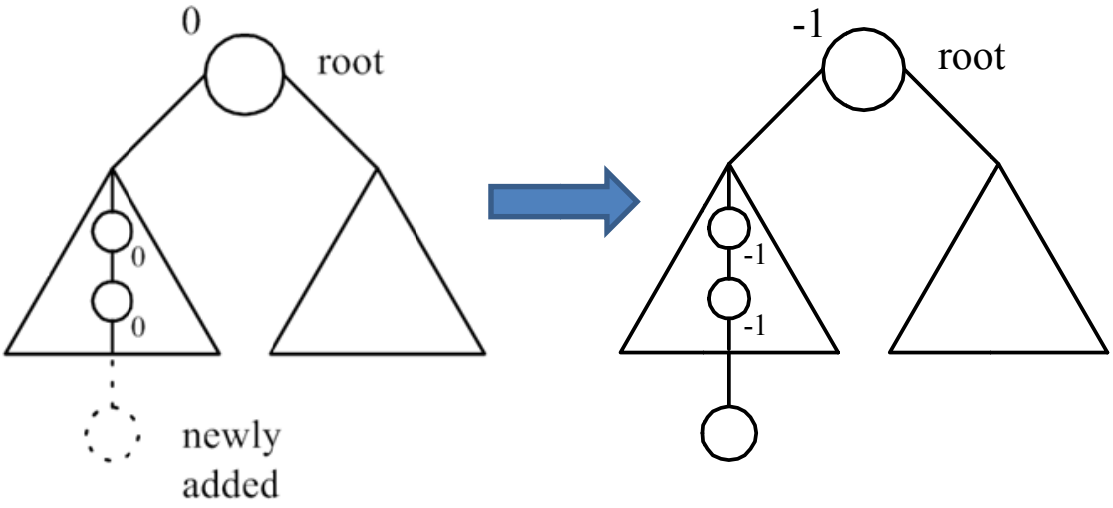


(B)AVL-tree 是一個高度平衡的二元搜尋樹，根據其 height balance of node v 的定義:

hb(v)= (height of right subtree) – (height of left subtree)

The hb(v) of every node never differ by more than 1.

(C)高度會增加一的情況，只有當新增加的節點一路往上看到 root 都是 0，高度才會在節點加入之後加一。




**(D)**因爲 AVL-tree 也是 Binary Search Tree 因此其 insert 的方法和 Binary Search Tree 一樣，必定從 leaf node 插入，但之後會進行調整來保持高度平衡。

**(g) Ans：A、C、D**

(A) 因爲 a 和 b mod p 的餘數相同，所以 a-b 必定是 p 的倍數且$(a-b) \bmod p = 0$

例: a=16，b=10，p=3，

a mod p = 16 mod 3 =(3*5+1) mod 3

b mod p = 10 mod 3 =(3*3+1) mod 3

則(16-10) mod 3=(3*5+1-3*3-1) mod 3 =(3*2) mod 3 =0

(B) $x = y$ 要改成 $x \bmod p = y \bmod p$

(C)&(D)

$QR = \{(x, y) \mid y \text{ is a quadratic residue mod } x\}$

$QNR = \{(x, y) \mid y \text{ is a quadratic non-residue mod } x\}$

$1^2 \equiv 1 \bmod 9$　　$2^2 \equiv 4 \bmod 9$

$3^2 \equiv 0 \bmod 9$　　$4^2 \equiv 7 \bmod 9$

$5^2 \equiv 7 \bmod 9$　　$6^2 \equiv 0 \bmod 9$

**2.(a)** 假設給予一 graph G(V,E)

Eulerian cycle: 即爲從 G 中任何一個頂點(vi∈ V)出發，經過所有的邊(ej∈ E)，而且只能經過一次，最後再回到原出發頂點的路徑。

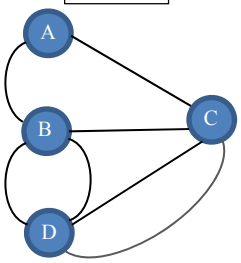
**2.(b)**

Eulerian cycle 的充分且必要條件: 因爲 Eulerian cycle 必須經過所有的邊回到出發的頂點，所以對於經過的頂點而言，最少必須一進一出，所有頂點的 degree 必需均爲偶數。
(注:degree:爲該頂點的臨邊數量)

Example:

以圖一爲例:

圖一




(1) Eulerian cycle:

由 A 點出發:

$A \rightarrow B \rightarrow D \rightarrow C \rightarrow D \rightarrow B \rightarrow C \rightarrow A$

(2) degree:

A 的 degree=2

B 的 degree=4

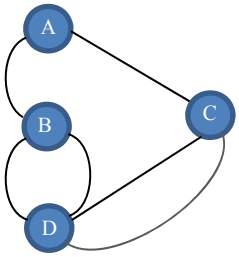
C 的 degree=4

D 的 degree=4

=> 皆爲偶數

以圖二爲例:

圖二




圖二是將圖一中的某條邊(B-C)給刪除，明顯可得：

不存在 Eulerian cycle

**3.**

演算法如下:

Step1:

將 n 個元素兩兩比較，分成大的一群$S_{max}$ ($|S_{max}| = \lceil n/2 \rceil$)，小的一群$S_{min}$($|S_{min}| = \lfloor n/2 \rfloor$)，(例如:A 和 B 比較(A>B)，A 須放到$S_{max}$，B 須放到$S_{min}$)

需要比較$\lfloor n/2 \rfloor$次，若元素爲奇數個，則把最後一個元素分到大的那一堆。

Step2:

從$S_{max}$中找出最大值，需比較$\lceil n/2 \rceil$-1 次

Step3:

從$S_{min}$中找出最小值，需比較$\lfloor n/2 \rfloor$次

經過 Step1~3 總共最多需比較⌊3n/2⌋次

**4.(a)**

Input: Two sequences A = c a b a b c and B = c d a a c c

(1) Subsequence of A: 將 A 裡面 0 個或多個元素刪掉(這些刪掉的元素可連續或不連續) 剩下的元素組成的 sequence 爲 A 的 subsequence

Example: sequence A= c a b a b c
則 cabbc(去掉第 4 個字母)、caac(去掉兩個 b) 、cabab，皆爲 A 的 subcequence

(2) Common subsequence of A and B: 當一個 A 的 subsequence 等同於 一個 B 的 subsequence 則這個 subsequence 稱爲 A 和 B 的 Common subsequence

Example: sequence A= c a b a b c and B = c d a a c c
則 ca、cc、caa、cacc 皆爲 A 和 B 的 Common subsequence

(3) longest common subsequence (LCS) of A and B: A sequence S, S 同時是 A 的 subsequence 也是 B 的 subsequence，而且 S 長度是所有 common subsequences 中最長的。

Example: sequence A= c a b a b c and B = c d a a c c
LCS = caac

**4.(b)**

Let $A = a_1\ a_2 \ldots a_m$ and $B = b_1\ b_2 \ldots b_n$
Let $L_{i,j}$ denote the length of the longest common subsequence of $a_1\ a_2 \ldots a_i$ and $b_1\ b_2 \ldots b_j$

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & \text{if } a_i = b_j \\ \max\{ L_{i-1,j}, L_{i,j-1} \} & \text{if } a_i \neq b_j \end{cases}$$

$L_{0,0} = L_{0,j} = L_{i,0} = 0$ for $1 \le i \le m$, $1 \le j \le n$.

由上述公式可求得 m*n 的矩陣 L，並可從 L[m][n]中得到 LCS 的長度，再利用 $L_{i,j}$ 矩陣，使用 backtracking 技術可以求得 LCS 之內容。

Example: sequence A= b a c a d and B = a c c b a d c b

| | | B | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | a | c | c | b | a | d | c | b |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| b | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| a | 0 | 1 | ←1 | 1 | 1 | 2 | 2 | 2 | 2 |
| A c | 0 | 1 | 2 | 2 | ←2 | 2 | 2 | 3 | 3 |
| a | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| d | 0 | 1 | 2 | 2 | 2 | 3 | 4 | ←4 | ←4 |

LCS 長度= 4,LCS= acad

**5.**

The bin packing problem:

n items $a_1, a_2, \ldots, a_n$, $0< a_i \le 1$, $1 \le i \le n$,

to determine the minimum number of bins of unit capacity(容量=1) to accommodate all n items.

Notations:

$S(a_i)$: the size of item $a_i$

OPT: # of bins used in an optimal solution

m: # of bins used in the first-fit algorithm

C(Bi): the sum of the sizes of $a_j$'s packed in bin $B_i$ in the first-fit algorithm

Step1:

總共有 n 個 item，且因爲所有 bins 的大小一定包含所有 items 的大小

所以可得:

$$\text{OPT}*1 \ge \sum_{i=1}^{n} C(a_i) \quad (1)$$

Step2:

在 $B_{(\text{first-fit algorithm 中的 bins})}$中任意兩個 bins 的 size 和必大於一；

如果不大於一，則兩個 bins 裡的 items 可以放到同一個 bin 之中，根據 first-fit algorithm

所以可得:

$C(Bi) + C(Bi+1) > 1$ (2)

$\sum_{i=1}^{m} C(B_i)$=C(B1)+C(B2)+…+C(Bm) > m/2 (3)

(由式子(2)推得)

$m < 2\sum_{i=1}^{m} C(B_i)$ (4)

Step3:

因爲放於桶子的量等於所有東西的大小的總和

$$2\sum_{i=1}^{m} C(B_i) = 2\sum_{i=1}^{n} S(a_i) \quad (5)$$

最後根據式子(1)和(5)可推得:

$m < 2\sum_{i=1}^{m} C(B_i) = 2\sum_{i=1}^{n} S(a_i) \le 2\text{OPT}$

得證: m < 2OPT 即在 first-fit algorithm 中使用的 bins 數小於在 optimal solution 中使用的 bins 數

**6.**

partition $\propto$ bin packing

instance of partition：

給予 $n$ 個數 $S=\{a_1,a_2,\cdots,a_n\}$，每一 $a_i$ 皆爲正整數，並且滿足

$$\sum_{a_i\in P} a_i = \sum_{a_i\notin P} a_i \ ，\ P\subseteq S$$

instance of bin packing problem：

$B$ 爲箱子的數量，$C$ 爲箱子的容量，$c_i$ 爲項目的大小，$1\le i\le n$。

令 $B=2$ 、 $C=\sum_{1\le i\le n} a_i/2$ 、 $c_i=a_i$ $(1\le i\le n)$

**(1)** Partition => bin packing

如果存在一個集合 $P$ 滿足 $\sum_{a_i\in P} a_i = \sum_{a_i\notin P} a_i$ ，則

$$\sum_{1\le i\le n} a_i/2 = \sum_{a_i\in P} a_i = \sum_{a_i\notin P} a_i = C \quad (\because \sum_{a_i\in P} a_i + \sum_{a_i\notin P} a_i = \sum_{1\le i\le n} a_i)$$

所以可以把 $a_i$ 成兩堆，令 $c_i=a_i, a_i\in P$ 放在箱子 $B_1$ ， $c_i=a_i, a_i\notin P$ 放在箱子 $B_2$ ，滿足 $\sum_{c_i\in B_1} c_i = \sum_{c_i\in B_2} c_i \le C$。因此，若存在 partition，則 bin packing problem 可找到一種組合方法，能夠把 $n$ 個項目裝在兩個箱子中。

**(2)** Partition <= bin packing

如果能夠用兩個箱子把 $n$ 個項目 $c_1,c_2,\cdots,c_n$ 裝起來，其中 $\sum_{1\le i\le n} c_i = 2C$，則在箱子 $B_1$ 中每一個項目的容量加總爲 $\sum_{c_i\in B_1} c_i = C$ ，同理 $\sum_{c_i\in B_2} c_i = C$ 。

明顯存在 partition，令 $P=\{a_i | a_i=c_i, c_i\in B_1\}$， $\overline{P}=\{a_i | a_i=c_i, c_i\in B_2\}$

滿足 $\sum_{a_i\in P} a_i = \sum_{a_i\notin P} a_i$ 。

因此可證得 partition $\propto$ bin packing。

**7.**

根據題目說明可得知在 cell 裡的數字是由它下面一層的兩個數字加總而成

For example:

根據圖一以 255 爲例:

255=L+R (1)

L=54+m (2)

R=67+m (3)

由(1)+(2)+(3)可得

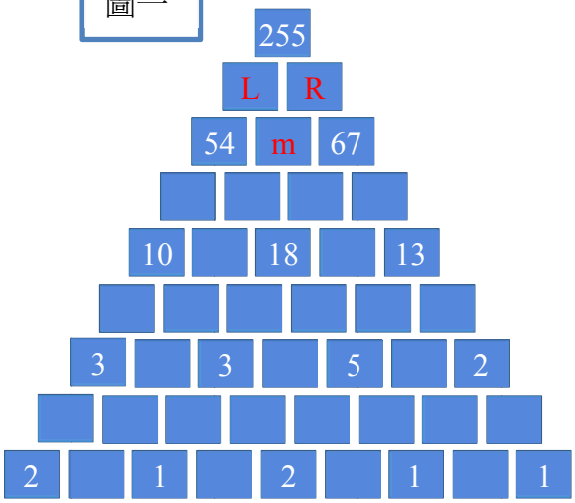
255=54+67+2*m

⇨ m=67

⇨ L=121

⇨ R=134

圖一



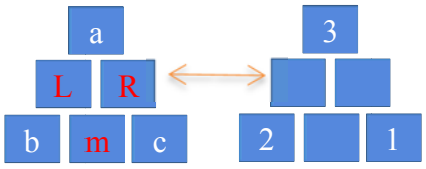

(1)由上例可推得，每次取三個已填數字的 cell 可以得出他們所涵蓋的三個空 cell 中的數字，如下圖所表示:

已知 a = 3, b = 2, c = 1

m=( a – b – c )/2 = 0

L=b + m = 2

R=c + m = 1




(2)最後由下而上，由左至右，根據(1)所述的方法即可將所有空 cell 的數字求得: