Design and Analysis of Algorithms
Final Exam., Jan. 15, 2013 參考解答

1.

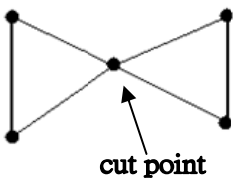
(a) NP: the class of decision problem which can be solved by a non-deterministic polynomial algorithm.
NP-complete: the class of problems to which are NP-hard and belong to NP.

(b) input: A, B
output: S
S 是sequence A的subsequence同時也是sequence B的subsequence，S 是 A 和 B 最長的 common subsequence; subsequence of A 是指 A 刪除 0 個或多個元素後所組成的 sequence.
Example: A = “cabbac”, B = “caabbc”
則“cabbc”、“caac”皆爲 A 與 B 的 common subsequence
又“cabbc”的長度爲 A 與 B 中所有的 common subsequence 中長度最長，因此 A 與 B 的 LCS 爲“cabbc”

(c) 有 n 個元素a_1, a_2, a_3,…, a_n並有各自的大小，每個箱子的容量 C，而 bin packing problem 就是找最少的箱子數將所有元素放入箱子.

(d) To reduce the time required for solving a problem, we can relax the problem, and obtain a feasible solution “close” to an optimal solution.

(e) bi-connected graph 是指一個 graph 中不存在 cut point，換言之，任 2 點之間會存在 2 條(以上)node-disjoint path.

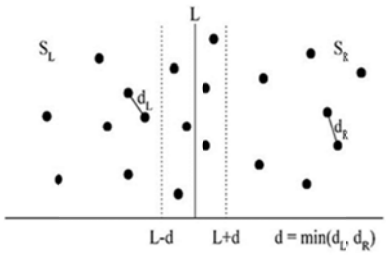


2.

I. 依 x 座標值排序座標點集合 S

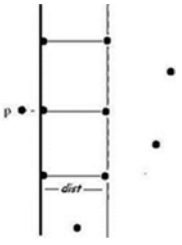
II. 若 S 只包含一座標點則 return 距離爲無限大

III. 找到一垂直線$x = x_{mid}$即右圖中直線 L，將 S 分成兩個大小一樣的子集S_L, S_R



IV. 遞迴呼叫步驟 II, III 的動作，得到左邊與右邊內的最小距離d_{Lmin}和d_{Rmin}

V. 找到分界線左右兩邊界$x = x_{mid} \pm \min\{d_{Lmin}, d_{Rmin}\}$內的點之最小距離$d_{LRmin}$，其中比對的兩點不可在分界線的同一邊。對於任意左邊的點 p，我們只需要考慮右邊矩形內的點，由於座標點之間最大距離不超過d_{Rmin}，因此矩形內最多六個點



VI. Return $\min\{d_{Lmin}, d_{Rmin}, d_{LRmin}\}$

時間複雜度：$T(n) = 2T(n/2) + O(n) = O(n \log n)$

3.

(a) Breadth-first search：tree 展開的時候，一層展開完畢後，再展開下一層。換言之，在展開 node x 的時候，會先將與 x 同一層的所有 node 均展開後，才會進入下一層。

Depth-first search：tree 展開的時候，一個 subtree 展開完畢後，再展開下一個 subtree。換言之，在展開 node x 的時候，會 recursively 先將以 x 爲 root 的 subtree 展開完畢後(一直展開至 leaf node 後，才會返回)，才會進入 x 的 brother；若 x 已無 brother，則回到 x 的 father。

Best-first search：tree 展開的時候，檢查目前所有已存活的 node，但其 son 尚未展開者，選擇評估值最佳者，爲下一個欲展開的 node。

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。

(b) Breadth-first search 使用 queue 以便記錄拜訪的順序。

Depth-first search 使用 stack 來管理

4.

Input: A set $S = \{a_1, a_2, \ldots, a_n\}$ of n elements.

Output: The k-th smallest element of S.

I. Divide S into $\lceil n/5 \rceil$ subsets. Each subset contains five elements. Add some dummy ∞ elements to the last subset if n is not a net multiple of S. $\Longrightarrow O(n)$

I. Sort each subset of elements. $\Longrightarrow O(n)$

II. Recursively, find the element p which is the median of the medians of the $\lceil n/5 \rceil$ subsets.. $\Longrightarrow T(n/5)$

III. Partition S into $S_1 = \{a_i | a_i < p,\ 1 \le i \le n\}$, $S_2 = \{a_i \mid a_i = p,\ 1 \le i \le n\}$ and $S_3 = \{a_i | a_i > p,\ 1 \le i \le n\}$, which contain the elements less than, equal to, and greater than p, respectively. $\Longrightarrow O(n)$

IV. If $|S_1| \ge k$, then discard S_2 and S_3 and solve the problem that selects the k-th smallest element from S_1 during the next iteration;
else if $|S_1| + |S_2| \ge k$ then p is the k-th smallest element of S;
otherwise, let $k' = k - (|S_1| + |S_2|)$, solve the problem that selects the k'-th smallest element from S_3 during the next iteration.
$\Longrightarrow T(3n/4)$

Time complexity：$T(n) = T(3n/4) + T(n/5) + O(n)$

Let $T(n) = a_0 + a_1 n + a_2 n^2 + \cdots,\ a_1 \neq 0$

$T(3n/4) = a_0 + (3/4)a_1 n + (3/4)^2 a_2 n^2 + \cdots$

$T(n/5) = a_0 + (1/5)a_1 n + (1/5)^2 a_2 n^2 + \cdots$

$T(3n/4 + n/5) = T(19n/20) = a_0 + (19/20)a_1 n + (19/20)^2 a_2 n^2 + \cdots$

$T(3n/4) + T(n/5) \leq a_0 + T(19n/20)$

$\Longrightarrow T(n) \leq cn + T(19n/20)$

$\leq cn + (19/20)cn + T((19/20)^2 n)$

$\vdots$

$\leq cn + (19/20)cn + (19/20)^2 cn + \cdots + (19/20)^p cn +$
$T((19/20)^{p+1} n),\ (19/20)^{p+1} n \leq 1 \leq (19/20)^p n$

$= \dfrac{1 - \left(\frac{19}{20}\right)^{p+1}}{1 - \frac{19}{20}} cn + b$

$\leq 20cn + b$

$= O(n)$

5.

Instance of exact cover decision problem：

exact cover decision problem 是指若 F 集合有 k 個子集合$F = \{s_1, s_2, s_3, \ldots, s_k\}$，每個子集合$s_i$皆由$u_1, u_2, u_3 \ldots, u_n$的元素中挑選 x 個所組成$(1 \leq x \leq n)$，而此問題是問，是否有$s_i$的聯集使得$\bigcup_{(s_i \in F)} s_i = \{u_1, u_2, u_3, \ldots, u_n\}$

Instance of sum of subset decision problem：

sum of subset decision problem 是指集合 A 由 k 個數值所組成$A = \{a_1, a_2, a_3, \ldots, a_k\}$，並給予 C 值，而此問題是問，是否能從 A 中取得部分元素並使得加總剛好爲 C

而 reduce 的方式是：假設$u_i = (k+1)^i$，$(k+1)$是爲了避免進位$(2^1 + 2^1 = 2^2)$，再將所有a_i都對應至s_i內的元素所組成的數值如下

$a_i = \sum_{j=1}^{j=n} e_{ij} (k+1)^j,\ e_{ij} = 1$ if $e_j \in s_i$ else $e_{ij} = 0$

並將 C 定爲u_1到u_n的加總，$C = \sum_{j=1}^{j=n} (k+1)^j$

exact cover decision problem $\Longrightarrow$ sum of subset decision problem

若存在$\bigcup_{s_i \in F} s_i = \{u_1, u_2, u_3 \ldots, u_n\}$則可以找到$a_i \in A$且$\sum a_i = C$，因 C 是由$u_1$到$u_n$的加總.

exact cover decision problem $\Leftarrow$ sum of subset decision problem
若存在$a_i \in A$且$\sum a_i = C$則可找到$s_i \in F$且$\bigcup_{s_i \in F} s_i = \{u_1, u_2, u_3 \dots, u_n\}$，因爲$C = \sum a_i \in A = (k+1)^1 + (k+2)^2 + \cdots + (k+1)^n$ ， $u_1 = (k+1)^1$ 、$u_2 = (k+1)^2$、…、$u_n = (k+1)^n$，然而a_i是由s_i所對應過來的數值，所以滿足$\bigcup_{s_i \in F} s_i = \{u_1, u_2, u_3 \dots, u_n\}$

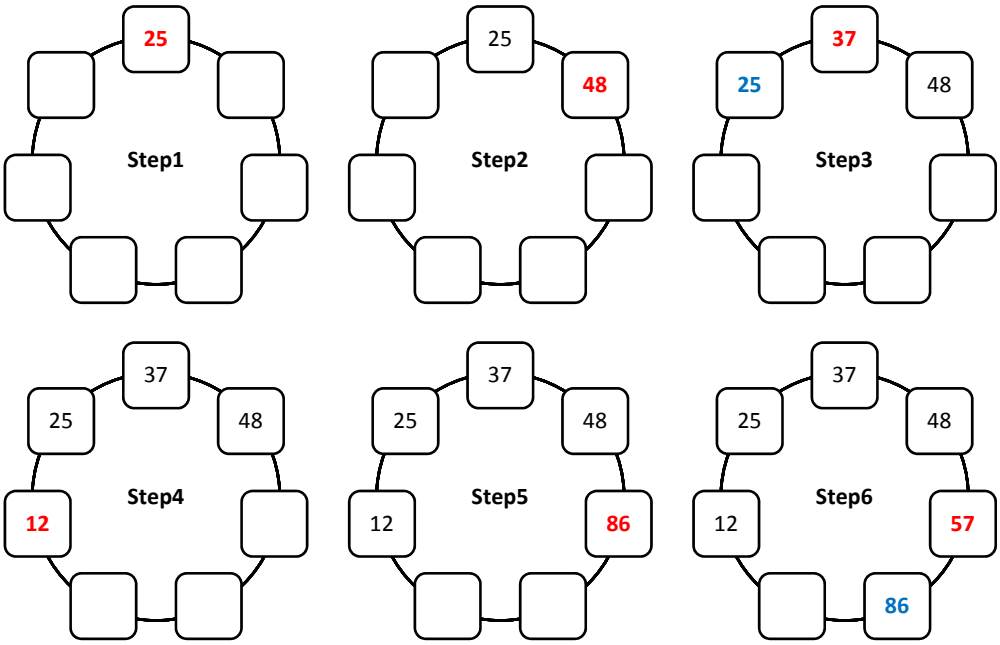
由此可證 exact cover decision problem reduce sum of subset decision problem.

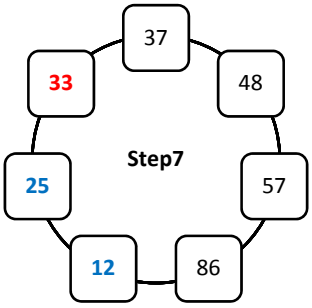
6.

(a) AVL tree 是一個高度平衡化的二元搜尋樹，對每一個 node 其左子樹的樹高與右子樹的樹高的差值，其值的絕對值不超過一。
(b) 當在輸入節點時(尚未 inserted)，若發現該節點到 root 的路徑上所有的節點的左右子樹樹高皆相同時，則輸入該點會使得新的 AVL tree 的樹高加一。
(c) 是的，因爲 AVL tree 爲一棵二元搜尋樹，故在輸入節點時與二元搜尋樹相似皆是先至於 leaf node 若要做調整再做調整，調整後未必一定在 leaf node。
(d) 不一定，在刪除節點時，可選擇任意節點刪除。

7.

Elements: 25, 48, 37, 12, 86, 57, 33
紅色：inserted；藍色：shifted。





8.

I. 依據L_i大小排序所有線段。

II. 找到$L_i \leq k < R_i$的線段(即覆蓋起始點 k 的線段)，取其中L_i最大的線段$S_a = [L_a, R_a]$ (即最靠近右邊的線段)。

III. 將R_a作爲起始點，即$k = R_a$，重複步驟 II 之動作，直到加入的線段覆蓋終點 m。

依據以上步驟由 Greedy algorithm 所加入的線段數量，即爲 minimum coverage 的數量|T|。