# Department of Computer Science and Engineering
# National Sun Yat-sen University
# Design and Analysis of Algorithms - Final Exam., Jan. 12, 2016

1. Multiple choices (There may be zero or more correct answers. If there is no correct answer, you should write down "None".) (24%)

   (a) Which statement(s) is correct? (A) If problem $A$ and problem $B$ are NP problems, then $A$ is polynomially reduces to $B$, and $B$ is polynomially reduces to $A$. (B) If problem $A$ is NP-complete and problem $B$ is P, then $B$ polynomially reduces to $A$. (C) If problem $A$ is NP-hard, then $A$ is NP-complete. (D) If any one of NP problems can be solved in polynomial time, then NP=P.

   (b) Which statement(s) is correct for the binary search? (A) The data elements must be sorted. (B) The data elements can be stored in a linked list. (C) The number of comparisons is 1 for the best case. (D) The binary search can be viewed as a prune-and-search method.

   (c) Which statement(s) is correct for the greedy method? (A) If an optimization problem can be solved by the divide-and-conquer method, then the problem can also be solved by the greedy method. (B) The shortest path problem of a graph with edges having positive weights can be solved by the greedy method. (C) The shortest path problem of a graph with edges having negative weights can be solved by the greedy method. (D) The 0/1 knapsack problem can be solved by the greedy method.

   (d) Which statement(s) is correct for the searching strategy? (A) The stack data structure should be used in depth-first search for guiding the search. (B) The stack 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 can always obtain the optimal solution.

   (e) Given a set $S$ of $n$ points on the 2D plane, which statement(s) is correct for the 1-center problem? (A) The number of farthest points from the center (solution) is at least two. (B) The center (solution) can be obtained by averaging the positions of the $n$ points. (C) If all of the $n$ points locate on the $x$-axis, then the center (solution) is equal to average $x$ value of the $n$ points. (D) For the constrained 1-center problem, it is possible that only one point is farthest from the center (solution).

(f) Which statement(s) is correct? (A) If $a$ mod $p$ = $b$ mod $p$, then $(a-b)$ mod $p$ = 0. (B) If $(a-b)$ mod $p$ = 0, then $a$=$b$. (C) ($a$ mod $p$) ( $b$ mod $p$) = ($ab$) mod $p$. (D) ($a$+$b$) mod $p$ = ($a$ mod $p$)+($b$ mod $p$).

2. Please give the definitions of (a) Voronoi polygon; (b) Voronoi diagram; (c) Delaunay triangulation. (12%)
3. Give the analysis of the time complexity of quick sort in the average case. (10%)
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%)

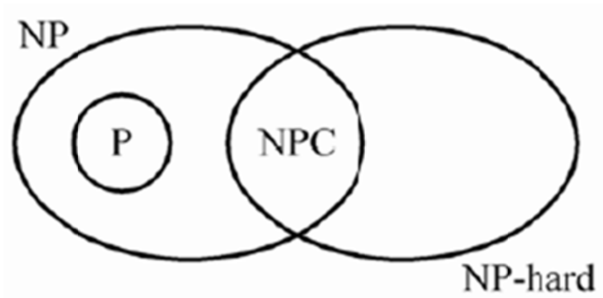
   (b) Give the dynamic programming method for calculating the LCS length. (6%)
5. In the self-organizing sequential search heuristics, what are the transpose heuristics, move-to-front heuristics and count heuristics? (12%)
6. There are two main steps for melding two skew heaps. Please present them. (6%)
7. Prove that the clique decision problem polynomially reduces to the node cover decision problem. (12%)
8. We are given a set $T$={$t_1$, $t_2$, $t_3$, …, $t_n$} of intervals, where each $t_i$ has a starting position $s_i$ and an ending position $e_i$. The minimal coverage problem is to find a subset $T'$ of $T$ such that the intervals of $T'$ can cover the line, ranging from 0 to $L$, $L \geq 0$, and $|T'|$ is minimized. Please design an algorithm to solve the minimal coverage problem and analyze the time complexity of your algorithm, Note that your algorithm should be in polynomial time. (12%)

# Design and Analysis of Algorithms
# Final Exam., Jan. 12, 2016 參考解答

1. B, ACD, B, AC, AD, A

## 1.

**(a) Ans：B**




(A) A 與 B 若同時為 NP-complete(2 者也是 NP problem)，才可以互相 reduces。

(B) 任何 NP problem 都能夠 reduce to NP-Hard，因為 P 問題也是 NP problem(見上圖)，所以能夠 reduce to NP-Hard。而 A 是 NP-complete，故 A 是 NP problem，也是 NP-hard。因此，*B* polynomially reduces to *A*.

(C) NP-complete 是 NP-hard 與 NP 的交集。例如，Halting problem(停機問題)是 NP-hard，但不是 NP 問題(undecidable)，故 halting problem 不是 NP-complete 的定義。但選項少給了一個 NP 條件，因此不能選 C。

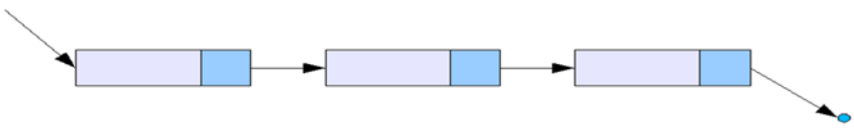
(D)若改為 If any one of NP-complete problems can be solved in polynomial time, then NP=P，此選項就會正確。

**(b) Ans：A C D**

(A) 進行 Binary search 的資料，事先必須已經 Sort。

(B) Linked list 無法直接去存取某個 Index 的資料內容，而是必須從頭(head)一個一個的去檢查每個 node 之資料，不符合 Binary search 的方法。

Linked list 需一個一個找下一個



(C) Binary search 的第一步驟為直接找尋中間的值，然後比較該值與所要搜尋值的大小。如

果中間值為即為要搜尋答案，一次比較就可得到答案，此 case 即為 best case。

(D) Binary search 每次比較後都會切除一半的資料，再去 Search，與 pruned-and-search 的方法吻合。

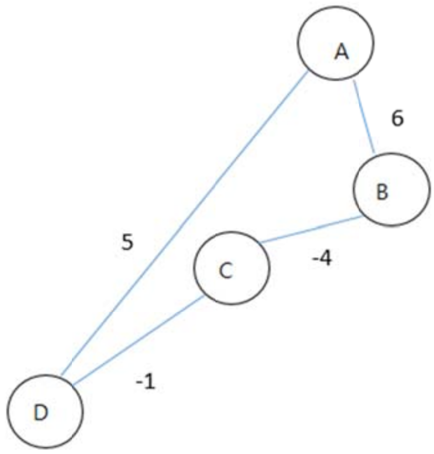
**(c) Ans：B**

(A) divide and conquer 以及 greedy 只是解決問題的技巧，但是在解一個 problem 時，前者能夠解決並不代表後者也一定可以得到正確的答案

(B) 每個邊的權重皆為正整數，即可使用 Dijkstra's algorithm 去解決 shortest path 的問題，Dijkstra's algorithm 是 greedy method。

(C) 含有負數值的邊，會導致無法用 greedy method (Dijkstra's algorithm)去找到最短路徑解，因為目前所記錄的最短路徑，未來更新時可能會得到更短的路徑。

Example：



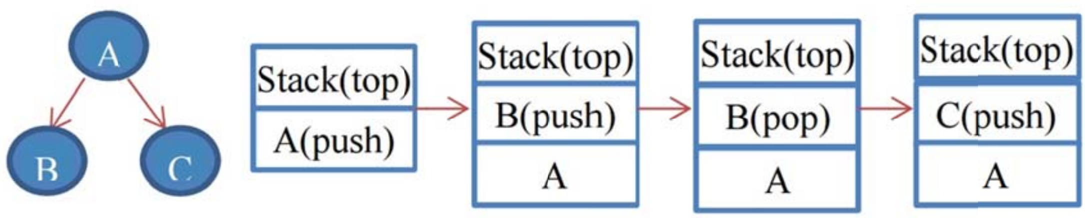

依 Greedy method 找到其最佳解為：5(A-D)，但最佳解應為：1(A-B-C-D)。

(D) 0/1knapsack problem 是 NP-Hard 的問題，greedy method 沒辦法解決 NP-Hard 的問題。但，knapsack problem 可以用 greedy method 來解決。

**(d)Ans：A C**

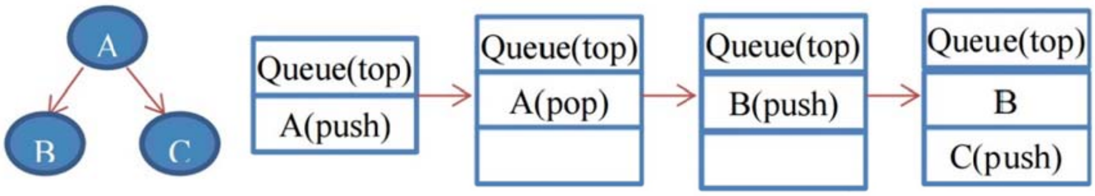
(A) DFS(depth-first search)：從某一節點(node)為出發點，不斷前進拜訪未被拜訪的結點，直到無路可走(leaf node)或是所有相鄰節點(brother node)都已被拜訪為止，然後退回前一節點(father node)，尋找尚未被拜訪的節點，直到所有相鄰節點都被拜訪為止。因為 DFS 須退回前一節點，所以必須記錄經過的所有節點；每次取出都是前一個拜訪的節點(father node)，所以 DFS 適合使用 stack。

Example：從 A 開始 DFS



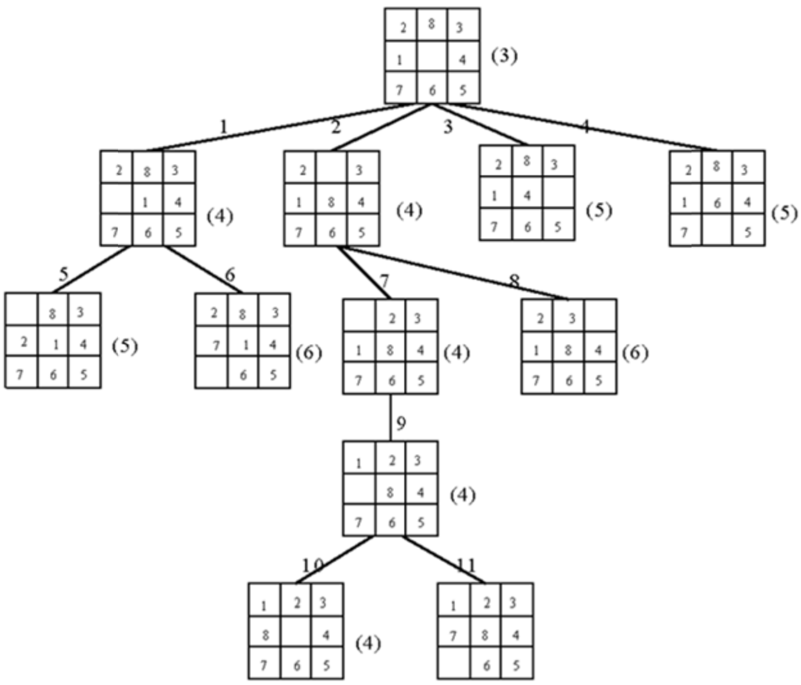
(B) BFS(breadth-first search)：拜訪節點時，會先拜訪完同一層所有節點之後，再拜訪下一層節點。因為 BFS 為一層一層拜訪，所以 BFS 適合使用 queue。

Example：從 A 開始 BFS



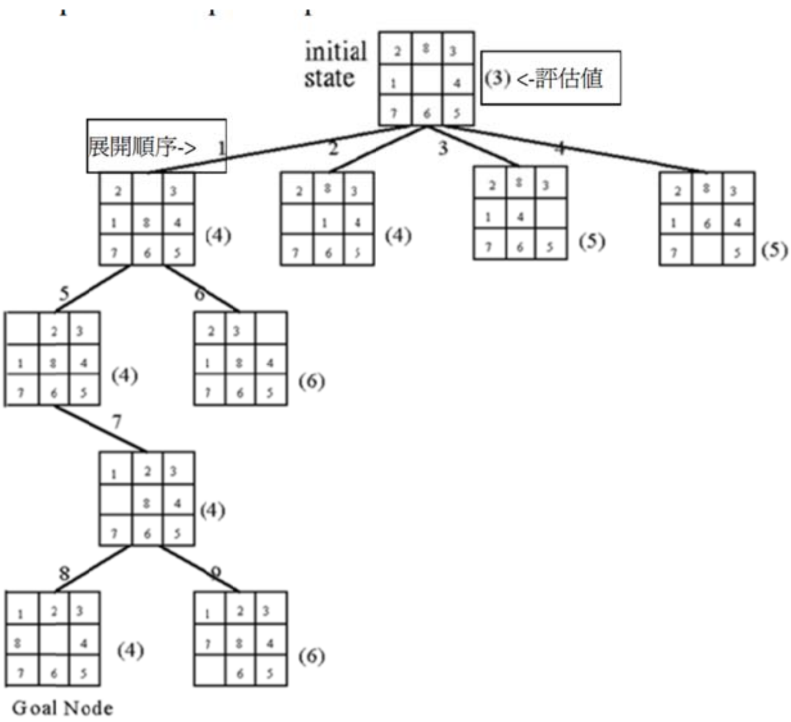

(C) Best-first search：拜訪節點時檢查目前展開的所有節點，但其 son 尚未展開者，選擇評估值最佳者為下一個欲展開之節點。在此過程中，需要尋找最小評估值、拜訪該節點後刪除該節點的資訊、新的展開點需插入資料結構。這些操作適合使用 priority queue。

Example：　An 8-puzzle problem



(D) Hill climbing：為 DFS 變形，展開節點後先給予每個子節點一個評估值，評估值為最佳者為下一個展開之節點。但是，Hill climbing 最後所得到的答案，可能不是最好的。

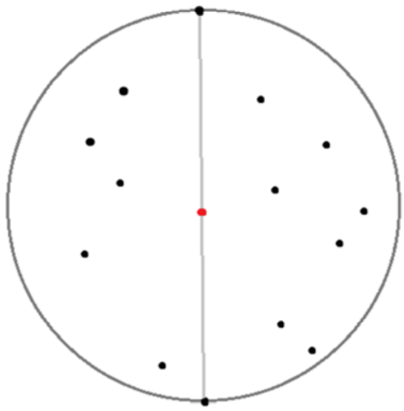
Example：An 8-puzzle problem




**(e) Ans：A D**

(A) 1-center problem 至少有 2 點會在解所形成的圓周上，因求出的解答為此圓的圓心，而此兩點即可決定圓的直徑。另一種情形，如果是三個點在圓上，則這三點所構成的三角形為銳角三角形。

Example：

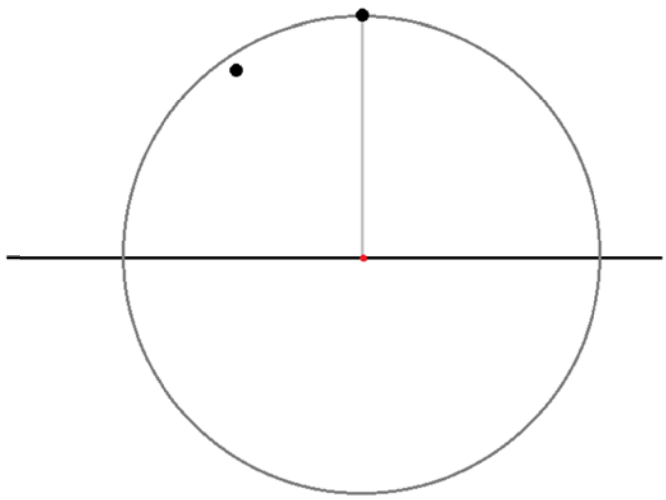


(B) 1-center problem，以所有點的平均值，做為圓心當作解答，經常是不正確。反例：有三點分 別是(2, 0)、 (3, 0)、(10, 0)，若直接求所有點的平均值則圓心為((2 + 3 + 10)/3, 0) = (5, 0)，但正確答案為(6, 0)。

(C) 1-center problem 中，若輸入的點都在 X 軸上，也不可以直接取平均值當作解答。反例：有三點分別是(2, 0)、(5, 0)、(10, 0)，其平均值為(5, 0)，但正 確答案為(6, 0)。

(D) 在 constrained 1-center problem 中，有可能只有一點在圓周上。

Example：



**(f) Ans：A**

(A) 令 $a = kp + c$，$b = hp + c,\ \ k, h \in I$，$(a - b)\ mod\ p = (k - h)p\ mod\ p = 0$。

(B) 假設$\mathrm{a} = 8$、$\mathrm{b} = 5$、$\mathrm{p} = 3$，$(8 - 5)\mathrm{mod}\ 3 = 0$，但是 $\mathrm{a} \neq \mathrm{b}$。

(C) 假設$\mathrm{a} = 3$、$\mathrm{b} = 4$、$\mathrm{p} = 5$，$(3\ \mathrm{mod}\ 5)(4\ \mathrm{mod}\ 5) = 12 \neq (12\ \mathrm{mod}\ 5 = 2)$。正確應為 $\big((\mathrm{a\ mod\ p})(\mathrm{b\ mod\ p})\big) mod\ p = (ab) mod\ p$。

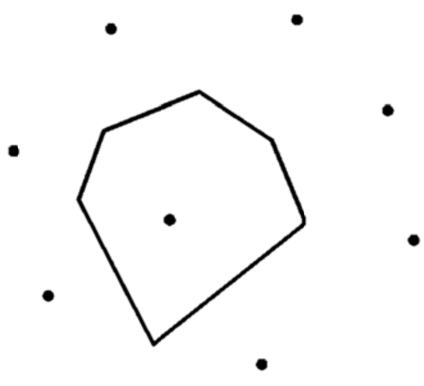
(D) 假設$\mathrm{a} = 3$、$\mathrm{b} = 4$、$\mathrm{p} = 5$，$(3 + 4)\mathrm{mod}\ 5 = 2 \neq (3\ \mathrm{mod}\ 5) + (4\ \mathrm{mod}\ 5) = 7$。正確應為 $(a + b) mod\ p = \big((\mathrm{a\ mod\ p}) + (\mathrm{b\ mod\ p})\big) mod\ p$。

## 2.

(a) Voronoi polygon：

在平面上，假設有多個點，以中間那個點為中心，跟其他點分別作中垂線形成各自的半平面，每個半平面只選擇朝向中心點的半平面，這些半平面的交集所形成的多邊形就是 Voronoi polygon。

$$V(i) = \bigcap_{i \neq j} H(P_i, P_j)$$

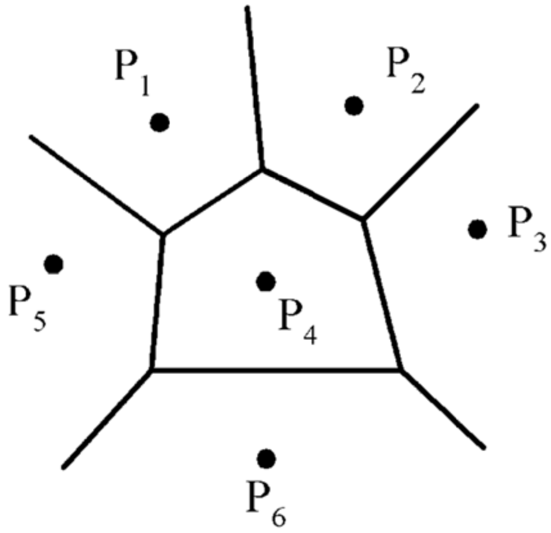


(b)Voronoi diagram

在平面上有許多點(如下圖)，以中間的點為中心對其他點作中垂線形成半平面，求出各點的Voronoi polygon，中心點的Voronoi polygon為封閉式的，其他點的Voronoi polygon為廣義的多邊形即開放式的，最後這些Voronoi polygon所組成的圖就是Voronoi diagram。

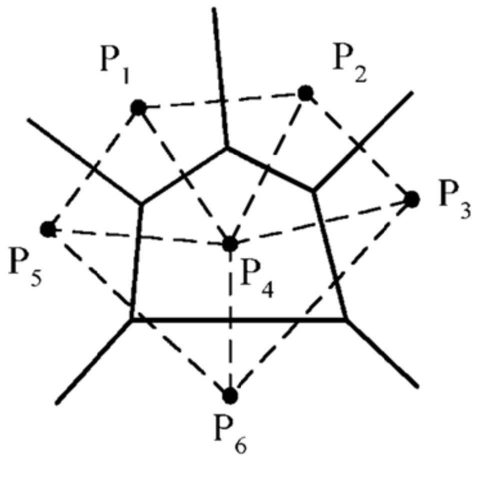
Voronoi diagram上的點為Voronoi points

Voronoi diagram上的邊為Voronoi edges



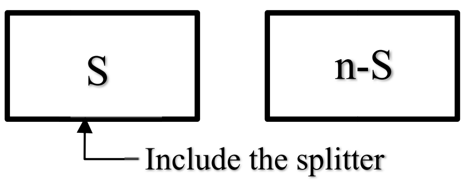

(c) Delaunay triangulation

Delaunary triangulation：對Voronoi diagram做三角化，將Voronoi diagram上的每兩個相鄰的點用虛線連在一起(如下圖)，其虛線所畫出來的圖形為三角形，也就是Delaunay triangulation （三角化）的圖形。三角形中間的Voronoi points為三角形的外心。




**3.**

Average time complexity: O(nlogn)



$T(n) = \underset{1 \le s \le n}{\text{Avg}}(T(s) + T(n-s)) + cn$ , c is a constant

$= \frac{1}{n}\sum_{s=1}^{n}(T(s) + T(n-s)) + cn$

$= \frac{1}{n}(T(1)+T(n-1)+T(2)+T(n-2)+\ldots+T(n)+T(0))+cn$ , $T(0)=0$

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

$(n-1)T(n) = 2T(1)+2T(2)+\ldots+2T(n-1) + cn^2$……(1)

$(n-2)T(n\text{-}1)=2T(1)+2T(2)+\ldots+2T(n-2)+c(n-1)^2$…(2)

(1) – (2)

$\Rightarrow(n-1)T(n) -(n-2)T(n-1) = 2T(n-1)+c(2n-1)$

$\Rightarrow(n-1)T(n) -nT(n-1) = c(2n-1)$

$\frac{T(n)}{n} = \frac{T(n-1)}{n-1} + c(\frac{1}{n} + \frac{1}{n-1})$

$=c(\frac{1}{n} + \frac{1}{n-1})+c(\frac{1}{n-1} + \frac{1}{n-2})+\ldots+c(\frac{1}{2} + 1)+T(1)$, $T(1) = 0$

$=c(\frac{1}{n} + \frac{1}{n-1}+\ldots+\frac{1}{2})+c(\frac{1}{n-1} + \frac{1}{n-2}+\ldots+1)$

$=c(H_n-1) + cH_{n-1}$ ,H is Harmonic number

$= c(2H_n-\frac{1}{n}-1)$

$\Rightarrow T(n) = 2\,cnH_n-c(n+1)$

$=O(n\log n)$

**4.**

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

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

Example: sequence A= a c b a b c

則acabc(去掉第3個字母)、acac(去掉兩個b)，皆為A 的subcequence

Subsequence of B：亦如上。

(2) Common subsequence of A and B：當一個A的subsequence與B的Subsequence相同時，則這個

subsequence稱為A和B的Common subsequence

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

則ac、aca、acc、cac 皆為A和B的Common subsequence

(3) longest common subsequence (LCS) of A and B: 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= a c b a b c and B = a c d a c c

LCS = acac

(b)

設 $A = a_1\ a_2 \ldots a_m$ and $B = b_1\ b_2 \ldots b_n$

$L_{i,j}$ ：表示 A、B 兩字串的 LCS 長度.

$$L_{i,j} = \begin{cases} L_{i-1,j-1} + 1 & ,\text{if } a_i = b_j \\ \max\{ L_{i-1,j}, L_{i,j-1} \} & ,\text{if } a_i \neq b_j \\ L_{0,0} = L_{0,j} = L_{i,0} = 0 & \text{for } 1 \leq i \leq m\ ,\ 1 \leq j \leq n \end{cases}$$

Example：Compute LCS by dynamic-programming

A = a c b a b c

B = a c d a c c

LCS = acac (長度=4)

| | ∅ | a | c | d | a | c | c |
|---|---|---|---|---|---|---|---|
| ∅ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| c | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| b | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| a | 0 | 1 | 2 | 2 | 3 | 3 | 3 |
| b | 0 | 1 | 2 | 2 | 3 | 3 | 3 |
| c | 0 | 1 | 2 | 2 | 3 | 4 | 4 |

## 5.

**(1)** Transpose Heuristics：當新字元加入後，會從最後一個位置跟前一個字元做交換的動作，若字元已經在 Sequence 中，則由目前位置與前一個字元進行對調。

範例：

| Query | Sequence |
|---|---|
| B | B |
| D | DB |

| | |
|---|---|
| A | DAB |
| D | DAB |
| D | DAB |
| C | DACB |
| A | ADCB |

**(2)** Move-to-the-Front Heuristics：當新字元加入後，會將字元擺到Sequence 最前端。若字元已經在 Sequence 中，則由目前的位置拿出，直接擺到Sequence 最前端。

範例：

| Query | Sequence |
|---|---|
| B | B |
| D | DB |
| A | ADB |
| D | DAB |
| D | DAB |
| C | CDAB |
| A | ACDB |

**(3)** Count Heuristics：計算每個字元出現的次數，出現次數較多者，會排比較前面。

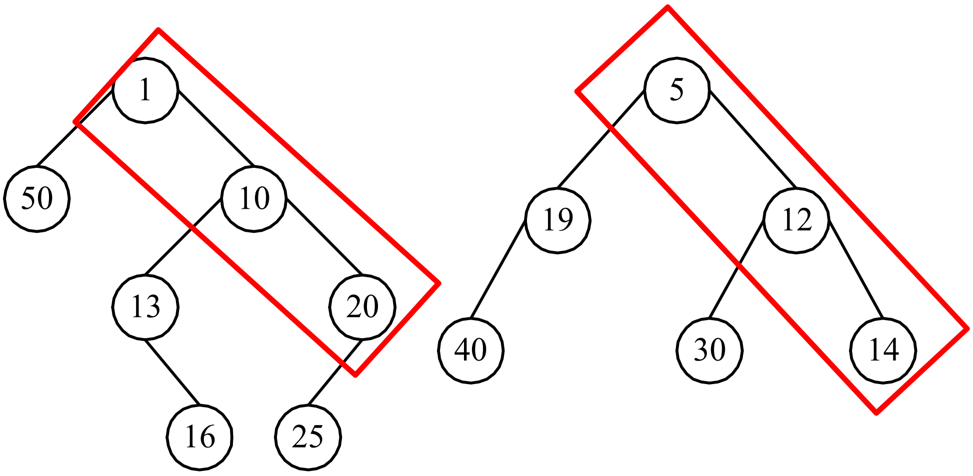
範例：

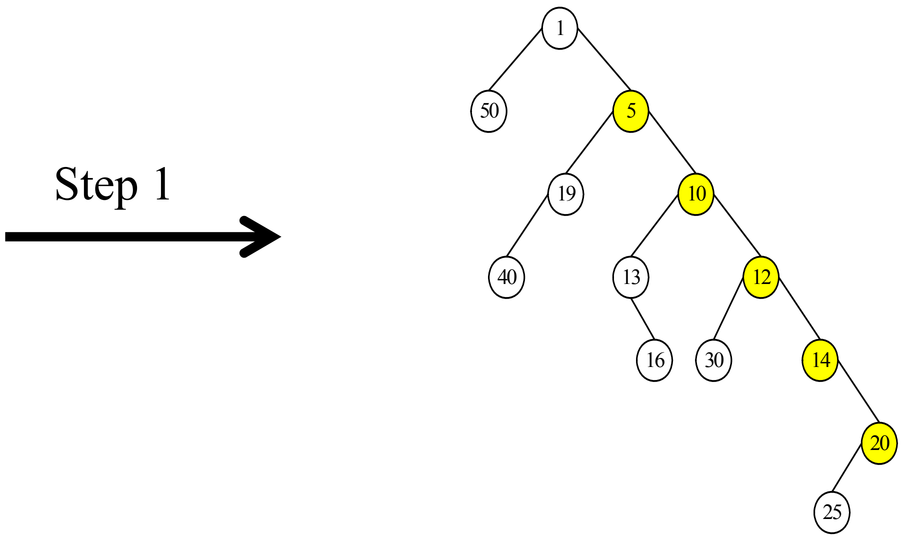
| Query | Sequence |
|---|---|
| B | B |
| D | BD |
| A | BDA |
| D | DBA |
| D | DBA |
| C | DBAC |
| A | DABC |

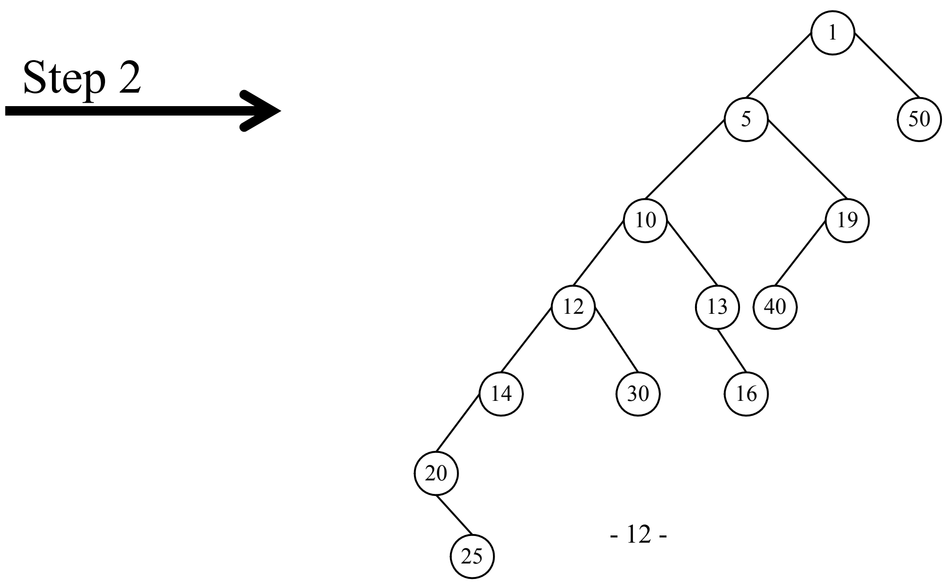
**6.**

Step 1：<u>合併(Merge)</u>右邊的路徑(Path)(即 root 和其右子樹的右子點，一直到右子點為空)。
Step 2：<u>交換(Swap)</u>右邊路徑上的左、右子點。

Example :



**7.**

Clique: 一個圖形 G=(V,E)的子圖 Q⊆V，而 Q 是完全子圖(Complete subgraph)。

Node cover set:一個圖形 G’=(V,E’)，N 為⊆V 節點之子集合(N⊆V)，而圖形的每條邊都至少有一個節點在 N 中。換言之，N 能涵蓋圖形上所有的邊。

轉換方式：

Instance of clique problem:

令 undirected graph G=(V,E)，|V|=n。假設 G 中有一個 clique Q⊆V 且 |Q|=k。

Instance of node cover problem:

令 G’為 G 的補圖，G’=(V, E’)，其中 E’={(u, v)| u∈V, v∈V and (u, v)∉E}，即 G’中的邊都不在 G 上，其中 node cover 為 V-Q，其|V-Q|=n-k。

(Clique→ Node cover)

Claim: 如果 G 有一個 Clique Q，則 V-Q 是 G’的一個 node cover set

Proof:

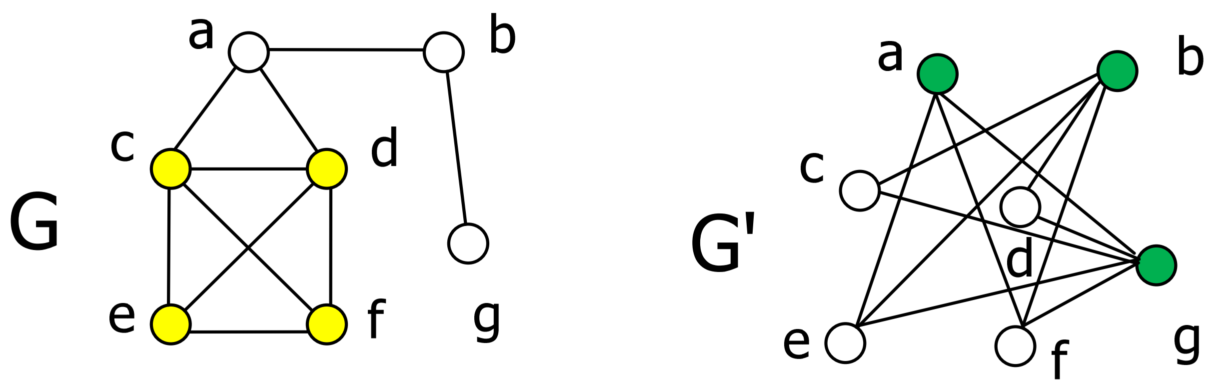
假設(u, v)為 G’中的一個邊，則(u,v)不是 G 的一個邊。由於 Q 是完全子圖，故 u 和 v 中至少有一個點不 Q 中。所以 u 和 v 至少有一個點存在於 V-Q 中，使得 V-Q 一定可以涵蓋(u,v)，故在 G’中，V-Q 是一個 node cover set。因此，如果在 G 中有一個 clique Q，則 V-Q 是 G’的一個 node cover set。

(Clique ← Node cover)

Claim: 如果 G’有一個 node cover set S⊆V, |S|=n-k，則 V-S 為 G 的 k-clique

Proof :

對於所有的 u, v∈V，如果(u, v)∈G’則 u∈S 或 v∈S 或兩者皆是。如果 u∉S 且 v∉S，則(u, v)∈E，因此所有在 V-S 上的任兩點都存在一條邊連接，故 V-S 是一個完全子圖(Complete Subgraph)，且大小為|V-S|=k，即為 k-clique。因此，如果 G’有一個 node cover set S⊆V, |S|=n-k，則 V-S 為 G 的 k-clique。



## 8.

$T = \{t_1, t_2, \ldots, t_n\}$
令 $t_i = [s_i, e_i]$
si = starting position
ei= ending position

根據題意，我們要找出可以cover整個line的最小集合T’ (T’ ⊆T)。從線段line的最左邊開始，必須找到一個涵蓋起點0(line的範圍是0~L)，且其ending position $e_i$是最大的$t_i$。將此$t_i$加入T’。加入後。整個問題變成：找出可以cover 線段line($e_i$~L)之最小子集合，利用同樣的方法，找出$t_j= [s_j, e_j]$，其中 $s_j \le e_i$。不斷地重複，直到整個line被cover。

利用Greedy method，演算法如下：

Step 0： 令 P = 0，其中P代表需要被cover的左端點，也就是需要被cover的範圍為[P,L]。

Step1：scan T，找到包含起P的 $t_i = [s_i, e_i]$ ($P \leqq s_i$)，而其$e_i$是最大的。將$t_i$加入T’，設定P= $e_i$。也就是需要被cover的範圍為改為[P=$e_i$, L]。

Step2：

需要被cover的範圍為[P, L]。重複步驟1，直到加入的t覆蓋終點L。

依據以上的步驟，所加入的線段數量，即為minimum coverage。

時間複雜度分析：

在Step1中需O(n)時間，scan 整個T找到$t_i$。Step2重複n次，不斷的scan T，因此Time complexity：$O(n^2)$。