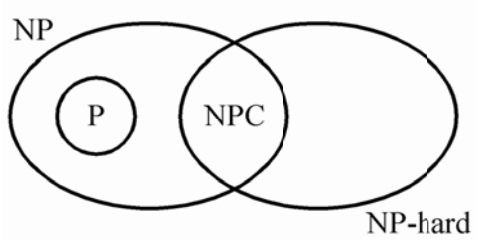
Design and Analysis of Algorithms
Final Exam., Jan. 14, 2014 參考解答

1. Multi choices

(a) Ans：A、B、D



(A) NP-complete : the class of problems which are NP-hard and belong to NP。
若 A 是 NP-hard，且 A 也是 NP，那麼 A is NP-complete。

(B) 因爲 NP-complete 爲 NP-hard 和 NP 的交集，故若 A 是 NP-complete，那麼 A 也是 NP-hard 跟 NP。

(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、B

(A) A 是 NP-complete，那麼 A 也是 NP-hard 跟 NP。而所有 NP problem 都可 reduce 到 NP-hard，也同樣可以 reduce 到 NP-complete。因此，NP problem 都可 reduce 到 A。

(B) A 與 B 是 NP-complete，那麼 A 與 B 也是 NP-hard 跟 NP。而所有 NP problem 都可 reduce 到 NP-hard 。故 A 與 B 彼此互相可以 reduce。

(C) B 不一定是 NP-complete，若要證明 B 是 NP-complete 必須先證明 B 是 NP problem，並且證明存在某個 NP-complete problem 可以 reduce 至 B。但本題 A 不是 NP-complete。

(D) 依題目敘述只能說明出 B 問題可以用 A 問題來解決，B 不一定是 NP-complete。本題若改成 then B is NP，則爲正確。

(c) Ans：B、C、D

$f(n) = \log\frac{n}{1} + \log\frac{n}{2} + \log\frac{n}{3} + \cdots + \log\frac{n}{n}$

= log n – log 1 + log n – log 2 + log n – log 3 + … + log n – log n
= nlog n – (log 1 + log 2 + log 3 + … + log n)
= nlog n – log (n!)

又 log (n!) 之 Lower Bound：

log(n!) = log(n(n – 1)…1)
= log2 + log3 +…+ log n
>$\int_1^n$ log x dx
= log e $\int_1^n$ ln x dx
= log e$[x \ln x - x]_1^n$
= log e (n ln n–n + 1)
= n log n–n log e + 1.44
≥ n log n–1.44n
=Ω (n log n)

nlog n – log (n!) < nlog n – (n log n–1.44n) = 1.44n=O(n)
所以 f(n) = O(n)。 答案不會低於 O(n)，
故答案應選 f(n) = O(n log n)、f(n) = O(n $\log^2$ n)、f(n) = O(n^2).

(d) Ans：A、C、D

(A) 0/1 knapsack problem 的元素只有取或不取，而普通的 knapsack problem，其元素是可分割的，故前者的解答不會比後者的解答大。
(B) 普通的 knapsack problem 不是 NP-hard。
(C) 0/1 knapsack problem 是 NP-hard。
(D) Decision version 的 0/1 knapsack problem 是 NP problem。

(e) Ans：B

(A) 1-center problem 至少有二點會在完成的圓的圓周上(可能只有二點)，因求出的解答爲此圓的圓心，而完成一個圓只要有圓心和半徑即可。
(B) 在 constrained 1-center problem 中，有可能只有一點在圓周上。
(C) 1-center problem 不可以直接求所有點的平均值當作解答。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爲((2 + 3 + 10)/3, 0) = (5, 0)，但正確答案爲(6, 0)。
(D) 1-center problem 中，若輸入的點都在 X 軸上，也不可以直接取中位數當作解答，反例：有三點分別是(2, 0)、(3, 0)、(10, 0)，其中位數爲(3, 0)，但正確答案爲(6, 0)。

(f) Ans：B、D

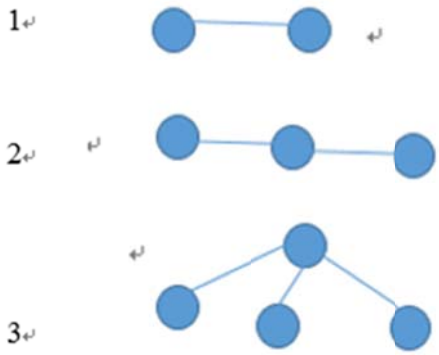
(A) Eulerian cycle 是經過圖上每個邊一次，且起點和終點一樣。

Hamiltonian cycle 是經過圖上每個點恰好都一次(亦即有些邊可能沒經過)，且起點和終點一樣。明顯地，Eulerian cycle 可能會大於 Hamiltonian cycle

例子：

Eulerian length=圖上 edge 加總 > Hamiltonian length=圖上 edge 加總

(B)

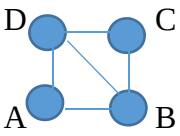


由上面的圖，可知奇數點的數量是偶數。

(C) The bottleneck traveling salesperson problem is to minimize the longest edge of a tour.

(D) bi-connected graph 是指一個 graph 中不存在 cut point，換言之，任 2 點之間存在 2 條或以上的 node-disjoint path。本題所談，任 2 點之間存在至少一條路徑，當然是正確的。

例子：



在無向bi-connected graph中刪掉任何點，任兩點nodes之間還是至少存在一條path 。

2. Ans：(9,1), (9,4), (9,7)

QR(x) = {(x,y) | y is a quadratic residue mod x}

列出所有平方的可能性如下：

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

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

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

$7^2 \equiv 4 \bmod 9$ $8^2 \equiv 1 \bmod 9$

因此得知，

(9,1), (9,4), (9,7) ∈ QR(9)

但(9,0)不屬於 QR(9)，因爲 GCD(9,0)不等於 1。

3.

令 G=(V,E)，A 代表已找到最短路徑的頂點集合，B 代表尚未找到最短路徑的頂點集合。假設 s 爲起點。

Step 1:一開始，A={s}, B=V-A，起點的距離 d(s)設爲 0，其他點距離)則設爲 s 至該點連線的長度，亦即 d(u)=w(s,u)，亦即 e(s,u)的長度，u∈B。若 e(s,u)不存在，則將 d(u)設爲無限大。

Step 2:

1. 從 B 中選擇一個與 s 距離最小的頂點 u，亦即最小的 d(u)，然後 A = A ∪ {u}, B=B-{u}。
2. 對與 u 相鄰的每個在 B 中頂點 v，進行距離更新，如果 d(u)+w(u,v) < d(v)，就把 d(v)更新成爲更短的距離 d(u)+w(u,v)

Step 3:

If B= ∅, stop; otherwise, go to Step 2.

Time complexity 分析：

Step1： 設每一頂點的距離初值需 O(n)時間

Step2 : B 中最多有 n 個頂點，需 O(n)時間找最小值，以及更新作業

Step3：迴圈作業，Step 2 與 3 合計是(n^2)

Time complexity=(n^2)，n=|V|

4.

當 Voronoi diagram 建立完成後，可用以下演算法找出 convex hull：

(1) 將 Voronoi diagram 的 edge 全部先找出來，全部檢查一遍，可以找到 Voronoi edge 是射線的情形。

(2) 找出一條 Voronoi edge 中的射線，同時可得到以此射線爲中垂線的兩點。

(3) 選出上一步的其中一點，此點必爲另一個射線的產生點，由步驟 1 可以馬上找到下一條對應的射線，也就是馬上找出產生下一個射線的點。

(4) 使用步驟 3 找到的點，重複步驟 3，一直到找出的點回到步驟 2 中的點。

(5) 步驟 3、4 所找出的點，依照被找出時的順序相連即爲所求。

時間複雜度：

1. Voronoi diagram 的 edge 最多是 3n – 6 條，全部掃一次的時間是 O(n)。
2. 可於步驟 1 時就先記錄一條，故不計時間。
3. 由步驟 1 建立的資料結構，找出下一個點只需 O(1)的時間。

4. 重複步驟 3，由於 edge 數最大爲 3n – 6，因此全部做完的時間爲 O(n)。
5. 連線得到 convex hull，同樣只需 O(n)

故時間複雜度總和爲 O(n) + O(1) + O(n) + O(n) = O(n)

5.

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 son 之後，先給予每個 son 一個評估值，以評估值最佳者爲 root 繼續 recursively 展開。

6.

(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，則由目前位置做往前交換的動作。

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，擺到 Sequence 最前端。

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。

Query	Sequence
B	B
D	BD
A	BDA
D	DBA
D	DAB
C	DABC
A	DABC

7.

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。

8.

4-subset problem 是指在一集合 S 中有 n 個正整數，S 中是否有四個相異的元素 a, b, c, d 可以滿足 a + b + c = d 的條件。

方法有如下兩種（考試時寫出第二種方法即可）：

(1) 算出所有 a + b + c 的解，再以 d 作爲 key 做 binary search。

步驟如下：

Step 1: 算出所有可能的 a + b + c 的解，也就是任取三個值，將其加總。需要時間爲 C(n,3) = O(n^3)

Step 2: 對所有 a + b + c 的 sum，進行排序，需要時間爲 O($n^3 \log n^3$) = O($3n^3 \log n$) = O($n^3 \log n$)

Step 3: 以 d 作爲 key 對 a + b + c 的 sum 進行 binary search，需要時間爲 O($n \log n^3$) = O($3n \log n$) = O($n \log n$)

時間複雜度共 O($n^3 \log$ n)

(2) 計算之前先做移項將所求調整爲 a + b = d - c，算出 a + b 與 d - c 的所有可能後，再以 d - c 爲 key 做 binary search。

步驟如下：

Step 1: 算出所有可能 a + b 的解，也就是任取二個值，將其加總。需要時間爲 C(n,2) = O(n^2)

Step 2: 對所有 a + b 的 sum，進行排序，需要時間爲 O($n^2 \log n^2$) = O($2n^2 \log n$) = O($n^2 \log n$)

Step 3: 算出所有可能 d-c 的解，也就是任取二個值，計算其差值。需要時間爲 C(n,2) = O(n^2)

Step 4: 以 d-c 作爲 key 對 a + b 的 sum 進行 binary search，需要時間爲 O($n^2 \log n^2$) = O($n^2 \log n$)

時間複雜度共 O($n^2 + n^2 \log n$) = O($n^2 \log n$)