HW13

第 3 組
104201025 張立欣
104303205 歐金榮
104303206 黃筱晴
104303542 林亦寧
105503512 趙德昊
105503516 游秉中
106503014 張秉洋

1. Exercises 34.1-5 Show that if an algorithm makes at most a constant number of calls to polynomial- time subroutines and performs an additional amount of work that also takes polynomial time, then it runs in polynomial time. Also show that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm.
   題目要問的是，假設有一個 polynomial time 的 subroutine 對他呼叫常數次，整體複雜度還是 polynomial，如過對他呼叫 polynomial 等級的次數，則整體複雜度為 exponential。
   可以假設有一個 subroutine square(a)為 n bit 的整數 a 做 a*a，時間複雜度為 O(n^2)，則有以下的程式碼。
   ```
   for i = 0 to k
   a=square(a)
   ```
   上述的程式碼每次完成一次，輸入大小就從 n 變成 2n，變大兩倍，因此最後一次輸入為(2^k)*n 時間複雜度為 O((4^k)*n^2)。
   當 k 為常數時，(4^k)為一常數，因此時間複雜度為 O(n^2)。
   當 k 為一多項式等級以上的數，例如 n，此時時間複雜度就是 O((4^n)*n^2)，exponential time。

2.

Show that the class P, viewed as a set of languages, is closed under union, intersection, concatenation, complement, and Kleene star. That is, if L_1, $L_2 \in P$, then $L_1 \cup L_2 \in P$, $L_1 \cap L_2 \in P$, $L_1 L_2 \in P$, $(L_1)^c \in P$, and $L_1{}^* \in P$.

Assume:
Since L_1 is in P, there exists machines M_1 that decides L_1;
Since L_2 is in P, there exists machines M_2 that decides L_2;
There' s a machine M, an input w.

$\boxed{L_1 \cup L_2 \in P \quad \text{union}}$

M(w) :
  Run M1 with input w.
  if (M_1 accepts w) then accept
  else
    Run M2 with input w.
    if (M_2 accepts w) then accept
    else reject

$\boxed{L_1 \cap L_2 \in P \quad \text{intersection}}$

M(w)：
  Run M_1 with input w.

if (M_1 accepts w) then run M_2 on w
  if (M_2 also accepts w) then accept
  else reject
else reject

$\boxed{L_1 L_2 \in P \quad \text{concatenation}}$ Concatenation 字 串 串 接 ， e.g. ‘abcdefg’ =’ abcd’ +’ efg’

input $w = a_1a_2 \cdots a_n$
M(w):
  for all i , $1 \le i \le n$
    Run M_1 with input $w_1 = a_1a_2 \cdots a_i$
    if (M_1 rejected w_1) then reject
    else run M_2 on $w_2 = a_{i+1}a_{i+2} \cdots a_n$
      if (M_2 rejected w_2) then reject
      else accept.

$\boxed{(L_1)^c \in P \quad \text{complement}}$

M(w):
  Run M_1 with input w
  if (M_1 accepts w) then reject
  else accept

$\boxed{L_1^* \in P \text{Kleene star}}$

M(w):
  if (w =ε) then accept.
  select a number m such that $1 \le m \le |w|$
  split w into m pieces such that $w = w_1w_2 \ldots w_m$
  for all i , $1 \le i \le m$
    run M1 on w_i
    if (M_1 rejected) then reject.
    else (M_1 accepted all w_i , $1 \le i \le m$) accept

3.
Exercises 34.2-3
Show that if HAM-CYCLE ∈ P, then the problem of listing the vertices of a Hamiltonian cycle, in order, is polynomial-time solvable.
Note 1: HAM-CYCLE is defined as “ Does a graph G have a Hamiltonian cycle? ”
Note 2: “ HAM-CYCLE ∈ P ” means that HAM-CYCLE is polynomial-time solvable.

```
for all node ∈ V
                              Ω (v)
    let Ev be the edges adjacency to current node
    for each pair { e1,e2 } ∈ Ev
        G’ = { V ,E – Ev ∪ (e1, e2) }          //挑選與 node 相連的兩條邊並刪除未被挑選的邊
        if G’ has a Hamiltonian cycle
            record (e1,e2)
            G = G’
            find next node which adjacency to e2
            break
    if no pairs be recorded          //如果第一次迴圈沒有找到，代表此圖沒有 Hamiltonian cycle
        return false
print all pair of record {e1,e2}
```

Worst case：

$C_2^{v-1} = \frac{(v-1)(v-2)}{2}$

$= \Omega(v^2)$

Time complexity: $\Omega(v^2) \times \Omega(v)$ → polynomial-time

4.

Pseudo code:

```
Topological-Sort(G);
Compute id[v] for each vertex v;
temp = NIL;
for each vertex v
    if id[v] == 0
        put v in Q
while Q is not empty
    remove a vertex v from Q
    if (temp != NIL) && (v is not in N(temp)) :
        return False
    output v
  temp = v
    for each vertex u in N(v)
            if --id[u] == 0
                put u in Q
return True
```

Time Complexity:
Topological Sort 為 O(V+E)，且 while loop 亦為 O(V+E)。
所以 T(n)=O(V+E)

5.

證明 P⊆ NP∩ co-NP

(1) P⊆NP

(2) 對任意 language L∈P

-->The complement of L ∈ P (The class **P** is closed under complementation.)

-->The complement of L ∈ NP (P⊆NP)

-->L ∈ co-NP (Let co-NP be the class of languages whose complement is in NP.)

由(1)和(2)推得 P⊆ NP∩ co-NP

6.

Proof：If NP 不等於 co-NP

- ⇨ L 屬於 NP 或 co-NP
- ⇨ L 不屬於 NP ∩ co-NP
- ⇨ By Q5，if L 不屬於 NP ∩ co-NP，then L 不屬於 P
- ⇨ Since L 屬於 NP 或 co-NP and L 不屬於 P
- ⇨ P 不等於 NP

7.

A language L is complete for a language class C with respect to polynomial-time reductions if $L \in C$ and $L' \leq_p L$ for all $L' \in C$. Show that $\emptyset$ and $\{0,1\}^*$ are the only languages in P that are not complete for P with respect to polynomial-time reductions.

ANS:

我們需要分別證明兩件事，分別是：

1. two languages are in P
2. two languages are in P but not complete.

1.

a. $\emptyset$ is in P

Because we can find an algorithm, like A, that rejects all the instance in $\emptyset$ in polynomial time.

b. $\{0,1\}^*$

Because we can find an algorithm, like B, that approves all the instance in $\{0,1\}^*$ in polynomial time.

2.

a. $\emptyset$ is not complete

If a language L is in P and complete, then any other language, we say L', in P should be able to reduce to L in polynomial time. Also, if $L' \leq_p L$ and an instance M is a yes-instance of L', then we must be able to find an function f(x) that reduces M into a yes-instance of L.

However, we can't find a function f(x) that sufficient the conditions above since $\emptyset$ has no yes-instance. Therefore, $\emptyset$ is not complete.

b. $\{0,1\}^*$ is not complete

According to 2-a, on the other hand, we can't find a function that reduces a no-instance of L' to be a no-instance of $\{0,1\}^*$ since $\{0,1\}^*$ has only yes-instance. Therefore, $\{0,1\}^*$ is not complete.