Data Structure Homework 1

學號 : 41247001S
姓名 : 盧昱安

Question 1 :

Use the definition of big-O to prove that $3n^2$ + $2nlog_2n^2 = O(n^2)$.
Provide appropriate constants c and $n0$. (20分)

Answer :

根據 Big-O 的定義，當 $f(x) \leq cn^2$ 則 $f(x) = O(n^2)$
故當 $3n^2$ + $2nlog_2n^2 \leq cn^2$ 時， $3n^2$ + $2nlog_2n^2 = O(n^2)$

我們只要能夠找到一個 c 滿足上述的條件，那$3n^2$ + $2nlog_2n^2$ 就會等於 $O(n^2)$

$3n^2$ + $2nlog_2n^2 \leq cn^2$

$= 3n^2$ + $2n * 2log_2n \leq cn^2$

$= 3n^2$ + $4n * log_2n \leq cn^2$

$= 3$ + $(4log_2n)/n \leq c$

由於 $(log_2n)/n \leq 1$ ，故 3 + $(4log_2n)/n \leq 7$
所以當 $c = 7$ 時， $3n^2$ + $2nlog_2n^2 \leq cn^2$
此時任何 $n0$ 只要滿足 $n \geq n0$, $(log_2n)/n \leq 1$ 皆可滿足條件

故證明了 $3n^2$ + $2nlog_2n^2 \leq cn^2$ 有成立的時候
也就是說 $3n^2$ + $2nlog_2n^2 = O(n^2)$

Question 2 :

Show that $4n^3 + 8n^2 + 2n$ = $\Omega()$.
Please find the maximum order for the big-Ω estimation.
Please also provide the values of c and n0 satisfying the definition of the big-Ω estimation.
(20分)

Answer :

根據 Big-Ω 的定義，當 $f(x) \geq c * g(x)$ 時， $f(x) = \Omega(g(x))$
由於題目所求為 **maximum order** 所以我們先假設 $g(x)$ 為 n^3
(因為 $deg(f(x)) = 3$)

$4n^3 + 8n^2 + 2n \geq cn^3$
$= 4 + 8/n + 2/n^2 \geq c$
此時若我們設 $c = 3$
$n >= 1$, $4 + 8/n + 2/n^2 \geq 3$
因此當 $c = 3$ 時 , $n0 = 1$

Question 3 :

Please determine a succinct big-Θ expression for the growth of the function $log(n^2) + n^2log(n^4) + 1000n^3 + 5000000n$.
You don't have to provide appropriate constants c1, c2, and n0 for the definition. However, please explain how to get your answer. (16分)

Answer :

根據 Big-Θ 的定義，當 $c1g(x) \geq f(x) \geq c2g(x)$ 時， $f(x) = \Theta(g(x))$

上課有提到過估計複雜度的時候可以看整個 $f(x)$ 成長最快的部分，
而 $log(n^2) + n^2log(n^4) + 1000n^3 + 5000000n$ 成長最快的一定是 $1000n^3$
因此我會設 $g(x)$ 為 n^3。
因為題目說明內寫說不必算出 $c1, c2, n0$
因此此題答案為 $\Theta(n^3)$，而原因是原式中成長最快的是 n^3

Question 4 :

Analyze and give the time complexity of the following program segments in terms of n. Please briefly explain your answer. (24分)

4-1 :

```
int value = 0;
for(int i=0;i<n;i++)
    for(int j=0;j<i;j++)
        value += 1;
```

Answer of 4-1

```
 //  code                    Freq            Total Steps
int value = 0;               1                    1
for(int i=0;i<n;i++)         n+1                  n+1
    for(int j=0;j<i;j++)     i+1                  ((1+n)*n)/2 + 1
        value += 1;          1                    ((1+n)*n)/2
```

Therefore, the time complexity = $((1 + n) * n)/2$ = $O(n^2)$

4-2 :

```
for (int i = 1; i < n; i++) {
    i *= k;
}
```

Answer of 4-2

```
// code                          Freq          Total Steps
for (int i = 1; i < n; i++) {     n                   n
    i *= k;                       n-1                 n-1
}
```

Therefore, the time complexity = $n - 1$ = $O(n)$

4-3 :

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

Answer of 4-3

```
// code                                  Freq          Total Steps
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {           (n/2) + 1       (n/2) + 1
    for (j = 2; j <= n; j = j * 2) {     log(n)          log(n)*(n/2) + 1
        k = k + n / 2;                      1            log(n)*(n/2)
    }
}
```

Therefore, the time complexity = $log(n) * (n/2)$ = $O(nlogn)$

Question 5

(20分)
(1) 為何表示一個程式時間複雜度(bigO)的 n 多項式會省略各項的常數值
(例如以 $O(n^2)$ 表示而不說是 $O(5n^2)$)?
(2) 若一個程式的執行步驟為 $5n^3 + 4n + 2$，為何通常稱此程式為 $O(n^3)$ 而不說是 $O(n^3 + n)$?

Answer

第一小題的原因是因為我們計算複雜度的時候，當 n 成長到一定的量後，係數已經幾乎不會影響整個計算過程的複雜度，例如當 n 到達幾千萬甚至更多之後。

再加上我們計算複雜度是為了大致上知道一個演算法的效率，而非準確的係數。
因此，我們在寫 Big-O Notation 的時候只會寫上他的成長速度，而不把係數列為重點。

第二小題的原因是因為 n^3 的成長速度太快了，所以當 n 改變的時候， n^3 的成長相較其他兩項差距太大，所以會省略掉成長速度較慢的項次，保留最重要也就是成長速度最快的項次。

兩小題的共通點都是因為我們在寫複雜度的時候重點是寫出他的效率，所以只保留最重要的部分。

中秋節快樂（ ╹▽╹ ）