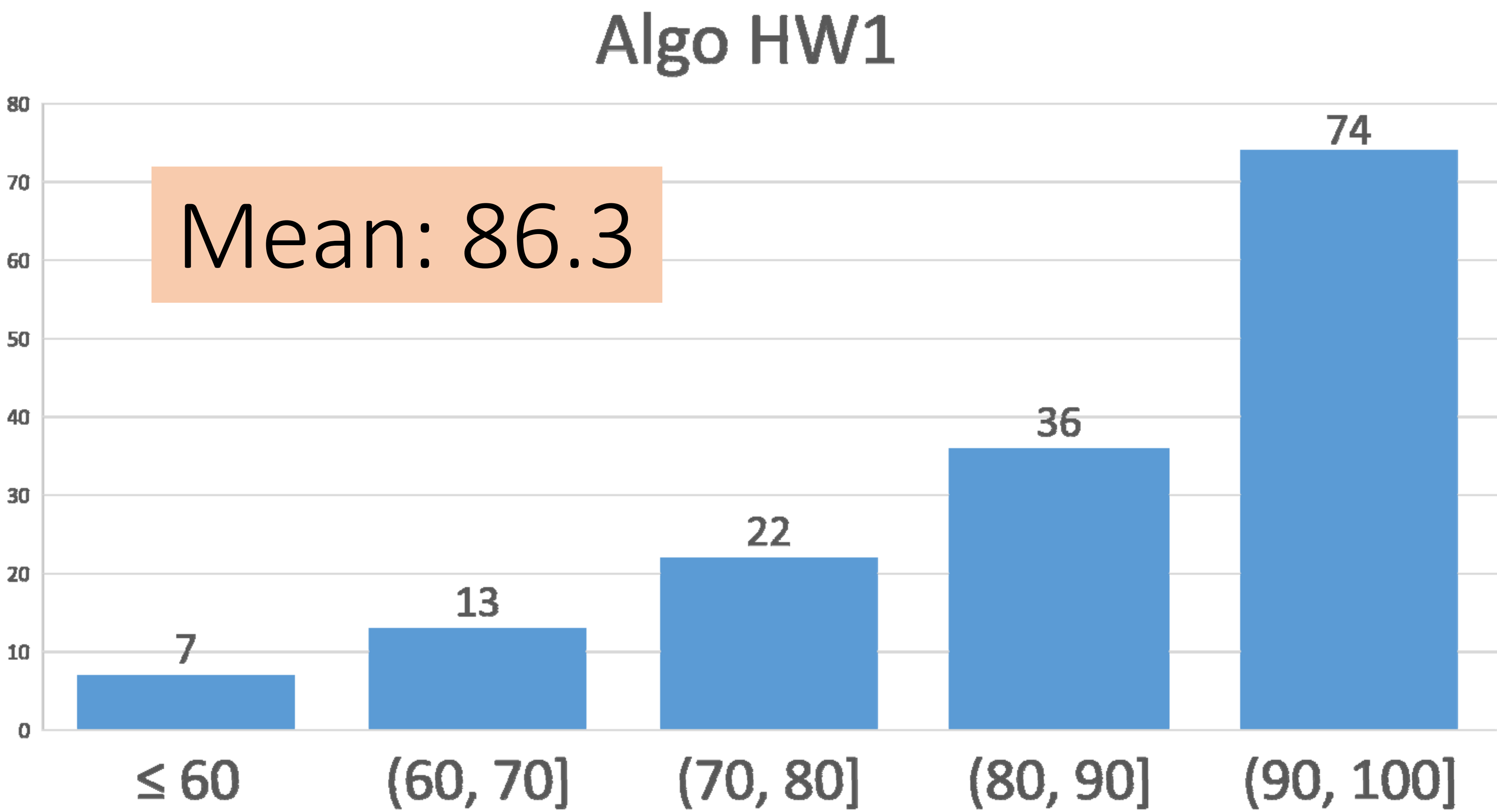
Algorithm 2019

# HW 1 Solution

教授: 謝孫源 講座教授
助教: 姚凱勛 李昀 鄭力瑋 楊奕正



Algo HW1
Mean: 86.3
80
70
60
50
40
30
20
10
0
7
13
22
36
74
≤ 60
(60, 70]
(70, 80]
(80, 90]
(90, 100]

1. (10pts) Explain why the statement, “The running time of algorithm A is at least $O(n^2)$.” is meaningless.

假設A的running time為T(A)，依題意: $T(A) \geq O(n^2)$。

判斷A的upper bound和lower bound:

- Upper bound: 因為$T(A) \geq O(n^2)$，無法確定上限。
- Lower bound: $O(n^2)$包含任何小於$n^2$的函數，故T(A)的下限為常數。

所有演算法皆符合此upper bound和lower bound，故此敘述沒有意義(沒有有用的資訊)。

The statement is a bit like saying: "The roof of my house is at a height which is at least ground level." True, but completely uninformative.

評分標準:

- 關鍵字: “Big-O的定義”(upper bound、數學型式…) & “至少”，+10分。
- 講到Big-O的定義，+5分
- 講到應該用omega或是應該用至多，+5分

2. (10pts) Come up with a real-world problem in which only the best solution will do. Then come up with one in which a solution that is “approximately” the best is good enough.

一定要best solution才可以(每次找到的答案都一樣):

- 一堆數字依大小排序
- 依索書號在圖書館找書
- 找老公

接近best solution就可以(每次找到的答案可能不一樣):

- 找圓周率: 3.1415926535 ≒ 3.14
- 用google map找兩地點間路線
- 找男朋友

評分標準:

- 缺少前提或條件設定，+9分
- Best solution和approximately best solution只寫其中一個，+5分
- 註: shortest path是最佳解(每次找到都一樣)，需加上情境或限制條件，如: 找google地圖上兩點適宜路徑、在限定時間內能找出的最短路徑。

3. (10pts) Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For input of size n, insertion sort runs in $\boldsymbol{8n^2}$ steps, while merge sort runs in $\boldsymbol{64n\lg n}$ steps. For which values of n does insertion sort beat merge sort?

$8n^2 < 64n\lg n \Longrightarrow \rightarrow n/8 < \lg n \Longrightarrow 2^{n/8} < n$

$n=8 \Longrightarrow 2^{8/8} = 2^1 = 2 < 8$ $\quad$ $n=16 \Longrightarrow 2^{16/8} = 2^2 = 4 < 16$

$n=32 \Longrightarrow 2^{32/8} = 2^4 = 16 < 32$ $\quad$ $n=64 \Longrightarrow 2^{64/8} = 2^8 = 256 > 64$

$n=(32+64)/2 = 48 \Longrightarrow 2^{48/8} = 64 > 48$

$n=(32+48)/2 = 40 \Longrightarrow 2^{40/8} = 32 < 40$

$n=(40+48)/2 = 44 \Longrightarrow 2^{44/8} = 44.8 > 44$

$n=(40+44)/2 = 42 \Longrightarrow 2^{42/8} = 34.4 < 42$

$n=(42+44)/2 = 43 \Longrightarrow 2^{43/8} = 42.4 < 43$

$\Longrightarrow 1 < n < 44$

評分標準:

- 只寫出n的不等式，+5分
- 答案的 < 寫≤，+9分

# 4. (10pts) Prove $\mathbf{lg}(\boldsymbol{n}!) = \Theta(\boldsymbol{n}\ \mathbf{lg}\boldsymbol{n})$.

- Consider big-O :
  - $\lg(n!) = \lg(n * (n-1) * (n-2) * \cdots) = \lg(n) + \lg(n-1) + \cdots + \log(1) \leq \lg(n) + \lg(n) + \cdots + \lg(n) = nlgn$

    $$\Rightarrow \lg(n!) = O(nlgn)$$
- Consider big-Ω :
  - $\lg(n!) = \lg\left(n * (n-1) * \cdots \left(\frac{n}{2}\right) * \cdots 1\right) = \lg(n) + \lg(n-1) + \cdots \lg\left(\frac{n}{2}\right) + \log(1) \geq \lg\left(\frac{n}{2}\right) + \lg\left(\frac{n}{2}\right) + \cdots + \lg\left(\frac{n}{2}\right) = \frac{n}{2} lg \frac{n}{2}$

    $$\Rightarrow \lg(n!) = \Omega(nlgn)$$
- By definition:
  - $lg(n!) = \Theta(n\ lgn)$.

5. (10pts) Partition the following functions by their asymptotic order. (That is, f and g are in the same partition if, and only if, $\boldsymbol{f} \in \Theta(\boldsymbol{g})$.) Then list them from the lowest asymptotic order to highest asymptotic order.

$$\boldsymbol{n},\ \ 2^n,\ \ n\lg n,\ \ n^3,\ \ n^2,\ \ 7n^5 - n^3 + n,\ \ n^2 + \lg n,\ \ e^n,$$
$$\sqrt{n},\ \ 2^{n-1},\ \ \lg\lg n,\ \ \lg n,\ \ \lg^2 n,\ \ n!,\ \ n^{1+\varepsilon}(0 < \varepsilon < 1)$$

- ***comparing rule:***
- ***Constant time < Logarithmic time < Polylogarithmic time < Polynomial time < Exponential time < factorial time***

- $lglgn < lgn < lg^2 n < \sqrt{n} < n < nlgn < n^{1+\varepsilon}(\mathbf{0} < \boldsymbol{\varepsilon} < \mathbf{1}) < n^2 + lgn = n^2 < n^3 < 7n^5 - n^3 + n < 2^{n-1} = 2^n < e^n < n!$

6. (10pts) Given a sequence of numbers a1, a2, a3, a4, a5, in which condition will the insertion sort has the most and the least swap to sort the sequence to ascending order?

- The worst case: descending order
- The best case: ascending order

a1,a2,a3,a4,a5 are input order!!!

7. (10pts) Answer “true” or “false” first, then explain the reason.

**(a)** $\mathbf{2^{n+2} = O(2^n)}$

**(b)** $\mathbf{2^{2n} = O\,(2^n)}$

- $\mathbf{2^{n+2} = O(2^n)}$
- $\mathbf{2^{2n} = O\,(4^n)}$
- You can use the definition of big O to explain it.

8. Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

**a) (5pts) $T(n)$ = $T(9n/10)$ + $n$ (Use Master Theorem)**

- **By master theorem,** a=1,b=10/9
- $n^{\log_{10/9} 1} = n^0, f(n) = n^1$
- $f(n) = \Omega(n^{0+\varepsilon}), \varepsilon > 1$ (case 3)
- $af\left(\frac{n}{b}\right) \leq cf(n) \Rightarrow$ 取$c = \frac{9}{10}, \quad \frac{9n}{10} \leq \frac{9n}{10}$ satisfy regularity condition
- $T(n) = \theta(n)$

8. Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

**b) (5pts) $T(n) = 4T(n/2) + n^2$ (Use Master Theorem)**

- **By master theorem**, a=4,b=2
- $n^{log_2 4} = n^2, f(n) = n^2 = \theta(n^{log_2 4} \, log^k n)$ ,k=0 (Case 2)
- $T(n) = \theta(n^{log_2 4} \, log^{0+1} n) = \theta(n^2 \; log\, n)$

**8. (20pts) Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.**

**c) (5pts) $T(n) = T(\sqrt{n}) + 1, T(2) = 1$ (Use substitution method)**

沒過程：扣分(red circle)

(method.1) $let\ 2^k = n, k = lgn, then$

$$T(n) = T(\sqrt{n}) + 1$$

$$\Rightarrow T(2^k) = T(2^{k/2}) + 1$$

$$= T(2^{k/2^2}) + 2$$

3pts

$$= T(2^{k/2^3}) + 3$$

$$\ldots\ldots$$

$$= T(2^1) + lgk$$

$$= 1 + lgk$$

$$\therefore T(n) = 1 + lgk = 1 + lglgn = \Theta(lglgn)$$ 2pts

If you only write:

$1 + lglgn = O(lglgn)$

(-1 points)

c) (5pts) $T(n) = T(\sqrt{n}) + 1, T(2) = 1$ (Use substitution method)

Wrong:

"Guess : $T(n) = lglg(n)$ "

Why?

(method.2) Guess : $T(n) = d \times lglg(n) + c$ ,

$$T(2) = d * lglg(2) + c = c,\ and\ T(2) = 1,$$

$$it's\ true\ when\ c = 1, so\ c = 1$$

$$if\ it's\ true\ for\ all\ n\ \leq k - 1, then\ when\ n = k,$$

$$\begin{aligned} T(k) &= T(\sqrt{k}) + 1 \\ &= d \times lglg(\sqrt{k}) + 1 + 1 \\ &= d \times lg\left(\frac{1}{2} lg(k)\right) + 2 \\ &= d \times (lg\frac{1}{2} + lglg(k)) + 2 \\ &= -d + d \times lg\, lgk + 2 \\ &= d \times lg\, lgk + (2 - d) \end{aligned}$$

$$it's\ true\ when\ d = 1, and\ by\ I.H.,$$

$$\therefore we\ get\ that\ T(n) = lglgn + 1 = \Theta(lglgn)$$

**8. (20pts) Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.**

**d) (5pts) $T(n) = T(n/4) + T(3n/4) + n$ (Use recursion-tree method)**

Lower upper寫反：扣分

$$n$$

$$\frac{n}{4} \qquad \frac{3n}{4}$$

$$\frac{n}{16} \qquad \frac{3n}{16} \qquad \frac{3n}{16} \qquad \frac{9n}{16}$$

$\log_4 n$ ...... ......

$T(1) = 1$ $\log_{\frac{4}{3}} n$

$T(1) = 1$

$n$

$n$

$n$ 3pts

......

$n$

$lower\ bound: \Omega(n * \log_4 n)$
$= \Omega(n\, lg\, n)$

$upper\ bound: O\left(n * \log_{\frac{4}{3}} n\right)$
$= O(n\, lg\, n)$

$= \Theta(n\, lg\, n)$ 2pts

| |
|---|
| **9. (10pts) Given the definition of Fibonacci number:** $$\mathbf{F}_n = \mathbf{F}_{n-1} + \mathbf{F}_{n-2}, \text{ for } n \geq 2$$ $$\mathbf{F}_0 = 0, \mathbf{F}_1 = 1$$ |
| **(1) Write a *Pseudocode* to compute $\mathbf{F}_n$ with *recursive* method. Please analyze your time complexity and give the answer with Asymptotic notation.** |

2pts

Function *Fib(n)*

if $n \leqq 1$

return $n$

else

return *Fib(n-1)* + *Fib(n-2)*

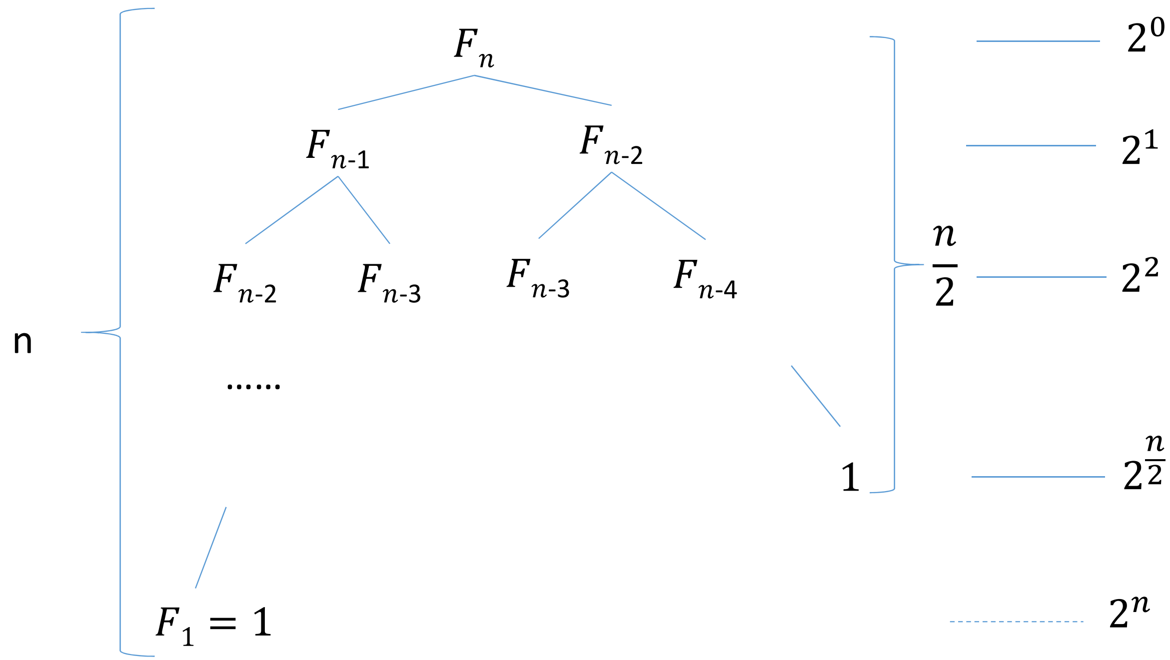
**9. (10pts) Given the definition of Fibonacci number:**

$$\mathbf{F}_n = \mathbf{F}_{n-1} + \mathbf{F}_{n-2}, \text{ for } n \geq 2$$

$$\mathbf{F}_0 = 0, \mathbf{F}_1 = 1$$

**(1) Write a *Pseudocode* to compute $\mathbf{F}_n$ with *recursive* method. Please analyze your time complexity and give the answer with Asymptotic notation.**

3pts Loose bound: use recurrence trees




$$upper\ bound: 2^0 + 2^1 + \cdots + 2^n = O(2^n)$$

$$lower\ bound: 2^0 + 2^1 + \cdots + 2^{\frac{n}{2}} = \Omega(2^n)$$

$$\therefore time = \Theta(2^n)$$

*true lower bound in this method :*

$$2^0 + 2^1 + \cdots + 2^{\frac{n}{2}} = \Omega\left(\sqrt{2}^n\right)$$

| |
|---|
| **9. (10pts) Given the definition of Fibonacci number:** <br> $$\mathbf{F}_n = \mathbf{F}_{n-1} + \mathbf{F}_{n-2}, \text{ for } n \geqq 2$$ <br> $$\mathbf{F}_0 = 0, \mathbf{F}_1 = 1$$ |
| **(1) Write a *Pseudocode* to compute $\mathbf{F}_n$ with *recursive* method. Please analyze your time complexity and give the answer with Asymptotic notation.** |

tight bound:(1)

$$let\ T(n) = a_n, then$$

$$a_n = a_{n-1} + a_{n-2} \Rightarrow a_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

$$\therefore T(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

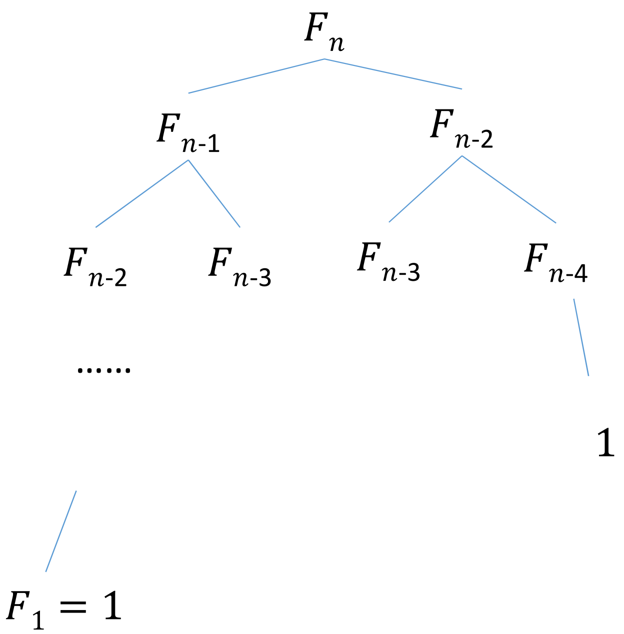
**9. (10pts) Given the definition of Fibonacci number:**

$$\mathbf{F}_n = \mathbf{F}_{n-1} + \mathbf{F}_{n-2}, \text{ for } n \geq 2$$

$$\mathbf{F}_0 = 0, \mathbf{F}_1 = 1$$

$$\begin{aligned}
time &= F_1 + F_2 + F_3 \dots\dots + F_{n-1} + F_n + F_{n-1} \\
&= F_2 + F_1 + F_2 + F_3 \dots\dots + F_{n-1} + F_n + F_{n-1} - F_2 \\
&= F_3 + F_2 + F_3 \dots\dots + F_{n-1} + F_n + F_{n-1} - F_2 \\
&= F_4 + F_3 \dots\dots + F_{n-1} + F_n + F_{n-1} - F_2 \\
&\quad \dots\dots \\
&= F_{n+1} + F_n + F_{n-1} - F_2 \\
&= F_{n+2} + F_{n-1} - F_2
\end{aligned}$$

# tight bound: (2)use recurrence trees




$F_n$: 1次(= $F_1$)
$F_{n-1}$: 1次(= $F_2$)
$F_{n-2}$: 2次(= $F_3$)
$F_{n-3}$: 3次(= $F_4$)
$F_{n-4}$: 5次(= $F_5$)
......
$F_2$: (= $F_{n-1}$)
$F_1$: (= $F_n$)
$F_0$: (= $F_{n-1}$)

(use公式解)

$$= \frac{1}{\sqrt{5}}\left( \left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+2} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1} \right) - 1$$

$$= \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n}\right)$$

9. (10pts) Given the definition of Fibonacci number:

$$F_n = F_{n-1} + F_{n-2}, \text{ for } n \geqq 2$$

$$F_0 = 0, F_1 = 1$$

(2) Write a *Pseudocode* to compute $F_n$ with *iterative (loop)* method. Please analyze your time complexity and give the answer with Asymptotic notation.

```
a ← 0
b ← 1
ans ← n
for i ← 2 to n
        ans ← a + b
        a ← b
        b ← ans
return ans
```

or

```
A[0] ← 0
A[1] ← 1
for i ← 2 to n
        A[i] ← A[i-1] + A[i-2]
return A[n]
```

2pts

2pts

(1)Explaim it is linear algo.

(2)Count (n-2)* $\Theta(1)$

(3)Count how many times each step will do

time = $\Theta(n)$

1pts

# Wrong example

- Assume $\mathrm{T(n)} = O(2^n)$

$\mathrm{T(n)} = T(n-1) + T(n-2)$

$= O(2^{n-1}) + O(2^{n-2})$

$= O(2^n)$

- You can't use your assumption of $O(2^n)$ to prove.
- You can do

Assume $\mathrm{T(n)} = O(2^n)$

$\mathrm{T(n)} = T(n-1) + T(n-2)$

$\leq d(2^{n-1}) + d(2^{n-2}) + C$

… …

# About prove

- 在證這類證明的時候，沒有約等於，沒有≒，no use of ≒
  - You should use ≧ or ≦ or $\Theta$

- You should prove what you assume.
  - Wrong e.g.” Guess :T(n)=2^n, but prove O(2^n) or ¾*2^n”

**(Bonus 10pts)：**

**Give asymptotic tight bounds for T(n) in the following recurrences.**

$$T(n) = 27T(n/3) + \Theta(n^3/\lg n)$$

因為是bonus寫一半的不給部份分 sorry~

$$let\ 3^k = n, k = \log_3 n, then$$

$$T(n) = 27T(n/3) + \Theta(n^3/lgn)$$

$$\Rightarrow T(3^k) = 3^3T(3^{k-1}) + \Theta(3^{3k}/k)$$

$$= 3^{3\times2}T(3^{k-2}) + \Theta(3^{3k}/k) + 3^3\Theta(3^{3(k-1)}/k-1)$$

$$= 3^{3\times2}T(3^{k-2}) + \Theta(3^{3k}/k) + \Theta(3^{3k}/k-1)$$

$$\ldots\ldots$$

$$= 3^{3k}T(3^{k-k}) + \Theta\left(\frac{3^{3k}}{k}\right) + \Theta\left(\frac{3^{3k}}{k-1}\right) + \Theta\left(\frac{3^{3k}}{k-2}\right) \ldots\ldots + \Theta\left(\frac{3^{3k}}{k-(k-1)}\right)$$

$$= 3^{3k}(\Theta(1)) + 3^{3k}\left(\Theta\left(\frac{1}{k}\right) + \Theta\left(\frac{1}{k-1}\right) + \Theta\left(\frac{1}{k-2}\right) \ldots\ldots + \Theta(1)\right)$$

$$= 3^{3k}(\Theta(1)) + 3^{3k}\left(\sum_{i=1}^{k} \Theta\left(\frac{1}{i}\right)\right)$$

$$= 3^{3k} + 3^{3k}(\Theta(\ln k)) = 3^{3k} + 3^{3k}(\Theta(lg\,k))$$

$$\therefore T(n) = n^3 + n^3\Theta(lglgn) = \Theta(n^3 lglgn)$$