PERFORMANCE EVALUATION

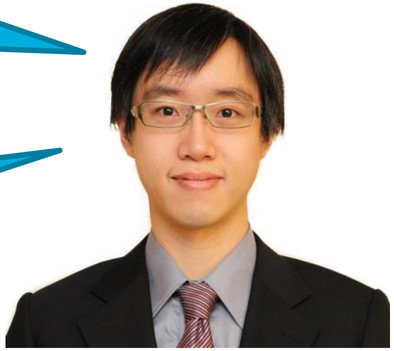
Prof. Michael Tsai

2013/2/26

Office Hour
已公布在課程網站

HW0 is due 23:59
on Thursday.

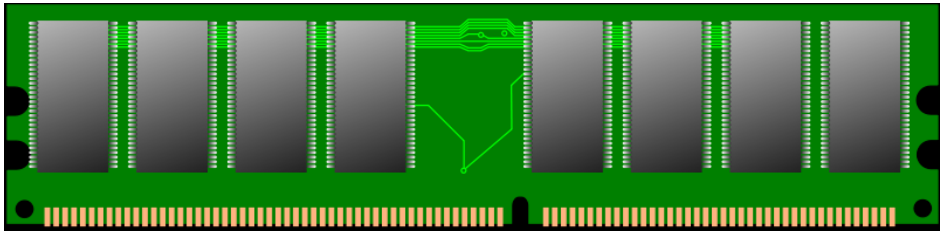
HW1 今天或明天
會公布



空間及時間複雜度

- 程式的空間複雜度:
  - 程式執行完畢所需使用的所有空間(記憶體)
- 程式的時間複雜度:
  - 程式執行完畢所需使用的(執行)時間
- <u>Goal: 找出執行時間/使用空間”如何”隨著input size變長(成長的有多快)</u>
- 什麼是input size?
- 問題給的input的”元素數量”, 如:
  - Array大小
  - 多項式最高項的次方
  - 矩陣的長寬
  - 二進位數的位元數目

空間複雜度



- 程式所需空間:
  1. 固定的空間
     - 和input/output的大小及內容無關
  2. 變動的空間
     - 和待解問題P的某個input instance I(某一個input)有關
     - 隨著input size變大而變大 ←我們需要注意的地方!
     - 跟recursive function會使用到的額外空間有關
- $S(P) = c + S_P(I)$

Example

```
float abc(float a, float b, float c) {
   return a+b+b*c+(a+b-c)/(a+b)+4.00;
}
```

- $S_{abc}(I) =?$ （變動空間）
- 只有固定空間.
- $S_{abc}(I) = 0.$

Example

```
float sum(float list[], int n) {
      float tempsum=0;
      int i;
      for(i=0;i<n;++i)
            tempsum+=list[i];
      return tempsum;
}
```

- $S_{sum}(I) =?$ （變動空間）
- $S_{sum}(I) = n$ （list陣列所占空間）

Example

```
function rsum(float list[], int n) {
    if (n) return rsum(list,n-1)+list[n-1];
    return 0;
}
```

- $S_{rsum}(I)$ =? （變動空間）
- list[]占n個sizeof(float)的大小
- 另外每個recursive call需要紀錄：
- float *list, int n, 還有return address. 假設這三樣東西需要k bytes （注意是list的指標，非整個陣列）
- $S(n) = S'(n) + sizeof(float) * n$
- $S'(n) = S'(n-1) + k$
- $S'(0) = k$
- $S'(n) = (n+1)k$

時間複雜度

- 一個程式*P*所需使用的時間*:*
  - Compile所需時間
  - 執行時間 (execution time or run time)
  - How to do it on workstation?

- Compile時間: 固定的. (例外?)
  - C (and other compiled programming languages)
    → One Compilation → Multiple Executions

- Run time:
  - 和input instance的特性有關!
  - 例如input size n - 我們可以把run time寫成T(n)

如何得到 $T(n)$?

1. 總執行時間
   - 花了多少時間? →有其他程式也在使用處理器!
   - 花了多少處理器時間? → 跟機器, 作業系統相關

2. 所執行的程式指令的數目 & 每個指令的時間

3. 比較數學的方法(使用function來代表程式執行的時間)

Is it good to use?
(方法1)

一些小假設

- 使用一個處理器(processor). 循序一次執行一個指令. (無法同時做超過一件事情)
- 常用的指令每個指令花固定時間(constant time)
  - +-*/%
  - 記憶體動作(讀取, 修改, 儲存)
  - 控制用的statement: 呼叫subroutine, branch等等

Insertion Sort

藍色: 已經排好的範圍
綠色: 目前正在處理的數字

i	1	2	3	4	5	6
A[i]	5	2	4	6	1	3

i	1	2	3	4	5	6
A[i]	2	5	4	6	1	3

i	1	2	3	4	5	6
A[i]	2	4	5	6	1	3

i	1	2	3	4	5	6
A[i]	2	4	5	6	1	3

i	1	2	3	4	5	6
A[i]	1	2	4	5	6	3

i	1	2	3	4	5	6
A[i]	1	2	3	4	5	6

Insertion-Sort(A)

```
for j=2 to A.length
    key=A[j]
    i=j-1
    while i>0 and A[i] > key
        A[i+1]=A[i]
        i=i-1
    A[i+1]=key
```

每回合選一個來做insertion

要排的數值是key

從前一個開始看

把比key大的往右邊移

最後把key填入空出來的那一格

j=6

i	1	2	3	4	5	6
A[i]	1	2	4	5	6	3

執行時間=?

n=A.length

t_i:j=i的時候那次while執行次數

時間(cost) 做了幾次

```
Insertion-Sort(A)
    for j=2 to A.length
        key=A[j]
        i=j-1
        while i>0 and A[i] > key
            A[i+1]=A[i]
            i=i-1
        A[i+1]=key
```

	時間(cost)	做了幾次
for j=2 to A.length	c_1	n
key=A[j]	c_2	$n-1$
i=j-1	c_4	$n-1$
while i>0 and A[i] > key	c_5	$\sum_{j=2}^{n} t_j$
A[i+1]=A[i]	c_6	$\sum_{j=2}^{n} t_j - 1$
i=i-1	c_7	$\sum_{j=2}^{n} t_j - 1$
A[i+1]=key	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6\left(\sum_{j=2}^{n} t_j - 1\right) + c_7\left(\sum_{j=2}^{n} t_j - 1\right) + c_8(n-1)$$

Best case

i	1	2	3	4	5	6
A[i]	1	2	3	4	5	6

- 最好的狀況?
- Best case
- 全部已經排好了.
- $t_j = 1, for\ j = 2, \ldots, n$
- $T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6\left(\sum_{j=2}^{n} t_j - 1\right) + c_7\left(\sum_{j=2}^{n} t_j - 1\right) + c_8(n-1)$
- $= (c_1 + c_2 + c_4 + c_5 + c_8)n + (c_2 + c_4 + c_5 + c_8)$
- $= an + b$

Worst case

i	1	2	3	4	5	6
A[i]	6	1	2	3	4	5

- 最糟的狀況?
- Worst case
- 全部已經排好了.
- $t_j = j, for\ j = 2, \ldots, n$
- $\sum_{j=2}^{n} t_j = \frac{n(n+1)}{2} - 1$
- $\sum_{j=2}^{n} t_j - 1 = \frac{n(n-1)}{2}$
- $T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6\left(\sum_{j=2}^{n} t_j - 1\right) + c_7\left(\sum_{j=2}^{n} t_j - 1\right) + c_8(n-1)$
- $= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n - (c_2 + c_4 + c_5 + c_8)$
- $= an^2 + bn + c$

Worst case, best case, and average case

- Average case: 把所有狀況所花的時間(空間)平均起來

- Worst case: 我們通常最常分析的
  - 最糟的狀況會花的時間(最多時間) →不可能更糟了(確定)
  - Worst case不代表不常發生 (常常發生): 例如 binary search找不到!
  - 很多時候, Average case可能也跟worst case差不了多少

- 例如: 如果隨意選擇n個數字來做insertion sort
  - $t_j = \frac{j}{2}$ ←大概每次會需要移動一半數目的elements
  - 最後算出來還是一個二次方多項式 (quadratic function), 和worst case相同

比較數學的方法—Asymptotic analysis

- “預測”當input的性質改變(通常是input size改變)時, 執行時間的成長速度(growth of the running time)
- 比較兩個做相同事情的程式的時間複雜度

- “程式步驟”不是那麼精確:
  - “3n+3” 比“3n+5” 快?
  - 通常 “3n+3”, “7n+2”, or “2n+15” 執行時間都相差不遠.
  - →我們將使用一個比較不準確的方式來描述執行時間…

- 相對的, 如果是$3n^2+5n+7$和$10n+50$, n^2和n就非常重要!
- "Asymptote" 是漸近線。

Example

Usually no (if the constants are close).

- Program P and Q
- $T_P(n) = c_1 n^2 + c_2 n$
- $T_Q(n) = c_3 n$

"小時候胖不算胖"

- N很大的時候, Q 會比 P 快, 不管c_1, c_2, c_3的數值是什麼.
- Example:
  - $c_1 = 1, c_2 = 2, c_3 = 100$, then $c_1 n^2 + c_2 n^2 > c_3 n$ for $n > 98$.
  - $c_1 = 1, c_2 = 2, c_3 = 1000$, then $c_1 n^2 + c_2 n^2 > c_3 n$ for $n > 998$

Break even point

- 需要知道 c_1, c_2, c_3 的數值嗎?
- No.

$$T(n) = an^2 + bn + c = \Theta(n^2)$$

Asymptotic Notation – Big OH

- Definition [Big "oh"]:
- $O(g(n)) =$
  $\{f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$
- "f of n is big oh of g of n" (是集合的成員)
- "=" is "is" not "equal" ("∈"的意思)
- ~~$O(g(n)) = f(n)$~~
- 可以想成是Upper Bound
  (王先生打了太多全壘打, 所以是upper bound)

Example

- $3n + 2 = O(n)$?
- Yes, since $3n + 2 \leq 4n$ for all $n \geq 2$.
- $3n + 3 = O(n)$?
- Yes, since $3n + 3 \leq 4n$ for all $n \geq 3$.
- $100n + 6 = O(n)$?
- Yes, since $100n + 6 \leq 101n$ for all $n \geq 10$.
- $10n^2 + 4n + 2 = O(n^2)$?
- Yes, since $10\mathrm{n}^2 + 4\mathrm{n} + 2 \leq 11n^2$ for all $n \geq 5$.

$f(n) \leq cg(n)$ for all $n, n \geq n_0$

Example

- $1000n^2 + 100n - 6 = O(n^2)$?
- Yes, since $1000n^2 + 100n - 6 \leq 1001n^2$ for all $n \geq 100$.
- $6 * 2^n + n^2 = O(2^n)$?
- Yes, since $6 * 2^n + n^2 \leq 7 * 2^{\mathrm{n}}$ for all $n \geq 4$.
- $3n + 3 = O(n^2)$?
- Yes, since $3n + 3 \leq 3n^2$ for all $n \geq 2$.
- $10n^2 + 4n + 2 = O(n^4)$?
- Yes, since $10n^2 + 4n + 2 \leq 10n^4$ for all $n \geq 2$.
- $3n + 2 = O(1)$?
- No. Cannot find c and $\mathrm{n_0}$. $3\mathrm{n} < \mathrm{c} - 2$ 無法永遠成立.

$f(n) \leq cg(n)$ for all $n, n \geq n_0$

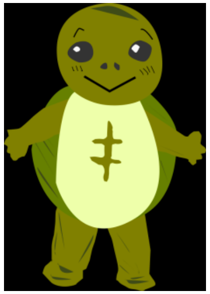
The World of Big Oh

- $O(1)$→constant
- $O(n)$→linear
- $O(n^2)$→quadratic
- $O(n^3)$→cubic
- $O(2^n)$→exponential

- $O(1), O(\log n), O(n), O(n\log n), O(n^2), O(n^3), O(2^n)$

Faster

Slower



On a 1 billion-steps-per-sec computer

n	n	$n \log_2 n$	n^2	n^3	n^4	n^{10}	2^n
10	$.01\mu s$	$.03\mu s$	$.1\mu s$	$1\mu s$	$10\mu s$	$10s$	$1\mu s$
20	$.02\mu s$	$.09\mu s$	$.4\mu s$	$8\mu s$	$160\mu s$	$2.84h$	$1ms$
30	$.03\mu s$	$.15\mu s$	$.9\mu s$	$27\mu s$	$810\mu s$	$6.83d$	$1s$
40	$.04\mu s$	$.21\mu s$	$1.6\mu s$	$64\mu s$	$2.56ms$	$121d$	$18m$
50	$.05\mu s$	$.28\mu s$	$2.5\mu s$	$125\mu s$	$6.25ms$	$3.1y$	$13d$
100	$.10\mu s$	$.66\mu s$	$10\mu s$	$1ms$	$100ms$	$3171y$	$4 * 10^{13}y$
10^3	$1\mu s$	$9.96\mu s$	$1ms$	$1s$	$16.67m$	$3.17 * 10^{13}y$	$32 * 10^{283}y$
10^4	$10\mu s$	$130\mu s$	$100ms$	$16.67m$	$115.7d$	$3.17 * 10^{23}y$	
10^5	$100\mu s$	$1.66ms$	$10s$	$11.57d$	$3171y$	$3.17 * 10^{33}y$	
10^6	$1ms$	$19.92ms$	$16.67m$	$31.71y$	$3.17 * 10^7y$	$3.17 * 10^{43}y$	

Big Oh 是 Upper Bound

- 但是沒有說它是多好的upper bound

- $n = O(n)$
- $n = O(n^2)$
- $n = O(n^{2.5})$
- $n = O(2^n)$

- 通常我們會把它取得越小(緊)越好.
- $3n + 3 = O(n^2)$ 
- $3n + 3 = O(n)$ 

一些條件

1. 正式的定義下, O(g(n))裡面的n為自然數={0,1,2,...}

2. O(g(n))的member f(n)必須為asymptotically nonnegative
   (也就是當n很大的時候, f(n)必須不為負)

3. g(n)本身也必須為asymptotically nonnegative

以上也適用於其他的asymptotic notation!

A Useful Theorem

- Theorem: If $f(n) = \boldsymbol{a_m n^m} + \cdots + a_1 n + a_0$, then $f(n) = \mathrm{O}(\boldsymbol{n^m})$.
- Proof:

$$f(n) \leq \sum_{i=0}^{m} |a_i| n^i$$

$$0 \leq n^{i-m} \leq 1$$

$$= n^m \sum_{i=0}^{m} |a_i| n^{i-m}$$

$$\leq n^m \sum_{i=0}^{m} |a_i|$$

, for all $n \geq 1$. So, $f(n) = \mathrm{O}(n^m)$.

Asymptotic Notation – Omega

- Definition [Omega]:
- $\Omega\big(g(n)\big)$ $=\{f(n)$: there exist positive constants c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$

- $f(n) = \Omega\big(g(n)\big)$
- "f of n is omega of g of n"

- 可以想成是Lower Bound

Examples

- $3n + 2 = \Omega(n)$
- since $3n + 2 \geq 3n$ for all $n \geq 1$.
- $3n + 3 = \Omega(n)$
- since $3n + 3 \geq 3n$ for all $n \geq 1$.
- $100n + 6 = \Omega(n)$
- since $100n + 6 \geq 100n$ for all $n \geq 1$.
- $10n^2 + 4n + 2 = \Omega(n^2)$
- since $10\mathrm{n}^2 + 4\mathrm{n} + 2 \geq n^2$ for all $n \geq 1$.
- $6 * 2^n + n^2 = \Omega(2^n)$
- since $6 * 2^n + n^2 \geq 2^\mathrm{n}$ for all $n \geq 1$.

$f(n) \geq cg(n)$ for all $n, n \geq n_0$

Examples

- $3n + 3 = \Omega(1)$
- $10n^2 + 4n + 2 = \Omega(1)$
- $6 * 2^n + n^2 = \Omega(n^{100})$
- $6 * 2^n + n^2 = \Omega(n^{50.2})$
- $6 * 2^n + n^2 = \Omega(n^2)$
- $6 * 2^n + n^2 = \Omega(n)$
- $6 * 2^n + n^2 = \Omega(1)$

$f(n) \geq cg(n)$ for all $n, n \geq n_0$

Discussion

- Omega is a lower bound.
- Should be as large a function as possible.

- Theorem: If $f(n) = a_m n^m + \cdots + a_1 n + a_0$ and $a_m > 0$, then $f(n) = \Omega(n^m)$.

- (證明: 請自己證證看!)

Asymptotic Notation – Theta

- Definition [Theta]:
- $\Theta\big(g(n)\big) =$

$\{f(n)$: there exist positive constants c_1, c_2, n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$

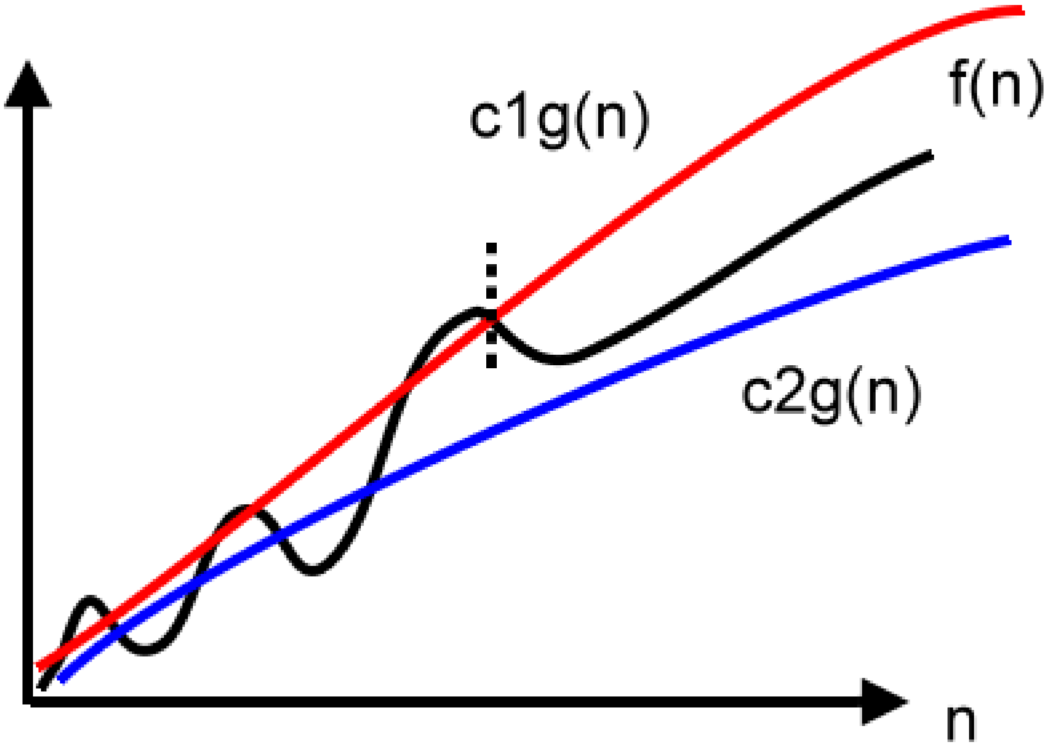
- $f(n) = \Theta\big(g(n)\big)$
- "f of n is theta of g of n"
- 三明治夾心, 同時具有$O(g(n)$和$\Omega\big(g(n)\big)$
- (Asymptotically tight)

Theta Platform

GMC Terrain

用圖表示

- Big Oh
- 紅色
- Omega
- 藍色
- Theta
- 紅色藍色都要



Example

- $3n + 2 = \Theta(n)$
- since $3n + 2 \geq 3n$ for all $n \geq 2$ and $3n + 2 \leq 4n$ for all $n \geq 2$.
- $3n + 3 = \Theta(n)$
- $10n^2 + 4n + 2 = \Theta(n^2)$
- $6 * 2^n + n^2 = \Theta(2^n)$
- $10 * \log n + 4 = \Theta(\log n)$
- $3n + 2 \neq \Theta(1)$
- $3n + 3 \neq \Theta(n^2)$
- $10n^2 + 4n + 2 \neq \Theta(n)$
- $10n^2 + 4n + 2 \neq \Theta(1)$
- $6 * 2^n + n^2 \neq \Theta(n^2)$
- $6 * 2^n + n^2 \neq \Theta(n^{100})$
- $6 * 2^n + n^2 \neq \Theta(1)$

$\mathrm{c_1 g(n)} \geq f(n) \geq c_2 g(n)$ for all $n, n \geq n_0$

Discussion

- More precise than both the “big oh” and omega notations
- It is true if and only g(n) is both an upper and lower bound on f(n).
- g(n) is an asymptotically tight bound for f(n) if f(n)=$\Theta(n)$
- 意思是說, f(n) equals g(n) within a constant factor (差常數倍而已).

- Theorem: If $f(n) = a_m n^m + \cdots + a_1 n + a_0$ and $a_m > 0$, then $f(n) = \Theta(n^m)$.
- (證明: 請自己證證看!)

o (小歐) & ω (小歐美加)

- o$\big(g(n)\big) =$ $\{f(n)$: for any positive constant c, there exists a constant n_0 such that $0 \le f(n) < cg(n)$ for all $n \ge n_0\}$
- 意思是g(n)是f(n)不緊的upper bound (長得比f(n)快)
- 也就是, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
- $\omega\big(g(n)\big) =$$\{f(n)$: for any positive constant c, there exists a constant n_0 such that $0 \le cg(n) < f(n)$ for all $n \ge n_0\}$
- 意思是g(n)是f(n)不緊的lower bound (長得比f(n)慢)
- 也就是, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

等式與不等式

- $n = O(n^2)$ (這個是什麼意思我們懂)

- $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ (????)
- 意思是 $2n^2 + 3n + 1 = 2n^2 + \mathrm{f(n)}$, 然後 f(n)= $\Theta(n)$

- $2n^2 + \Theta(n) = \Theta(n^2)$ (???)
- 不管左邊asymptotic notation代表的function怎麼選擇, 一定有一種方法選擇右邊的asymptotic notation代表的function使得等號成立.

- $2n^2 + 3n + 1 = 2n^2 + \Theta(n) = \Theta(n^2)$ (???)

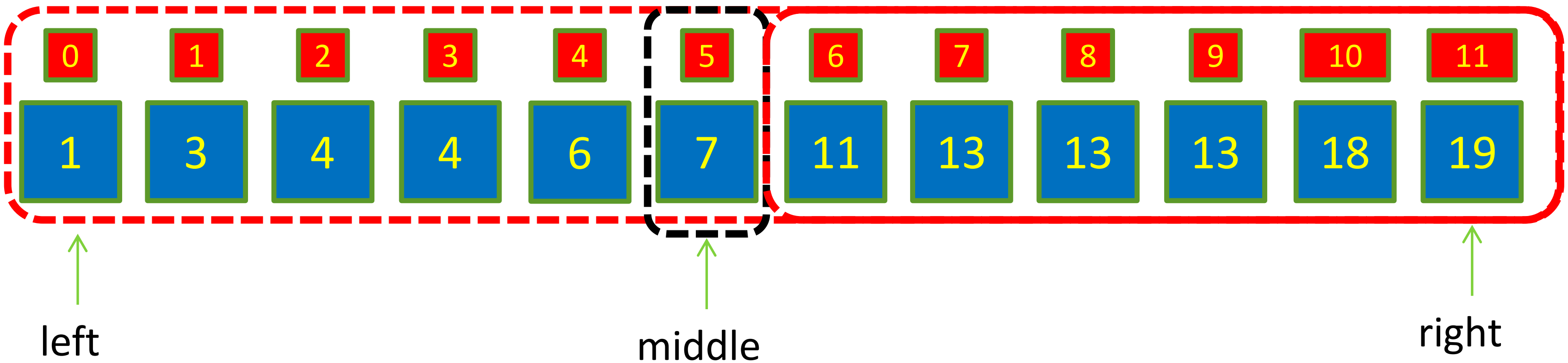
Another Example

```
int binsearch(int list[], int searchnum, int
left, int right) {
   int middle;
   while(left<=right) {
       middle=(left+right)/2;
       switch(COMPARE(list[middle], searchnum)) {
              case -1: left=middle+1;  break;
              case 0: return middle;
              case 1: right=middle-1;
       }
   }
   return -1;
}
```

Worst case: $\Theta(\log n)$

Best case: $\Theta(1)$

```
searchnum=13;
```




```
middle=(left+right)/2;
left=middle+1;
```

Half the searching window every iteration.

Another Example: recursive binsearch

```
int binsearch(int list[], int searchnum, int left, int right)
{
    int middle;
    if (left<=right) {
        middle=(left+right)/2;
        switch(COMPARE(list[middle], searchnum)) {
                case -1:
                        return binsearch(list, searchnum, middle+1, right)
                case 0:
                        return middle;
                case 1:
                        return binsearch(list, searchnum, left, middle-1)
        }
    } else
        return -1;
}
```

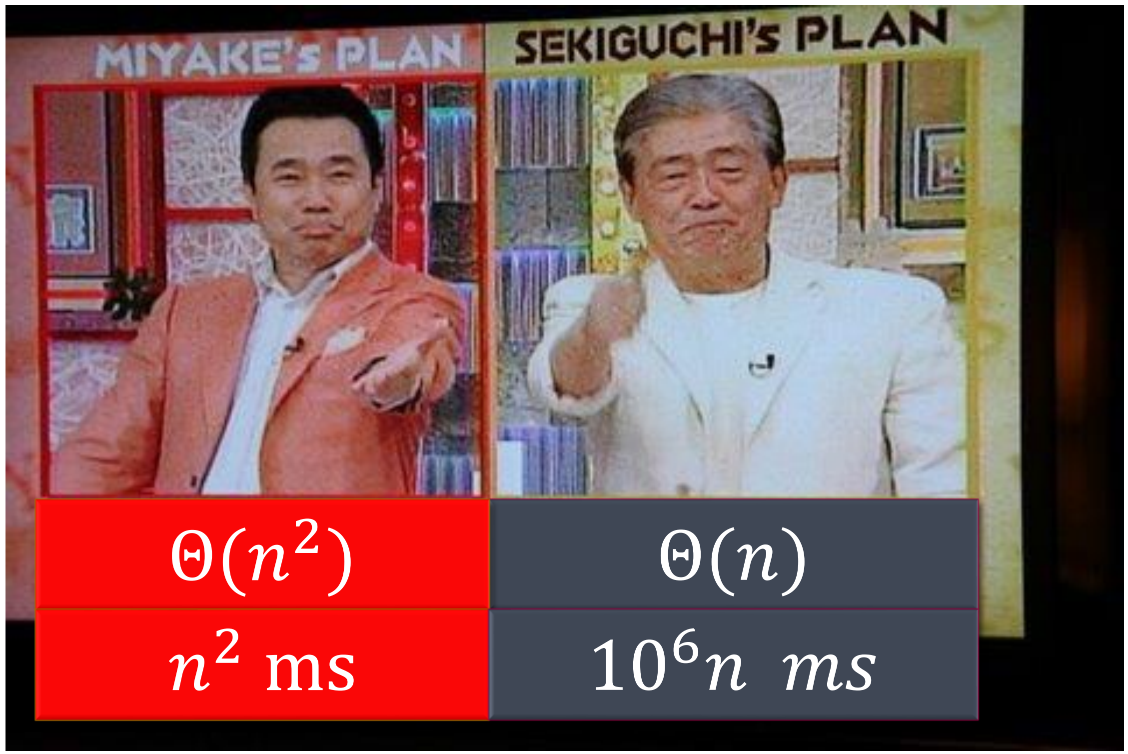
$$\boxed{\mathrm{T}(n) = T\left(\frac{n}{2}\right) + O(1)}$$

$$\boxed{\mathrm{T}(1) = O(1), T(0) = O(1)}$$

$$\boxed{\mathrm{T}(n) = T\left(\frac{n}{2}\right) + c = \left(T\left(\frac{n}{4}\right) + c\right) + c = \cdots = T(1) + c\,(n \log n)}$$

$$\boxed{\mathrm{T}(n) = O(n \log n)}$$

演算法東西軍: 兜基??



$\Theta(n^2)$	$\Theta(n)$
n^2 ms	$10^6 n\ ms$

哪一個好...

$\Theta(n^2)$	$\Theta(n)$
n^2 ms	$10^6 n \ \ ms$

- n很大的時候右邊比左邊好

- 但是n會很大嗎?
- 有時候不會
- 如果n永遠小於10^6 ??
- 結論:要看n大小 (實際上寫程式的時候適用)

Reading Assignments

- Cormen 2.1, 2.2, 3.1