CHAPTER 1

BASIC CONCEPT

All the programs in this file are selected from

Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed
"Fundamentals of Data Structures in C",
Computer Science Press, 1992.

How to create programs (Waterfall Model)

- Requirements
- Analysis
- Design: Architecture Design
  - Detailed Design – Data Type 、Algorithm
  - User Interface Design (option)
- Coding and Refinement
- Verification and Testing
- Maintain

Algorithm

Definition

An *algorithm* is a finite set of instructions that accomplishes a particular task.

Criteria

- Input $\geq$ 0
- Output $\geq$ 1
- Definiteness: clear and unambiguous
- Finiteness: terminate after a finite number of steps
- Effectiveness: instruction is basic enough to be carried out

Recursion

Definition:
函數(程式)中包含有自身呼叫的敘述

種類:
Direct Recursion, Indirect Recursion, and Tail Recursion

優缺點(與 Iteration比較):
優: 容易設計、方便閱讀、節省 code space
缺: 執行時對 stack space 之需求較高、執行效率較差

考題型態:
數學類(N!、費式數列)、資料結構類(Binary tree traversal、Quick sort)、其他(河內塔、直線切割平面)

Recursion

型態:

數學類(N!、費式數列)、資料結構類(Binary tree traversal、Quick sort)、其他(河內塔、直線切割平面)

技巧:

找出 (a) 終止條件 (b) 遞迴關係

if (a) then (結束 or 回傳值)
else return (b)

Data Type

Data Type

A *data type* is a collection of *objects* and a set of *operations* that act on those objects.

ex)

Integer: Data type

Collection of objects: -n, -(n-1), …, 0, 1, 2 … n

Set of operations: +, -, ×, ÷, > …

Abstract Data Type

Abstract Data Type

An *abstract data type(ADT)* is a data type that is organized in such a way that the **specification** of the objects and the operations on the objects is separated from the representation of the objects and the **implementation** of the operations.

Abstract Data Type

ex) Sack is an ADT

specification of the objects:

1. a set of elements
2. top pointer
3. stack size

the specification of the operations:

1. push (i, s)
2. pop (s) $\rightarrow$ i

Abstract Data Type

ex) Sack is an ADT

representation of the objects :

Array	Link-list
top: integer=0	top: pointer=nil
size: integer=n	no size limit

Abstract Data Type

ex) Sack is an ADT

the implementation of the operations

push(i, s)	push(i, s)
if stack_full then error	new t
else	t^data=i
top=top+1	t^link=top
s[top]=i	top=t

Performance Analysis or Measurement?

Performance Measurement (machine dependent)

Performance Analysis (machine independent)

- space complexity: storage requirement

  fixed space requirements: code space, simple type variable, fixed size structure variable, constant

  variable space requirements: structure parameters, recursion

- time complexity: computing time

Time Complexity compute the step count

- Introduce variable count into programs
- Tabular method
  - Determine the total number of steps contributed by each statement
    step per execution × frequency
  - add up the contribution of all statements

Iterative summing of a list of numbers

***Program 1.12:** Program 1.10 with count statements (p.23)*

```
float sum(float list[ ], int n)
{
   float tempsum = 0; count++; /* for assignment */
   int i;
   for (i = 0; i < n; i++) {
        count++;          /*for the for loop */
        tempsum += list[i]; count++;  /* for assignment */
   }
   count++;        /* last execution of for */
   return tempsum;
   count++;        /* for return */
}
```

$\boxed{2n + 3 \text{ steps}}$

Tabular Method

***Figure 1.2:** Step count table for Program 1.10 (p.26)

Iterative function to sum a list of numbers

steps/execution

Statement	s/e	Frequency	Total steps
float sum(float list[ ], int n)	0	0	0
{	0	0	0
float tempsum = 0;	1	1	1
int i;	0	0	0
for(i=0; i <n; i++)	1	n+1	n+1
tempsum += list[i];	1	n	n
return tempsum;	1	1	1
}	0	0	0
Total			2n+3

Recursive summing of a list of numbers

***Program 1.14:** Program 1.11 with count statements added (p.24)

```
float rsum(float list[ ], int n)
{
        count++;      /*for if conditional */
        if (n) {
                count++;  /* for return and rsum invocation */
                return rsum(list, n-1) + list[n-1];
        }
        count++;
        return list[0];
}
```

$\boxed{2n+2}$

Recursive Function to sum of a list of numbers

***Figure 1.3:** Step count table for recursive summing function (p.27)*

Statement	s/e	Frequency	Total steps
float rsum(float list[ ], int n)	0	0	0
{	0	0	0
if (n)	1	n+1	n+1
return rsum(list, n-1)+list[n-1];	1	n	n
return list[0];	1	1	1
}	0	0	0
Total			2n+2

Tabular Method

```
for (i=1; i<n, i++)
{
 a=(b+c)*d/e
}
```

```
for (i=1; i<n, i++)
{
 T1=b+c
 T2=T1*d
 T3=T2/e
}
```

n?

3n?

Asymptotic Notation (O)

Definition

$f(n) = O(g(n))$ iff there exist two positive constants c and n_0 such that $f(n) \leq cg(n)$ for all n, $n \geq n_0$.

Examples

- $2n+3=$
- $2n+3=$
- $3n^2+2n+3=$

Theorem (p.36)

If $f(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$ 則 $f(n) = O(n^m)$

Asymptotic Notation (O) (upper bound)

Examples

- $n^3 + n\log n + 5n^2$
- $n^6 + 100n^7 + 5000n^9 + 2^n$
- $n^2 + n^2 \log n + 5$

Basic Principle

O(1): constant

O(logn)

O(n): linear

O(nlogn)

O(n^2): quadratic

O(n^3): cubic

O(2^n): exponential

O($\frac{n}{2}^{n/2}$)

O(n!)

O(n^n)

*Figure 1.7:Function values (p.42)

		Instance characteristic n					
Time	Name	1	2	4	8	16	32
1	Constant	1	1	1	1	1	1
$\log n$	Logarithmic	0	1	2	3	4	5
n	Linear	1	2	4	8	16	32
$n \log n$	Log linear	0	2	8	24	64	160
n^2	Quadratic	1	4	16	64	256	1024
n^3	Cubic	1	8	64	512	4096	32768
2^n	Exponential	2	4	16	256	65536	4294967296
$n!$	Factorial	1	2	24	40326	20922789888000	26313×10^{33}

*Figure 1.8:Plot of function values(p.43)

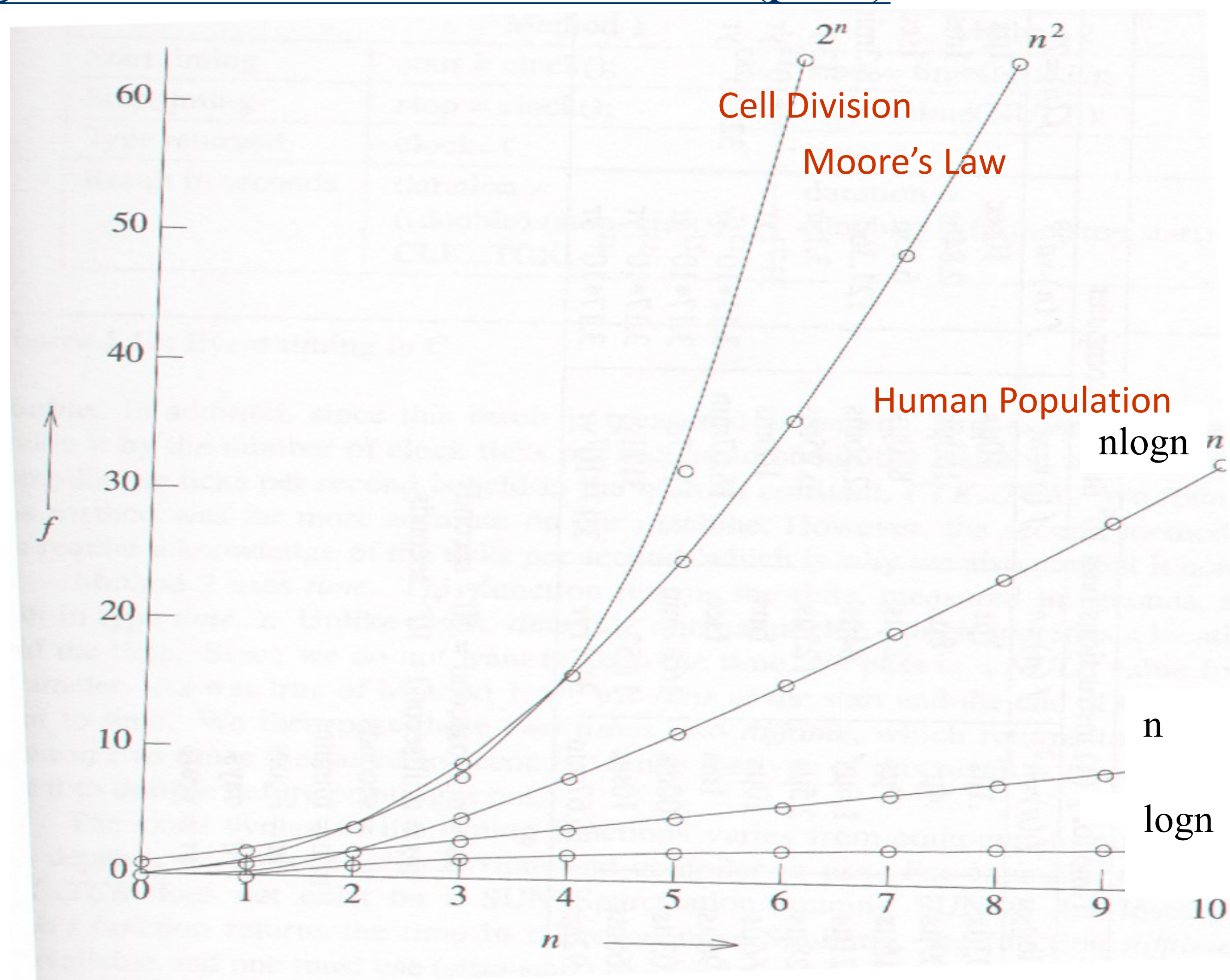

*Figure 1.9: Times on a 1 billion instruction per second computer(p.44)

	Time for $f(n)$ instructions on a 10^9 instr/sec computer						
n	$f(n)=n$	$f(n)=\log_2 n$	$f(n)=n^2$	$f(n)=n^3$	$f(n)=n^4$	$f(n)=n^{10}$	$f(n)=2^n$
10	.01μs	.03μs	.1μs	1μs	10μs	10sec	1μs
20	.02μs	.09μs	.4μs	8μs	160μs	2.84hr	1ms
30	.03μs	.15μs	.9μs	27μs	810μs	6.83d	1sec
40	.04μs	.21μs	1.6μs	64μs	2.56ms	121.36d	18.3min
50	.05μs	.28μs	2.5μs	125μs	6.25ms	3.1yr	13d
100	.10μs	.66μs	10μs	1ms	100ms	3171yr	$4*10^{13}$yr
1,000	1.00μs	9.96μs	1ms	1sec	16.67min	$3.17*10^{13}$yr	$32*10^{283}$yr
10,000	10.00μs	130.03μs	100ms	16.67min	115.7d	$3.17*10^{23}$yr	
100,000	100.00μs	1.66ms	10sec	11.57d	3171yr	$3.17*10^{33}$yr	
1,000,000	1.00ms	19.92ms	16.67min	31.71yr	$3.17*10^{7}$yr	$3.17*10^{43}$yr	

μs = microsecond = 10^{-6} seconds
ms = millisecond = 10^{-3} seconds
sec = seconds
min = minutes
hr = hours
d = days
yr = years

L'hospital Rule

Ex) $n^{0.01n}$ V.S. $(1.001)^n$

如果所有的嘗試都失敗
為何不使用計算機?
(一般的研究所考試均允許使用計算機)

屏科大87)

$A:(1.001)^n,\ B:n^{0.0001n},\ C:2^{\log n},\ D:n!,\ E:n\log n,$

(a) D最高 (b) C最低 (c) B高於A (d) E低於A (e) 以上皆是

D>B>A>E>C

中原資工89)

Arrange the following function into asymptotic asceding order

$n,\ 2^n,\ 192,\ n\log n, \sqrt{n},\ n!$

$$192,\ \sqrt{n},\ n,\ n\log n,\ 2^n,\ n!$$

輔大資管90)

Which of the following statements is not correct?

(a) 20 is O(1)

(b) n(n-1)/2 is O(n^2)

(c) max(n^3 , $10n^2$) is O(n^3)

(d) $15n^4 + 10n^3 + 100n^2 + 2^n$ is O(n^4)

(e) If p(x) is any k^{th} degree polynomial with a positive leading coefficient, then p(n) is O(n^k)

(d)

中原資工90)

List the functions below from lowest to highest order

$n,\ n-2n^3+5n^5,\ \sqrt{6n},\ \log(\log n),\ O(1),\ 2n!$

$$O(1),\ \log(\log n), \sqrt{6n},\ n, n-2n^3+5n^5,\ 2n!$$

台大電機89)

Ordering by asymptotic growth rates in asceding order

$$(\log n)!,\ (n+1)!,\ \log(n!),\ 4^{\log n},\ (3/2)^n$$

$$(\log n)! = O((\log n)^{\log n})$$

$$\log(n!) = O(n \log n)$$

$$4^{\log n} = n^2$$

$$\log(n!),\ 4^{\log n},\ (3/2)^n,\ (\log n)!,\ (n+1)!$$

交大資料)

比較 $(1.001)^n$, $n^{0.0001n}$, $2^{\log n}$, $n!$, $n\log n$ 之大小

$$2^{\log n}, n\log n, (1.001)^n, n^{0.0001n}, n!$$

Asymptotic Notation (Ω) (Lower bound)

Definition

$f(n) = \Omega(g(n))$ iff there exist two positive constants c and n_0 such that $f(n) \geq cg(n)$ for all n, $n \geq n_0$.

Examples

- $3n+3 = \Omega(n)$ /* $3n+3 \geq 2n$ for $n \geq 1$ */
- $6 \cdot 2^n + n^2 = \Omega(2^n)$ /* $6 \cdot 2^n + n^2 \geq 6 \cdot 2^n$ for $n \geq 1$ */

Asymptotic Notation (θ)

Definition

$f(n) = \theta(g(n))$ iff there exist three positive constants c_1, c_2 and n_0 such that $c_1 g(n) \le f(n) \le c_2 g(n)$ for all n, $n \ge n_0$.

Examples

- $3n+2= \Omega(n)$ /* $3n+2 \ge 3n$ for $n\ge 2$ */
- $3n+2= O(n)$ /* $3n+2 \le 4n$ for $n\ge 2$ */

成大電機)

Let $f(n)=\sum_{i=1}^{n}\log i$, show that f(n)=O(nlogn)

Ex)

Show that the following statements are correct

(a) $n^2 + 10^{100} n \in O(n^2)$

(b) $\sum_{i=1}^{n} i^2 \in O(n^3)$

(c) $n! \in O(n^n)$

屏科大)

Which of the following statements is correct?

(a) $\sum_{i=1}^{n} i^2 \in O(n^3)$

(b) $n^3 \cdot 2^n + 6 \cdot n^2 \cdot 3^n = O(n^2 \cdot 2^n)$

(c) $n^k + n + n^k \log n = \theta(n^k \log n)$

(d) $10n^3 + 15n^4 + 100n^2 \cdot 2^n = O(n^2 \cdot 2^n)$

(e) $\log(n!) = \Omega(\log(n^n))$

Ex)

```
For i=1 to n do
    For k=(i+1) to n do
    x=x+1
```

以 x=x+1 之執行次數當 T(n)，求 T(n)

Ex)

```
For i=1 to n do
    For j=i to n do
        For k=j to n do
         x=x+1
```

以 x=x+1 之執行次數當 T(n)，求 T(n)

Ex)

```
For k=1 to n do
    For i=0 to k-1 do
        For j=0 to k-1 do
         x=x+1
```

以 x=x+1 之執行次數當 T(n)，求 T(n)

Ex)

$T(n)=2T(n-1)+T(1)$, $T(1)=1$

求 $T(n)$

Ex)

$T(n)=T(n/2)+T(1)$, $T(1)=1$

求 $T(n)$

Ex)

$T(n)=2T(n/2)+n$, $T(1)=1$

求 $T(n)$

Ex)

$T(n)=T(n-1)+n$, $T(1)=1$

求 $T(n)$

Ex)

$T(n)=T(n-1)+T(n-2)$, $T(0)=0$, $T(1)=1$

求 $T(n)$