Chapter 1 Recursion

No.
Date

Recursion ⟶ Breaks problem into small problem → still same problem
(easy to solve) until solve

A binary search — Repeatedly havles the data collection and searches the one half that could contain the item.

— Uses a divide and conquer strategy [分而击之]

sample1: Writing a String Backward

Problem: Given a string, write it in reverse order

Recursive solution: diminishes by 1 the length of string [字串长度减一]

Base case: write the empty string backward. 停止条件

sample 2: GCD problem

Problem: compute the GCD of two nonnegative integers x and y

solution: gcd(x, y) = x if y = 0
= gcd(x, y mod x) if y > x
= gcd(y, x mod y) otherwise

~~smarter solution~~: gcd2(x, y) = y if x mod y = 0
= gcd(y, x mod y) otherwise

base case: gcd1(x, y) = x if y = 0, gcd2(x, y) = y, if x mod y = 0

why gcd2 smarter: gcd2 saves one recursive call when x divides y.
(using box trace)

sample 3: $K^{th}$ smallest Item in an Array

solution: KSmall(K, anArray, first, last)

if (K < pivotIndex − first + 1)
枢纽
左半 return Ksmall(K, anArray, first, pivotIndex−1)
else if (K == pivotIndex − first + 1)
return p
右半 else
return Ksmall(K, anArray, pivotIndex+1, last)

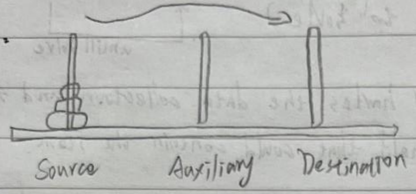
选择一個递迴

chenyu culture

Date : :

sample 4 : Tower of Hanoi problem

problem :




移動盤子到終點，大的在下，小的在上

求最短次數 到終點

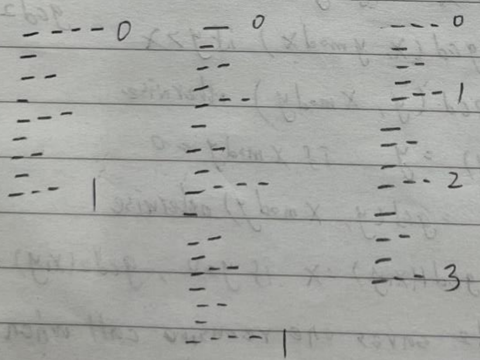
solution : 考慮最小盤子數的解法，f(2)，可求得 f(3) ...... 以此類推

three plate → two plate → solve !

```
if (numDisks is 1)
    Move disk from source to destination
else
    solveTowers (count-1, source, spare, destination)
    solveTowers ( 1 , source, destination, spare)
    solveTowers (count-1, spare, destination, source)
```

sample 5 : drawTicks — Binary Recursion

problem :



drawing ticks

solution :

找規律 ,
- An interval with a central tick length L-1
- An single tick of length L
- An interval with a central tick length L-1

⋮

No.
Date

sample 6 : Fibonacci Sequence

problem: 1、1、2、3、5、8 …… 求: 第K項

Solution:

```
int rabbit (int n)
if (n≤2)
    return 1
else
    return rabbit(n-1) + rabbit(n-2);
```

Note that n at least double every other time

that is, $n \geq 2^{k/2}$ 指數成長!

非常沒效率!

better solution:

```
if K=1 then
    return (K,0) // base case K=1 → (F1,F0)
else
    (i,j) = linearFibonacci (K-1)
    return (i+j, i) // FK = FK-1 + FK-2 , FK-1
```

呼叫次數線性成長

better choose!

sample 7: computing powers - 算次方

problem: giving integer n and double x, founding $x^n$ = ?

solution:

```
double power2 (double x, int n)
if (n==1)
    return x;
else
    return x.power(x,n-1);
```

( inefficient way )

☆

```
double power3 (double x, int n)
    if (n==1)
        return x;
else
    double halfpower = power3 (x, n/2);
    if (n%2 ==0)
        return halfpower.halfpower;
else
    return x.halfpower.halfpower;
```

※ 乘法次數與遞迴次數大幅減少!

sample 8: organizing a parade

problem: 在長度n時, 可以有幾種遊行組合

parade will consist of floats (花車) and bands in a single line

One band cannot be placed immediately after another!

solution: P(n) = 總組合, F(n) = 花車最後, B(n) = 樂隊最後

$$P(n) = F(n) + B(n)$$

∴ $F(n) = P(n-1)$ [花車殿後], $B(n) = F(n-1) = P(n-2)$ [樂隊殿後]

$p(n) = p(n-1) + p(n-2)$ #

Part of Fibonacci sequence!

$p(1) = 2$, $p(2) = 3$, $p(n) = p(n-1) + p(n-2)$ for $n > 2$

Sample 9、哪一個整數的平方最接近且低於30?

solution: binary search!

先取中點 $c = (a+b)/2$ → 二分法

base case: $c \times c <= n$ && $(n < (c+1) \times (c+1))$

※ Acker Function: 遞迴次數遞增急快!

$$\begin{aligned} \text{Acker}(m,n) &= n+1 \quad \text{if } m=0 \\ &= \text{Acker}(m-1,1) \quad \text{if } n=0 \\ &= \text{Acker}(m-1, \text{Acker}(m, n-1)) \quad \text{otherwise} \end{aligned}$$

Sample 10 Mr. Spock's Dilemma (八大行星的組合數)

problem: Choosing k out of n Things (組合數)

simpify issue: $C(n,k) <$ 必選地球 $k-1, n-1$ / 不選地球 $k, n-1$ $\Rightarrow C(n,k) = C(n-1,k-1) + C(n-1,k)$

solution:

Base case:

$C(k,k)$ 全去, $C(n,0)$ 全不去, if $k > n$ 時為 0 (無組合)

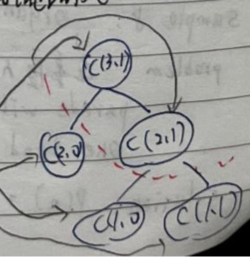
over all: $C(n,k) =$

$$\begin{cases} 1 & \text{if } k=0 \\ 1 & \text{if } k=n \\ 0 & \text{if } k>n \\ C(n-1,k-1) + C(n-1,k) & \text{if } 0<k<n \text{ otherwise} \end{cases}$$

※ 如何知道遞迴呼叫次數? (Binary Trees)

Leaf nodes (葉節點): base cases 的次數

Internal nodes (內部節點): non-base cases 的次數



C(n,k) observations:

- base cases 只會回傳 1，∴ C(n,k) = 1+1+1+1....
- |leaf nodes| = C(n,k)
- |leaf nodes| - |internal node| = 1
- |internal nodes| = C(n,k) - 1

1-18. Tail-Recursive function

有些遞迴能改為迴圈方式稱作 Tail Recursion.

ex.

```
void countDown(int n){
  if (n>0){
    cout<<n<<endl;
    countDown(n-1);
  }
}
```

Summary of Recursion:

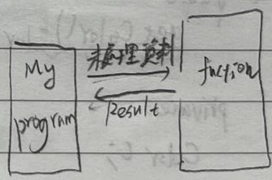
1. 遞迴定義 2. 問題簡化 3. 終止條件 4. 保證終止.

# Chapter 2 Data Abstraction

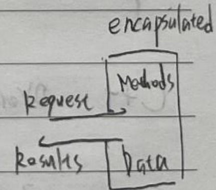
Modularity — Isolates errors
Eliminates redundancies (減少重複性)

- 如何判斷是一個好程式? High cohesion and loose coupling (高內聚，低耦合)
  高內聚: 讓每一個函式只做一件事情
  低耦合: 傳遞的參數愈少愈好

Functional abstraction:




- specifications: Detail how the module behaves independent of the implementations
- 資訊隱藏




C++ classes —

- An object is an instance of a class
- A class defines a new data type
- A class contains data members and methods 成員
- By default, all members in a class are private
  * but you can specify them as public
- Encapsulation hides implementation details 封裝

No.
Date

C++ classes : header file

```
const double PI = 3.14159;
class sphere
{
  public:
    Sphere();  // default constructor
    sphere( double initialRadius);  // constructor
    void setRadius( double newRadius);
           ⋮  ( data members)
  private:
    double theRadius;
};
```

constructors : Create and initialize new instances of a class.
Destructor : Destroys an instance of an object when the object's ends. (記憶體釋放)

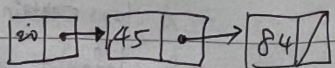
C++ Inheritance : (繼承)

```
class ColoredSphere : public Sphere
{ public:
     Color getColor() const;
  private:
     Color c;
}
```

父類別 = sphere (大)
↕
子類別 : ColorSphere (小)

C++ Overloading : 增加使用的彈性!

Chapter 3   Linked list and pointers

Linked list : 就像火車一節一節的 

與 array 之比較 : array : 需要移動資料
Linked list : 不需要移動資料

Chuyu culture

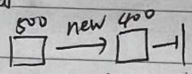
pointer：指標類似於房子的門牌

Declaration of integer pointer: int *p;

How to use: p=&x; &x 代表房子的門牌

*p → 房子的空間

The new operator: p = new int;

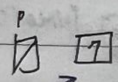



用完 pointer 記得 delete!

- delete p; 釋放記憶體
- p=NULL; // safeguard

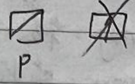
*! memory leak:

p=NULL;




造成記憶體浪費

delete p;
p=NULL;




動態陣列宣告: int arraysize=50; double *anArray = new double[arraysize];

指標 vs 陣列 取得 element 方法: *(anArray+2) ≡ anArray[2]

linked list travel:

```
for(Node* cur = head; cur!=NULL; cur=cur->next)
    cout << cur->item;
```

linked list delete:

the node in middle

```
prev->next = cur->next (prev->next->next);
```

first node

```
head = cur->next;
cur->next = NULL;
delete cur;
cur = NULL;
```

] 不能互換

linked list insert:

a node between two nodes

```
newPtr->next = cur;
prev->next = newPtr;
```

] 可以互換

at the beginning

```
newPtr->next = head;
head = newPtr;
```

] 不能互換

at the end

```
newPtr->next = NULL;
prev->next = newPtr;
```

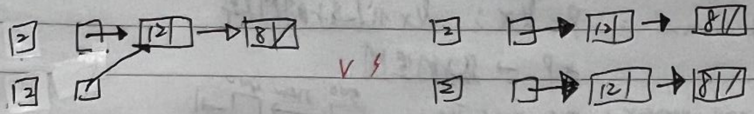
] 可以互換

No.
Date

e.g. List :: ~List
{ while ( ! isEmpty())
remove();
}

※ 動態配置的記憶體需要自己寫 Destructors!

Shallow Copy vs Deep copy




Array-Based vs Pointer-Based Implementations

- size : 空間配置 Pointer 較優
- storage requirements : Pointer 較不省空間
- retrieval : Array : 常數時間 — 較快
  Pointer : 線性時間 — 較慢

- Insertion and deletion :
  - Array-based : 搬移資料
  - 勝 Pointer-based : 更新資料

. Use File to Saving and Restoring a Linked list.

ifstream 讀入宣告 out stream 寫入宣告

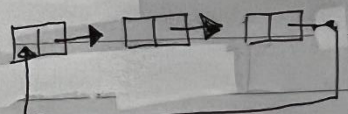
e.g. ifstream in ; e.g. outstream out ;

in >> item ; 讀檔 out << item ; 寫檔

. 進行 linked list 遍歷 → 利用 head !

. Variation

- circular Linked list
- Dummy head Node ⇒ to eliminate the special node !
- Double Linked list (雙向指標)



< circular Linked list >

Double Linked list
{
Node * precode
Node * next
}

Chapter 4

No.
Date :

- 定義語言 Defining Language

  x | y (x or y)

  xy or x·y (緊連著)

- <number> = <digit><number> | <digit>
- <digit> = 0 | 1 | 2 | 3 | 4 | ... | 9 |
- <addition> = <digit> + <addition> | <digit>
- 2位3值以上 <addition> = <number> + <addition> | <number>

The Basic of grammar → recognition algorithm (辨識算法)

isId (遞迴), isId suffix, 終止條件：單一字元 / 遞迴時：扣除最後一字元

Palindromes (迴文)

ex. Anna, 38+83=121

<pal> = empty string

<cn> | a <pal> a | b <pal> b | ... | z <pal> z

<cn> = a | b | c | ... | z.

$A^n B^n$

<Legal-word> = empty string | A <Legal-word> B

Algebraic Expression (代數)

- Infix 中序運算式 ex: A+B
- Prefix 前序 〃 ex: +AB
- Postfix 後序 〃 ex: AB+

Advantage of prefix, postfix:

- No precedence rules (優先權)
- No association (結合律)
- No parenthese (括弧)

※ 中序轉前序

((a+b) * c)

⇒ * + abC

No.
Date : :

prefix

- Grammar
  <prefix> = <identifier> | <operator><prefix><prefix>
- recursive recognition
  - Base Case = One lomer case is prefix exp
  - Recursive = <operator><prefix><prefix>

* 重要特性：一個前序式後面再接上非空字串不一定是前序式.

- Back tracking (回溯)
  - 八皇后問題 8! 種方法 = Place eight queens on the board so that no queen can attack any other queen

思路：遇到僵局就退回，避開會被攻擊的位置

- Recursive
  - Base case = 八欄填滿
  - Recursive step = 續填其他欄位.