一. 遞迴.

(1). 原理：讓問題越來越小，直到消失.

優點：可以用相同的方法解小問題. 精簡程式碼

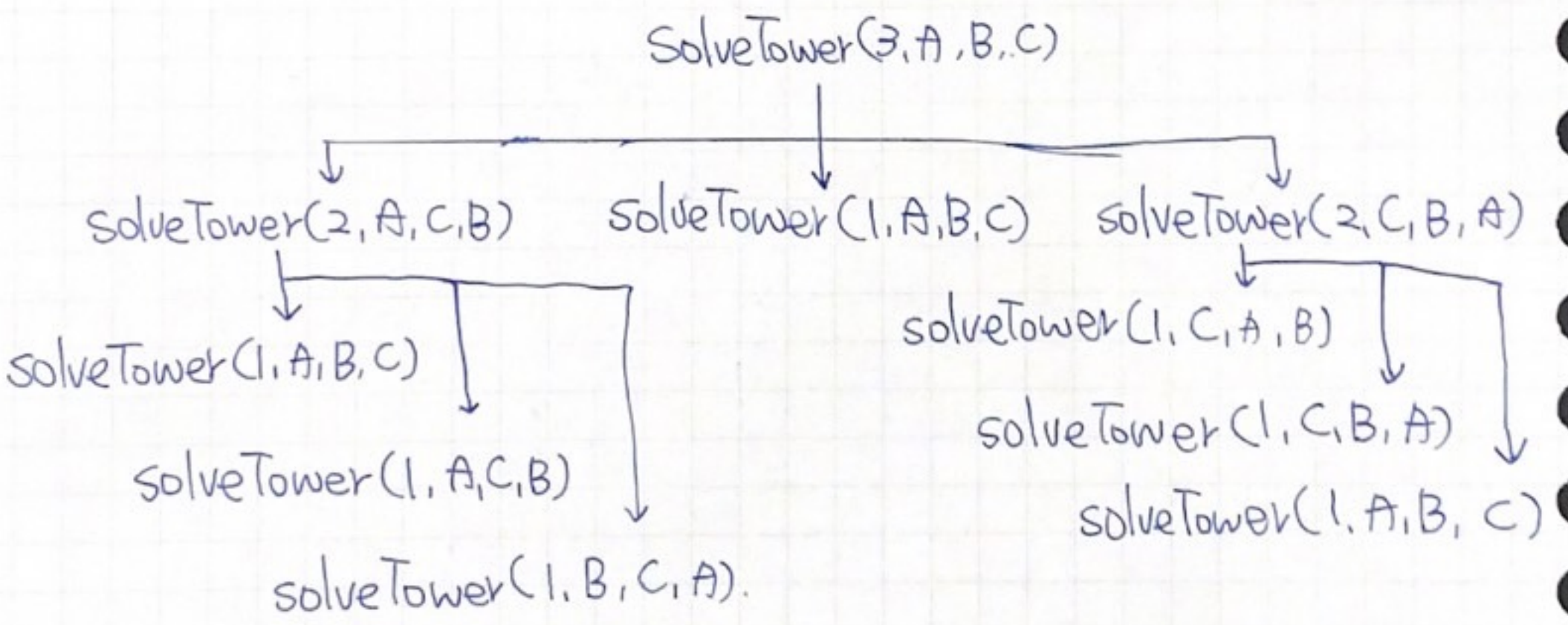
divide and conquer 分而擊之

遞迴呼叫在不同程式段的位置，結果大不同

例：
```
int sum(int a, int b) {   // assume a > b.
    if (b == a)  return a;
    return sum(a, b+1) + b;
}
```

1. 遞迴定義 sum(a+...b) = sum(a+...+b+1) + b
2. 問題簡化 b → b+1
3. 終止條件 b == a
4. 保證中止 natural numbers a > b

Towers of Hanoi (solveTower(個數, 起點, 終點, 輔助)).



▲Multiplying Rabbits

rabbit(n) = rabbit(n-1) + rabbit(n-2).

費氏數列：1. 1. 2. 3. 5. 8. 13. 21...

① int rabbit (int n). {

    if (n <= 2) return 1 ;

    else return rabbit(n-1) + rabbit(n-2);

}

呼叫次數以指數成長 $n_k \geq 2^{\frac{k}{2}}$, bad.

② linearFibonacci (int k) {

    if (k == 1) return(k , 0).

    else { (i , j) = linearFibonacci (k-1)

        return (i+j , i)

    }

呼叫次數以線性成長 k , good.

二. 抽象化.

描述 ADT 運算在前, 之後進行 ADT 實作

目地. 假設. 輸入. 輸出.

C++ Classes: Constructors (類別).

Inheritance in C++ (繼承). [父類別: Sphere (大)
子類別: ColorSphere (小)

Overloading in Class Rational (多載).

[Private: only class instances
Protected: subclass instances
Public: any class instances]

[覆載 (overriding)
多載 (overloading).

ADT Polynomial

1. degree(): 最高次項的指數.
2. coefficient (in power): Power次項的係數.
3. changCoefficient (in newCoefficient, in power): 改變

⚠ 非負

ex: $p = 4x^5 + 7x^3 - x^2 + 9$.

- p.degree() = 5
- p.coefficient(3) = 7
- p.changCoefficient(-3, 4) = $p = 4x^5 - 3x^4 + 7x^3 - x^2 + 9$.
- p.changCoefficient(p.coefficient, p.degree) = $p = 9x^5 - 3x^4 + 7x^3 - x^2 + 9$

C++ Namespace

using namespace 使用命名空間

std 標準函式庫

using namespace std;

C++ Exceptions 例外處理

try: 設定保護範圍.

catch: 捕捉例外狀況.

三. 鏈結串列.

陣列：需移動資料.

鏈結串列：不需移動資料.

int *p ; 還是空的.

一般變數直接分配.

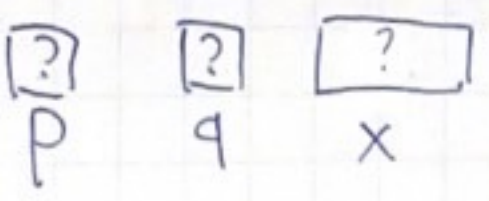
p = new int ; 有意義. (有可能要不到).

delete p; 歸還記憶体空間.

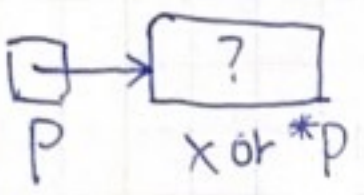
p = NULL ; 遺忘空間內容.

ex:

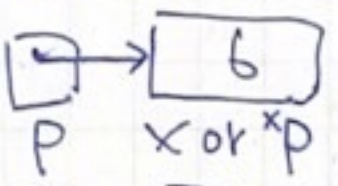
(a). int *p, *q;
int x;



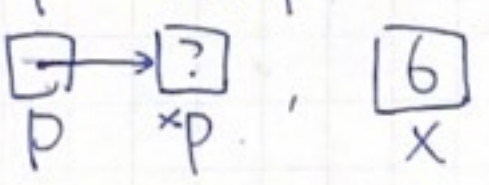
(b). p = &x;



(c). *p = 6;

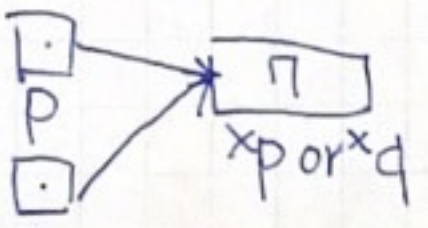


(d). p = new int ;

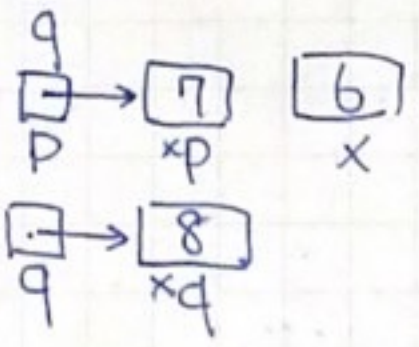


(e) *p = 7;

(f). q = p;



(g). q = new int;
*q = 8;



(h). p = NULL

(i). delete q;
q = NULL;

A Pointer-Based Implementation of the ADT List

List(const List &aList) 複製建構.

```
struct ListNode {          節点結構.
   ListItemType item;
   ListNode *next;
};
```

ListNode *find(int index) const ; 定位搜尋.

檢索
- 陣列：常數時間（快）
- 串列：線性時間（慢）.

新增/刪除
- 陣列：shifting（搬移）.
- 串列：traversal（走訪）.

各有好壞.

動態陣列

```
int arraySize = 50;
double *anArray = new double[arraySize];
```

陣列名稱 = 指標.

Pointer-Based Linked Lists

```
struct Node {
    int item;
    Node *next;
};
```


```
Node *p;
p = new Node;
head = new Node;
```

delete 先, NULL 後.