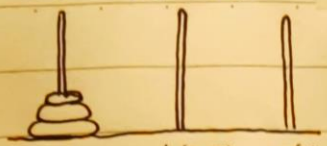
No. Date

單元一 recursion

Iteration	vs	Recursion
通常時間較短		通常時間較長
不需儲存 return 且不用在 function 間一直跳轉		需一直呼叫 function 且不斷傳遞參數
stack 快速成長		有可能呼叫太多次 function call stack 直到溢位
在實作一些複雜題目 程式較繁瑣		把大問題分成小問題 (分而擊之)

■ No. ■ Date

河內塔的遞迴方法



想法

SolveTowers (count, source , destination , spare)

if (count is 1)
  // 把盤放進所設的終桌
else {
  solveTower (count-1, source, spare, destination)
  solveTower (1 , source , destination , spare)
  solveTower (count-1, spare , destination , source)

■ No. ■ Date

費氏數列 9 遞迴方法

1, 1, 2, 3, 5, 8, 13, 21

Base Case

f(1), f(0)

Recursive

$f(n) = f(n-1) + f(n-2)$ if $(n>1)$

※注意呼叫次数の差異

(想出呼叫次数少の想法)

$n_1 = 1$

$n_2 = 1$

$n_3 = n_2 + n_1 + 1 = 3$

$n_4 = n_3 + n_2 + 1 = 5$

⋮

No. Date

Recursive Solution

① 不斷呼叫自己

② 需最少1個 base case (終止條件)

③ 把問題逐漸變小直到 base case

常解問題

the factorial of N

Search in Array

Fibonacci series

Towers of Hanoi

Combinatorial numbers

No. Date

尾端遞迴

```
Void CountDown (int n){   //可將 if 改成 while 變成迴圈
    if (n>0) {
      cout << n << endl
      countDown (n-1)      * tail Recursion ✓
    }
}

int fact (int n) {
    if (n==0)              //不是 tail Recursion
        return 1;  → result
    else
        return n* fact(n-1)
        return fact2(n-1, n*result);
```

Packages

No. Date

單元二 資料抽象化

物件導向優點：

① 將各項資料分開存放
② 能輕鬆找到所需資料
③ 各項獨立運作
④ 利用 public, protect, private 保護 or 公開 隱私資料

ADT List Operation

Create a List
Destroy a List
Is empty( )
Determin the number of items in lists
Insert items
Delete items
retrieve items

■ No. ■ Date

ADT Sorted List

① 維持原本 item data 在 Sorted List

② 插入 or 刪除皆對其值，非存放位置

依值排序

Constructors 建構子

① 是一個 class 裡的子程式

② 用來初始化物件 // new

③ 必須與類別 class 同名

④ 不可 return 值

⑤ 可帶入參數

⑥ 可有多个建構子，型態不可相同

■ No. ■ Date

Destructor

① 物件的成員函數，沒有 返回值 And 引數
② 1个 class 只有一個 Destructor
③ 自動建立預設解構函數 編譯器
④ 可以 設成 private
⑤ 當物件要結束 ex：程式 end …… 會被拿出來

怎麼做例外處理

for handling an error

① function 表明 error occurred by 丟出例外情況

try block 設定保護範圍

// 試看看有沒有出錯

```
try {
  ......
}
```

catch block 捕捉例外狀況

// 處理例外情況

```
catch {
  ......
}
```

不同例外狀況處理程序

No. Date

單元 3　鏈連結串列

Pointers

① 一個指標包含 address (記憶体的位置)

ex:

```
int * p;        // 不是 NULL
p = &x;
*p = 6;
p = new int;
*p = 7;
```

■ No. ■ Date

Linked List

① 如果 head 是 NULL，linked list 為 empty

② 如果 head 本來有存 data，此時 head = NULL，則會 Lost cell (memory leak)

How to Delete

① 先用 cur 指向要刪除的 node，前一个 prev -> next = cur -> next;
略過再 delete cur; ⇒ cur = NULL;

How to Insert

① 先用 newPtr 指向要加入的 node，cur 為要加入 node 的後一項，prev 為要加入 node 的前一項.

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

Ex:
```
for (prev = NULL, cur = head;
    (cur != NULL) && (newValue > cur -> item);
    prev = cur,  cur = cur -> next;) {
}
```

■ No. ■ Date

Remarks

* new and delete 使記憶体能夠動態的分配跟循環

Public method

IsEmpty()

getLength()

Insert

remove

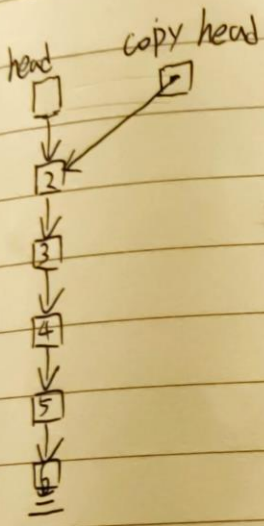
retrieve

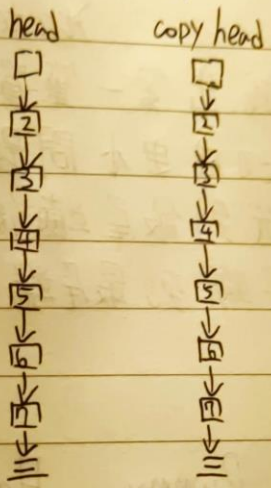
Private method

find

■ No. ■ Date

Shallow Copy vs Deep Copy





Array vs Linked List

用隱密的方法存data

用明確的方法存data

可用確切位置找data

得遍歷 Linked list

複製需要求相同大小array

可用 shallow Copy
Deep Copy

■ No. ■ Date

Linked List Recursion

Strategy

① 找出第一筆資料 (head)
② 切成更小問題
③ 先到最尾端後 return 回來做事
④ 不斷到最尾端且不斷做事

Dummy Head Node

① 永遠是空的
② 放在全部資料的最前端
③ 當 Insert, Delete 時，prev 可指向 dummy head node，而不用指向 NULL

單元 4　以遞迴解題

語法：符號的使用

- x | y ：　x 或 y

xy or x•y：　x 緊接著 y

判斷是否為迴文

ex:

```
isPal (in w:string): boolean
    if (w is the empty string or w is of length1)
        return true;
    else if (w's first and last characters are the same letter)
        return isPal (w minus its first and last chara
    else
        return false;
```

遞迴呼叫扣頭尾字元

■ No. ■ Date

- 寫一套遞迴文法描述至少1个字母所組成字串的語言，第一个字母必須是大寫，而其餘字母必須是小寫

運算式

((a+b)*c)/d　　中序運算式

/*+abcd　　前序運算式

ab+c*d/　　後序運算式

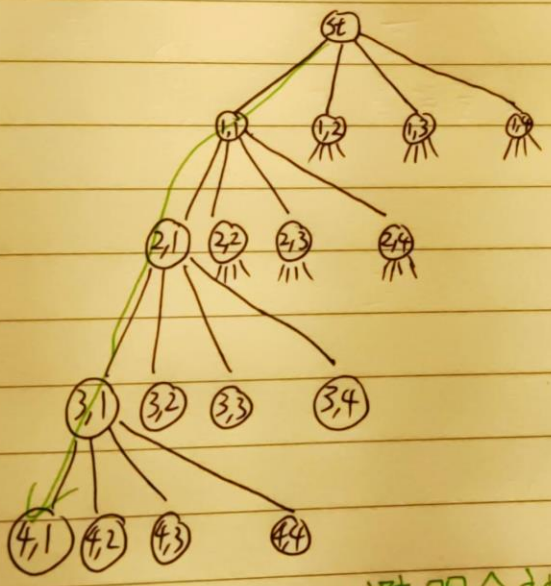
prefix (前序) 和 postfix (後序) 优点

- No. 优先权
- No. 結合律
- No. 括弧
- 簡易文法
- 較直觀

■ No. ■ Date

Backtracking

八皇后問題



避開會被攻擊的位置

終止條件:
8欄填滿

遞迴呼叫:
續填其餘欄位