Ch.1 遞迴

Today: / /

(工具、方法)
遞迴：希望問題可以越來越小 ⇒ 直至解決問題

同一問題 ⇒ 相同方法解決類似問題只需一種程式碼
⇒ 精簡、易解釋

divide and conquer 分而擊之

Ex 階層、最大公因數
☆(搜尋)、量式排列、組合數
河內塔…等(可用遞迴)

Ex1 倒印①字串

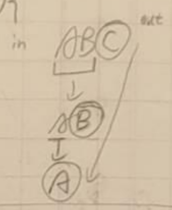
Step 1 縮小問題 ⇒ 字元數量↓
Step 2 Base Case (停下來) ⇒ 空字串

輸出記數: "t" [s='cat' size=3] → "a" [s='ca' size=2] → "c"(遞迴前) [s='c' size=1] → 0
└ 邊輸出邊跑

輸出第1: 'c' s=cat → 'a' s=at → 't' s=t → s=0

" s='cat' → s='at' → s='t'
t a c
跑完才輸出 (遞迴後)

```
void writeBackward (string s, int size){
  if (size>0){          // 字元位置, 數量
  cout << s.substr(size-1, 1);
  writeBackward (s, size-1); (pointA)
  }
}
```



Practice 1-1 a>b

```
int sum (int a, int b){   // int sum=0
 if a≥b
  return sum(a, b+1)+b;
}
```

```
if (b==a)
  return a;
return sum(a,b+1)+b;
```

Ex2 GCD

gcd1(x,y) = x if y=0
= gcd1(x, y mod x) if y>x
= gcd1(y, x mod y) otherwise

[x=9 y=6 gcd1(6,3)] → [x=6, y=3 gcd1(3,0)] → [x=3 y=0 return3]

gcd2(x,y) = y if x mod y = 0
= gcd(y, x mod y) otherwise

[x=9, y=6 gcd2(6,3)] → [x=6, y=3 return 3] ⇒ gcd2 is more efficient

special case: x<y, gcd2 = gcd1,

Binary Search with an Array (anArray, first, last, value)

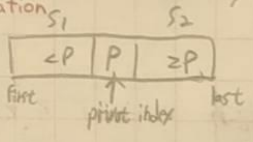
```
if {base case}
⇒else{ int mid
   if value = Array[mid]
   else if (value < Array[mid] ..., first, mid-1, ..  求解左半邊
   else                      ..., mid+1, last, ..  求解右半邊
```

Ch1 遞迴

My Questions
Problems & Difficulties needing exploration

找第k小
Step 1 找樞鈕
Step 2 只找一邊

排序，可找到所選樞鈕之正確位置

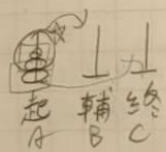


效率有機會更快

My learning weather report

Linear Recursion
① 終止條件
② 只撰一遞迴

河內塔：2^n-1 次　　B→C　C→B
對2來說 輔→終 終→輔



⇒最底到C，其餘在B

$f(h) = f(h-1) +$

Binary Recursion
→ Draw Ticks

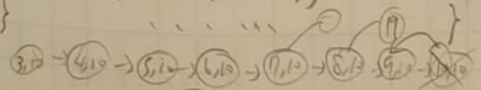

```
drawTicks (length)
Input : length of tick
Output : ruler with tick of the
  given length in the middle and
  smaller rulers on either side
```

```
drawTicks (length)
if (length>0) then
  drawTicks (length-1)
  draw tick of the length
  drawTicks (length-1)
```

My Opinions
Thoughts, inspirations, and suggestions

// Linear recursion　　3-10

```
int sum (int a, int b){
  if(a=b)
    return a;
  return a+sum(a+1,b);
}
```



遞迴 8 call ♛ (n)

記憶 +：7 (n-1)

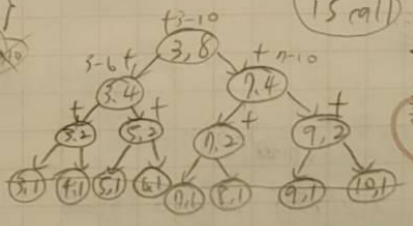
// Binary recursion　　分半，判斷 >mid or <mid

```
int sumB (int a , int h) {
  if(h==1)
    return a;
  return sumB(a, h/2) + sumB(a+h/2, h-h/2);
}
```

絕對才找到 分組
當 first > last ⇒ 找完了

first last

15 call 遞迴 (2n-1)



+：? 計算 (n-1)

在對的時間，做對的事，
以表明對人的重視。

My Notes

Important Concepts worth keeping

ch1 遞迴

Today: / /

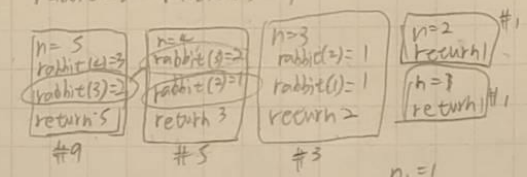
兔子問題：兔子隔一個月有繁衍能力 ⇒ Fibonacci sequence 費氏數列

1st : 1 pair
2nd : 1 pair
3rd : 2 pairs
4th : 3 pairs
5th : 5 pairs
6th : ? pair

目前的數量 + 可繁殖的能力

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

rabbit(1) = rabbit(2) = 1



recursive to recursive time

$n_1 = 1$
$n_2 = 1$
$n_3 = n_1 + n_2 + 1 = 3$

⇒ n_k at least doubles every over time ⇒ $n_k \geq 2^{\frac{k}{2}}$ ⇒ 呼叫次數以指數成長

⇒ 因重覆執行 bad ⇒ 以空間換時間 (儲存陣列)

```
linear(k):
  if(k=1)
    return(k,0)
  else
    (i,j) = linear(k-1)
    return (i+j, i)
```

(5,3) (3,2) (2,1) (1,1) (1,0)
5 4 3 2 1

呼叫次數以線性成長

求 x^n

1. 迴圈

```
power1(x,n)
while(n>0){
  result *= x;
  --n;
}
```

$q^4 = (((1 \times q) \times q) \times q) \times q$
4 multiplications

2. 遞迴

$x^0 = 1$
$x^n = x \cdot x^{n-1}$, if $n > 0$

```
power2(x,n){
if(n=1)
  return x;
else
  return x* power2(x,n-1);
}
```

$q^4 = q \times (q \times (q \times q))$
4 multiplications
4 recursive call

3. 遞迴

$x^0 = 1$ ← 二分法 效率↑

$x^n = (x^{\frac{n}{2}})^2$, if n>0, n even

$x^n = x \times (x^{\frac{n}{2}})^2$, if n>0, n odd

```
power3(x,n)
if(n=0)
  return 1;
else{
  double halfpower = power3(x, n/2);
  if(n%2==0)
    return halfpower * halfpower;
  else
    return x * halfpower * halfpower;
}
```

$q^4 = ((q \times 1) \times q) \times q) \times 1$
4 4

$q^4 = (q \times q) \times 1$
2 2
3

"Do not worry about tomorrow; for tomorrow will care for itself. Each day has enough trouble of its own." (New Test...

ch1 遞迴

My Questions
Problems & Difficulties needing exploration

My learning weather report

隊伍排列

最後 / 最後 — 多步推

$P(n) = F(n) + B(n) \Rightarrow P(n) = P(n-1) + P(n-2)$

P○□○ — 旗手, 步兵, 不相連

$F(n) = P(n-1)$, $B(n) = F(n-1) = P(n-2)$

Base $P(1) = 2$ □○ , ○□

$P(2) = 3$ □○ , ○□ , ○□ → 費氏數列

P1-4
getValue (1, 30, 30) → getValue (1, 14, 30) → getValue (1, 6, 30) → getValue (4, 6, 20)

$c = \frac{31}{2} = 15$, $c = \frac{15}{2} = 7$, $c = \frac{7}{2} = 3$, $c = 5$

225 > 30 , 49 > 30 , 9 < 30 , 5×5 < 30 < 6×6 ⇒ return

5 ⇐ a=1, b=30 ⇐ a=1, b=14 ⇐ a=1, b=6 ⇐ cout a=4, b=6

P1-5
Acker(m,n) = n+1 — m=0
= Acker(m-1, 1) — n=0
= Acker(m-1, Acker(m, n-1)) , 其他

Acker(1,2) = Acker(0, Acker(1,1))
= Acker(0, Acker(0, Acker(1,0)))   [Acker(0,1) = 2]
= Acker(0, Acker(0,2))   [3]
= ④

k 取 n
一、 從選 x → (k-1)(n-1) recursion
不選 x → k (n-1)
① 簡化

$c(n,k) = c(n-1, k-1) + c(n-1, k)$

base { $c(k,k) = 1$ or $c(n,0) = 1$ } — if(k>n)

c(n,k)
1 — if k=0
1 — if k=n
0 — if k>n
— if 0<k<n

My Opinions
Thoughts, inspirations, and suggestions

$C(4,2) = 6$ 二元樹

c(3,1) → c(2,0), c(2,1) → c(1,0), c(1,1)
c(3,2) → c(2,1), (2,2) → c(1,0), c(1,1)

遞迴次數 ⇒ 2 C(n,k) − 1

c(n,k) = |葉子| − |非葉子| = 1 [c(n,k) − 1]
base
6 − 5 = 1 (✓)

尾端遞迴

ex.
```
write Backward (string s, int size){
  if (size > 0)
  { cout << s.substr(size-1, 1);
    write Backward (s, size-1)
  }
}
```
有公式可循
→ 易改成迴圈

部分編譯器執行時會轉迴圈執行增快 [效]

Summary — ex $C(n,k) = C(n-1, k-1) + C(n-1, k)$
1. 定義
2. 簡化
3. 終止條件 — $c(k,k) = 1$, $c(n,0) = 1$
4. 保證終止 — check

不要為明天憂慮，因為明天自有明天的憂慮，一天的難處一天當就夠了。《新約聖經》

密碼 cipher key

My Notes

Important Concepts worth keeping

ch1 遞迴

Today: / /

Conclusion:

遞迴：解決問題的一項工具，相同問題又需要一程式碼

優：精簡，易解釋

缺：難 debug

Step 1、定義：找出相關但較小的問題。

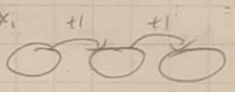
Step 2、簡化：統整小問題，使原本的問題可縮小。

Step 3、終止條件：找到其最小的問題(base case)的答案，使問題可被解決。

Step 4、確保終止：確定所有情況都有對應的 base case。

二元搜尋：用分而擊之的策略(divide and conquer) (Step1. Step2. Step3. Step4.) → 找k小

Linear recursion (線性遞迴)：① 終止條件 ② 只擇一遞迴



呼叫次數線性成長

ex. sum

Binary recursion (二元遞迴)：① 終止條件 ② 每次遞迴性務減半

ex: h取k

Fibonacci (費氏數列)：1, 1, 1+1=2, 1+2=3, 2+3=5

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

$T(1) = 1$

$T(2) = 1$

呼叫次數指數成長

ex: 兔子問題 (迴圈較好)

(尾端遞迴)：① ☆最後才遞迴 ② 易轉迴圈

ABCDE

EDCBA

心得：

這次學習了遞迴的觀念，了解具体的步驟，之後不再像之前自己寫的遞迴卻不知道那裡出錯，也認識了一些遞迴的種類感覺可以更清楚的將問題分解後寫成遞迴，還知道遞迴較適用性務重要高的問題，收高匪淺。

So in everything, do to others what you would have them do to you...
《New Testament》

ch2 抽象化

My Questions
Problems & Difficulties needing exploration

My learning weather report

Data Abstraction

描述.OOT運算在前，主從進行 OOT宣作

函數式 v.s. 物件導向式 (O.O.P) ～ 所有東西都是物件，相似的歸類在一起

記什麼資料 ～ 描述方式：① Attributes (adj, N.) ② Behaviors (V.)

*使用者可產生錯：Encapsulation，封裝 ～ hide inner details

Inheritance 繼承 ～ reused

Polymorphism 多型 ～ 擴充

運算合約 ～ 1. Purpose
2. Assumption
3. Input
4. Output
+
例外狀況

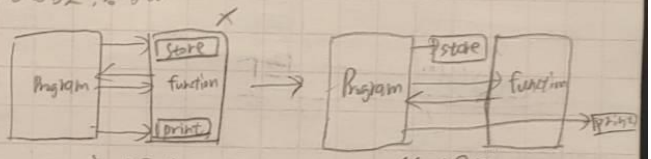
Abstrace Data Types：(motives)

o Modularity (模組化) → 容易寫，容易讀，容易改

好的方法：

Cohesion：高內聚

Coupling：低耦合

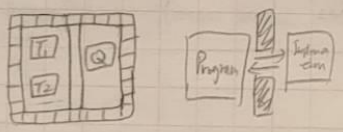


高耦合　　低耦合

Functional abstraction (功能性的抽象化)

描述與宣作分開 ⇒ 資訊隱藏 (宣作)

(concepts)



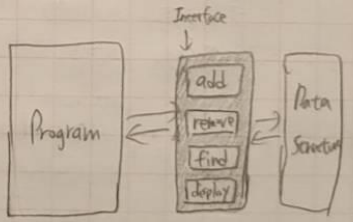
My Opinions
Thoughts, inspirations, and suggestions

(goals) Data abstraction (資料抽象化)

conclusion

OOT = data collection + a set of operations on data

分開
- Specification 描述 → for user
- Implementation 宣作



a wall of OOT operations isolates a data structure from the program that uses it.

密碼 cipher key

所以無論何事，
你們想要人怎樣待你們，
你們也要怎樣待人。
《新約聖經》

Ch2 抽象化

Today: / /

Specifying ADTs: Grocery list

predecessor 先行者
successor 後繼者

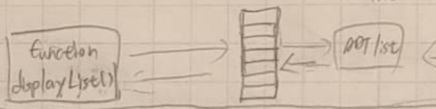
有順序: milk, egg, butter, apple, bread, chicken

從資料看 ⇒ 擴充性高

⇒ ADT list Operation

- 建構 createList()
- 解構 destoryList()
- 是否為空 isEmpty() {query}
- 計算個數 getLength() {query}
- 插入 insert(in index, in newItem, out success) ← 原位置往後 (index: int, newItem: ListItemType, success: boolean)
- 刪除 remove(in index, out success) ← 後面往前移
- 檢索 retrieve(in index, out dataItem, out success) {query} (dataItem: ListItemType)

ex.




```
displayList(in aList)        // aList: List   顯示
  for (p=1 to aList.getLength())
  { aList.retrieve(p, dataItem, success);
    Display dataItem;
  }
```

```
reverseList(in aList, out success)      // aList: List, success: boolean
{ for (i=1 to aList.getLength()-1)
    aList.retrieve(1, dataItem, success);
    aList.remove(1, success);
    aList.insert(aList.getLength()-i+2, dataItem, success);
}
```

```
replace(in aList, in i, in newItem, out success)   // aList: List, i: int, newItem: ListItemType, success: boolean
{ aList.remove(i, success);
  if (success)
      aList.insert(i, newItem, success);
```

⇒ ADT sort list

- 是否為空 sortEmpty() {query}
- 計算個數 sortGetLength() {query}
- 新增 sortInsert(in newItem, out success) (newItem: ListItemType, success: boolean)
- 移除 sortRemove(in index, out success) (index: int)
- 檢索 sortRetrieve(in index, out dataItem, out success) {query}
- 定位 sortPosition(in anItem, out isPresent) {query} (anItem: ListItemType)

```
listHoliday(in year: integer)
  date = firstDay(year)
  while (isBefore(date, firstDay(year+1)))
  { if (isHoliday(date))
      write(date, "is Holiday");
    date = nextDay(date);
  }
```

從資料 → 問題

appointment consist of a date, time and purpose

createAppointmentBook()

- make appt 建構, 新增
- cancel appt 取消
- is appt 是否有約
- check appt 約會目的

} 以此來寫函式

Ch2 抽象化

My Questions
Problems & Difficulties needing exploration

Implementing ADTs

Constructors → 宣告, class X{ X(); }
沒有 return (可加動態)

class 可有複數 constructor

Destructor → 解構

```
#include <iostream>
#include "Sphere.h"
using namespace std
int main()
{ Sphere unitSphere();   宣告,建構
  Sphere mySphere(5.1);
  cout << mySphere.getDiameter() << endl; 執行
}
```

class Rational { protected: long n; long d; void reduce(void);
public: Rational add(Rational);
Rational add(long);
}
Rational Rational::add(Rational r){}
Rational Rational::add(long i){
Rational r;
r.setRational(i, 1);
return add(r);

My Opinions
Thoughts, inspirations, and suggestions

Array Based ADT

ex: ADT polynomial
- degree 指最高次
- coefficient 係數 (pwer)
- changecoefficient 換係數 (new, power)

P2,3
1. 題子最高項係數(P)
2. x^3 係數 +5 (P)
3. polynomial 相加(P,Q)

1. display(p.coefficient(p.degree));
2. changecofficient (p.coefficient(3)+5, 3)
3. for(i=0; i<p.degree(); i<q.degree; i++)
r.changecoefficient (p.coefficient(i)+q.coefficient(i), i);

My learning weather report

* C++ classes
由物件組成, 包含 data members, methods 分開
通常均為 private, 但可以改成 public
default 私密
描述 Classname.h (header file)
實作 Classname.cpp (implementing file)

```
#include "Sphere.h"
enum Color {R, B, G, Y};
class ColoredSphere : public Sphere
{ public:
  Color getColor() const;
  ...
private:
  Color c;
}
```

父類別 Sphere
base class
子類別 ColoredSphere
可替換

ColorSphere is a Sphere

Private : only class
Protected : subclass in stages
Public : any class
(全域函式可使用)

class Integer : public Rational {
public:
void setRational(long, long);
void setRational(long);
}

overriding 覆載 前提 有父類別
overloading 多載 相同函式不同行為

繼承父類別後 直接在子類別重寫(重新定義) 可透過一樣的method 得到理想結果。

密碼 cipher key

終始功：以終為始

My Notes

Important Concepts worth keeping

Today: / /

ch2 抽象化

- isEmpty()
- getLength()
- insert()
- remove()
- retrieve

P.2.4 implemented by array list

```
(int) degree(){
    return aList.getLength()-1;
}
(int) coefficient(in (int) power){
    aList.retrieve(power+1, aCoefficient, success);
    if(success)  return aCoefficient;
    else         return 0;
}
```

```
changeCoefficient(in (AT) new, in (int) power){
    aList.remove(power+1, success);
    if(success)
      aList.insert(power+1, new, success);
}
```

C++ Namespaces 管理宣告

(::) scope 限定範圍

using declaration 使用命名空間

create

```
namespace x{
int count=0
void abc();
}
```

use

```
using namespace x;
count+=1;
abc();
```

C++ Exceptions 例外處理

try block 設保護

```
try
{
statement(s);   跑出問題才用
}
```

catch block 捕捉

```
catch ( ExceptionClass, identifier)
{
statement(s);
}
```

ex.

```
#include <std except>
#include <string>
using namespace std;
class LiseIndexOutOfRangeException
    public out_of_range;
{ public:
    LiseIndexOutOfRangeException (const string
                & message = "")
            :out_of_range(message.c_str())
        {}
};
```

```
try { ... throw(type); ... }
catch(type1){}
catch(type2){}
catch(...){}
```

跟switch類似

跑臉

"We recognize individual differences with respect to talents, character, capability, and background. We believe that full development of one's potential signifies success."

ch2 抽象化

My Questions
Problems & Difficulties needing exploration

Conclusion:

ADT = data + operation (Construct)

Step1. 目的: 想要運用 data 完成的事

Step2. 假設:

Step3. 輸入:

Step4. 輸出:

好的模組化: 高內聚(功能單一化), 低耦合(分工)

模組分為 描述, 實作(隱藏)

~~.h → ~~.cpp

建構 → create~() {}

解構 → destroy~() {}

插入 → insert(~~) {}

刪除 → remove(~~) {}

檢查 → retrieve(~~) {}

class combine

Attributes 靜態/屬性

Behaviors 動態/運算

3 characteristics

Encapsulation 封裝 隱藏細節

Inheritance 繼承 再利用

Polymorphism 多型 多工

父類別 ↕ 子類別 ~ 擴充

子類別繼承父類別的 public & protected

overloading (自動找符合參數的; 多載): 同樣函式, 不同行為

overriding 覆載: 將父類別之函式覆寫為想要之功能 — 同樣名字, 內容改寫

My Opinions
Thoughts, inspirations, and suggestions

Namespace (分類管理)

using namespace std (標準函式庫)

scope resolution operator (::)

⇒ 相同名稱, 不同範圍

Exception (方便除錯)

try block ~ 設定保護範圍

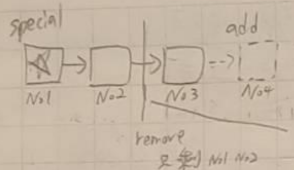
catch block ~ 接 try 所丟出的出錯

心得:

了解了抽象化概念, 可以更輕易的整理資料並且還了解了利用資料去建構想要的功能, 非常有趣。

Ch3 鏈結串列

Today: / /

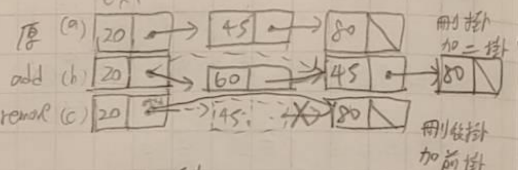


why? ∵ 記憶体有限

Array 陣列 v.s. Linked List 鏈結串列

- fixed size
- data must shifted
- doesn't require shifting

ex.



指標 (Pointers) = 門牌

contain location or address ⇒ 有門牌 沒房子

宣告 int *p; 住門牌
資料型態 (可為 class)
undefined but no Null

int x; address-of

① p = &x; &x = 房子x的門牌

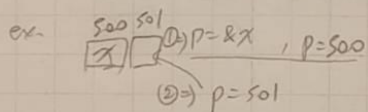
② p = new int; 申請新房子

互相搭配
- Dynamic allocation ⇒ 有彈性，自己管理記憶体
- If cannot allocate memory ⇒ exception std::bad_alloc in <new> header

delete p; 歸還房子

p = Null; 徹底遺忘門牌

ex.



Ex.

申請 (a) int *p, *q;
int x;

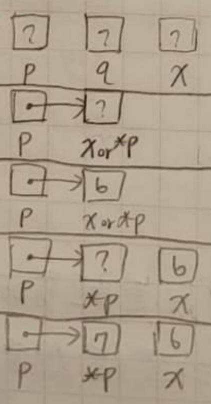
掛門牌 (b) p = &x;

b, c 不可對調!

(c) *p = 6;
搬往新家

重新配置 (d) p = new int;

掛新家號 (e) *p = 7;

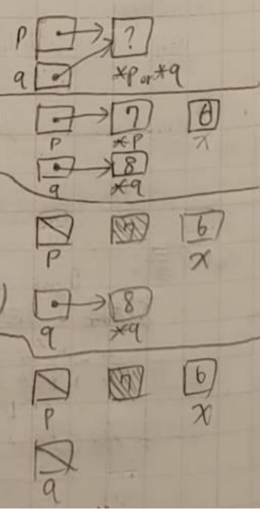


(f) q = p;

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

(h) p = Null;
memory leak
(會佔用記憶体)

(i) delete q;
q = Null;



Ch3 鏈結串列

My Questions
Problems & Difficulties needing exploration

My learning weather report

P 3-1

陣列名稱=指標

```
int *p, *q;
p = new int;
*p = 1
q = new int;
*q = 2;
*p = *q + 3
cout << *p << ", " << *q << endl;   (5, 2)
p = q;   memory leak
cout << *p << ", " << *q << endl;   (2, 2)
p = new int;
*p = 7
cout << *p << ", " << *q << endl;   (7, 2)
delete p;
cout << *p << ", " << *q << endl;   (?, 2)
p = Null;
q = Null;   memory leak
```

Dynamic Allocation of Arrays

```
int arraySize = 50;
double *anArray = new double [arraySize];
anArray[2] = *(anArray + 2)
```

增加陣列 {
```
double *oldArray = anArray;
anArray = new double [3*arraySize];
for (int index=0; index < arraySize; ++index)
    anArray[index] = oldArray[index];
delete [ ] oldArray;
```
配置更大空間

Example: Save/Copy a File

```
#include <iostream>
#include <string>
#include <cstdio>

#define SID_LEN 12
#define SR_NUM 5

using namespace std;

typedef struct student
{ char sid[SID_LEN];
  int score;
} studentType;

void saveFile (FILE *fp, studentType [], int);

int main (void)
{ FILE *outfile = Null;
  string fileName = "DSsample.dat";
  studentType allS[SR_NUM] = {
    {"10097113", 60}, {"10197102", 70}
    ... };
  outfile = fopen(fileName.c_str(), "a");   write file
  If (outfile != Null)
      saveFile (outfile, allS, SR_NUM);
  return 0;
}

void saveFile (FILE *fp, studentType dA[], int no)
{ for (int i=0; i<no; i++)
    { fwrite (&dA[i], sizeof(dA[i]), 1, fp);
      cout << dA[i].sid << " " << dA[i].score << endl;
    }
  fclose(fp);
}
```

Example: Save/Copy a file

```
void saveFile (FILE*, studentType[], int);
int main (void)
{ FILE *infile = Null, *outfile = Null;
  string fileName = "DSsample1.dat";
  int studentNo = 0;
  studentType *buf S;

  infile = fopen (fileName.c_str(), "r");
  if (infile != Null)
  { fseek (infile, 0, SEEK_END);
    studentNo = ftell(infile) / sizeof(studentType);
    rewind (infile);
    try {
      buf S = new studentType [studentNo];
      for (int i=0; i < studentNo; i++)
          fread (&buf S[i], sizeof(studentType), 1, infile);
      fileName = fileName.substr(0, 8) + "2.dat";
      outfile = fopen (fileName.c_str(), "a");
      if (outfile != Null)
          saveFile (outfile, buf S, studentNo);
      delete [] buf S;
    }
    catch (bad_alloc &ba)
    { cerr << endl << "bad_alloc caught: " << ba.what() << endl;
    }
    fclose (infile);
  }
return 0;
}
```

My Notes

Important Concepts worth keeping

Ch3 鏈結串列

Today: / /

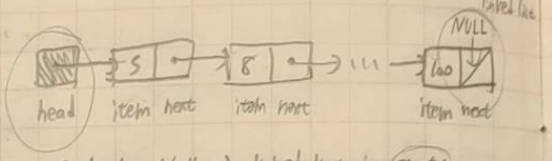
Pointer Based Linked Lists

a node in a linked list is usually a struct

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


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

The head pointer points to the first node in a linked list

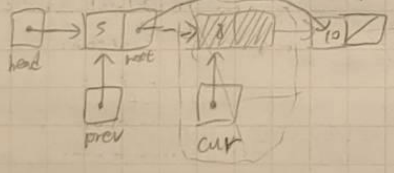


if head = Null => linked list is empty

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

走訪

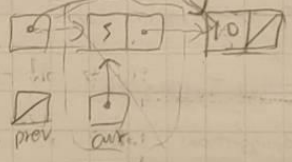
```
prev->next = cur->next;   or   prev->next = prev->next->next;
```



刪除

<special case>

delete head => head = cur->next;

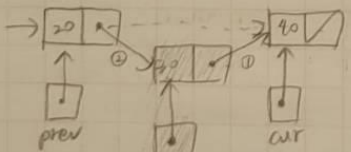



```
delete cur;
cur = NULL;
```

avoid dangling reference

---

① newPtr->next = cur;
② prev->next = newPtr;



新增

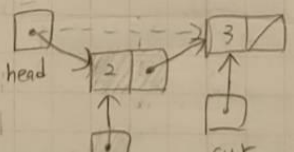
<special case>

add head

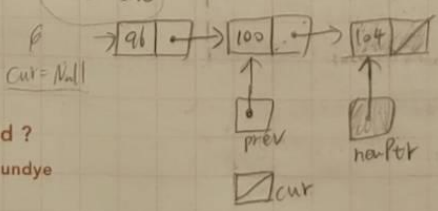
head = cur

newPtr->next = head;

head = newPtr;



add end



cur = Null

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

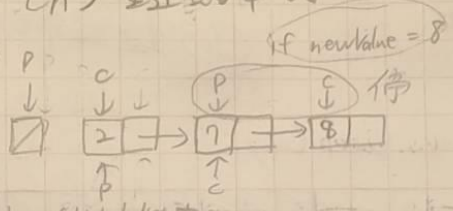
TEAM:
Together Everyone Argue and Mad ?
Together Each Achieve More ! —Sundye

Ch3 鏈結串列

My Questions
Problems & Difficulties needing exploration

My learning weather report

已排列的鏈結串列




```
Node *prev, *cur;
if (head != Null)
{for ( prev = Null , cur = head;          在Linked list內
      (cur ! Null) && (newValue > cur->item);
      prev = cur  , cur = cur->next );
  if(prev = NULL)
      head = cur->next                    delete
  else prev->next = cur->next
  cur->next = NULL;
  delete cur;
  cur = NULL;
}
```

```
if (prev = NULL)                          insert
  { newPtr->next = head;
    head = newPtr;
  }
else
  { newPtr->next = cur;
    prev->next = newPtr;
  }
```

A Pointer-Based implementation in ADT list

- ☐ Public method
  - isEmpty
  - getLength
  - insert
  - remove
  - retrieve
- ☐ Private method
  - find
- ☐ Private data members
  - head 串列首
  - size 節點數
- ☐ Local variables to methods
  - cur 現在節點
  - prev 前一節點

My Opinions
Thoughts, inspirations, and suggestions

```
/ListP.h/
#include "ListException.h"
#include "ListIndexOutOfRangeException.h"
typedef int ListItemType;
class List
{ public;                                  ①
  List ();                                 建
  List (const List& aList)                 copy
  ~List ();                                解
  bool isEmpty() const;                    是空
  int getLength() const;                   個數
  void insert (int index, const ListItemType & newItem)   插入
       throw (ListIndexOutOfRangeException, ListException);
```

```
刪除 void remove (int index)
       throw (ListIndexOutOfRangeException);
檢查 void retrieve (int index, ListItemType & dataItem) const
       throw (ListIndexOutOfRangeException);
private:
  struct ListNode
  { ListItemType item;
    ListNode *next;
  };
  int size;
  ListNode *head;
定位 ListNode *find (int index) const;
搜尋
};
```

```
List::~List()                              destructor
{ while (!isEmpty())
    remove(1);
}
```

密碼 cipher key

團隊成效可好可壞，運作之妙在於一心。

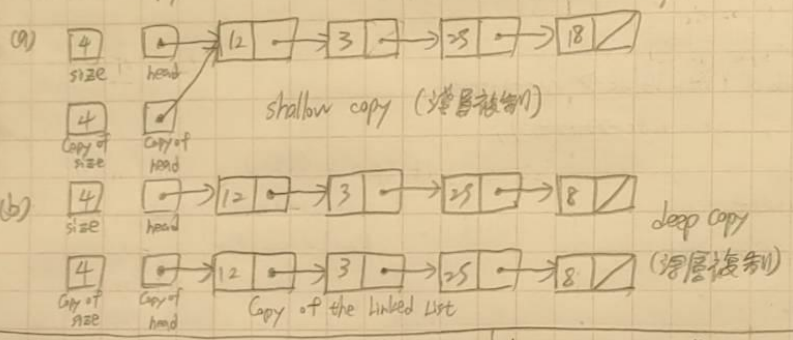
My Notes

Important Concepts worth keeping

Ch3 鏈結串列

Today: / /

Shallow Copy v.s. Deep Copy



靜態配置

```
// ListP.cpp
#include <cstddef>
#include <new>
#include "ListP.h"
using namespace std;

// 複制對象
List::List(const List &aList)
  : size(aList.size)                    // 複制建構
{ if (aList.head == NULL)
    head = NULL;
  else {
    head = new ListNode;
    head->item = aList.head->item;
    ListNode *newPtr = head;
    for (ListNode *origPtr = aList.head->next;
         origPtr != NULL; origPtr = origPtr->next)
    { newPtr->next = new ListNode;
      newPtr = newPtr->next;
      newPtr->item = origPtr->item;
    }
    newPtr->next = NULL;
  }
}

bool List::IsEmpty() const              // 是否空
{ return size == 0; }

int List::getLength() const             // 個數
{ return size; }

List::ListNode *List::find(int index) const
{ if ((index < 1) || (index > getLength()))
    return NULL;
  else                                  // 定位搜尋
  { ListNode *cur = head;
    for (int skip = 1; skip < index; ++skip)
      cur = cur->next;
    return cur;
  }
}

void List::retrieve(int index, ListItemType &dataItem) const
    throw (ListIndexOutOfRange Exception) -> LIOORE
{ if ((index < 1) || (index > getLength()))
    throw LIOORE("LIOORE: retrieve index out of range");
  else
  { ListNode *cur = find(index);        // 搜尋
    dataItem = cur->item;
  }
}

void List::insert(int index, const ListItemType &newItem)
    throw (LIOORE)
{ if ((index < 1) || (index > getLength()))   // int newLength = getLength()+1;
    throw ("LIOORE: Insert index out of range");
  else
  { try { ListNode *newPtr = new ListNode;    // 插入
      size = newLength;
      newPtr->item = newItem;
      if (index == 1)
      { newPtr->next = head;
        head = newPtr; }
      else
      { ListNode *prev = find(index-1);
        newPtr->next = prev->next;
        prev->next = newPtr; }
    } end try
```


Everything worthwhile is uphill. —John Maxwell

ch3 鏈結串列

My Questions
Problems & Difficulties needing exploration

```
void ListLL remove (int index)
    throw (LIOORE)
{ ListNode *cur;
  if ((index<1) || (index>getLength()))
    throw ("LIOORE: remove out of range");
  else
  { --size;                          刪除
    if(index==1){
        cur = head;
        head = head->next;}
    else {
        ListNode *prev = find (index-1);
        cur = prev->next;
        prev->next = cur->next;}
  cur->next = NULL;
  delete cur;
  cur = NULL;
  }
}
```

ADT

My learning weather report

Array-Based v.s Pointer-Based

	Array-Based	Pointer-Based
Size	固定	彈性
Storage (空間)	少	多
Retrieve	const (快)	depend on i (慢)
Insert / Delete	need shift 搬移資料 (慢)	need traversal 走訪 (快)

Save & restore LinkedList by File

only data not pointer

use (tail) pointer

My Opinions
Thoughts, inspirations, and suggestions

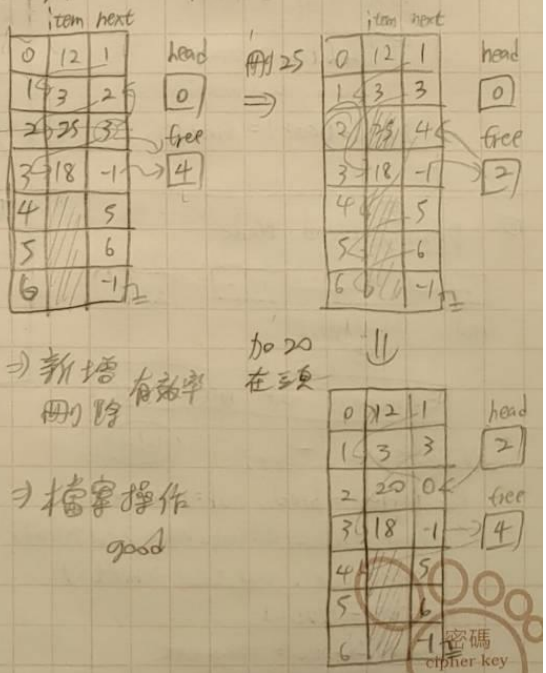
save (寫檔)

```
ofstream outFile(fileName);
for(Node *cur = head; cur!=NULL; cur=cur->next)
  outFile << cur->item << endl;
outFile.close();
```

(讀檔)

```
ifstream inFile(fileName);
int nextItem;
if( inFile >> nextItem)
 { try { head = new Node;
         head->item = nextItem;
         head->next = NULL;
         tail = head;
         while(inFile >> nextItem)
         {  tail->next = new Node;
            tail = tail->next;
            tail->item = nextItem;
            tail->next = NULL; }
infile.close();
```

Array-Based Implementation of LinkedList



凡是值得做的事，都是上坡路。
—約翰·愛斯威爾

My Notes

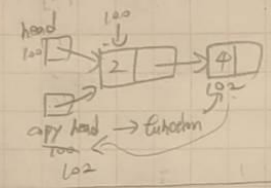
Important Concepts worth keeping

Ch3 鏈結串列

Today: / /

Pass a Linked List to a Method

傳址&：更改指標內容

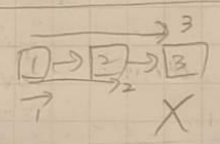


Licked Lise Recursively

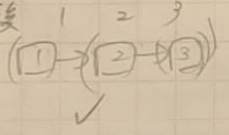
① 遍迴前 印

② 遍迴後 印

① 前



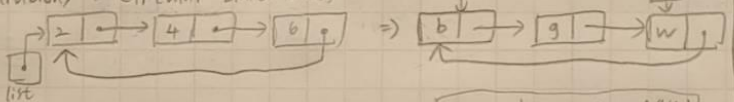
② 後



操作第一個較方便

Varations : Circular Linked Lists

①



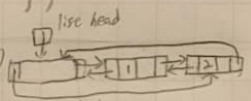

```
if (list != NULL)
{ Node *first = list->next
  Node *cur = first;
  do {
      show (cur->item);
      cur = cur->next;
  } while (cur != first);
}
```

no node contains NULL

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

② Doubly Linked List

雙向



★3個不能空，可結合

delete

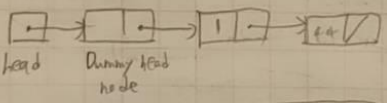
好用但佔優為空間

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

Insert

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

② Dummy Head Node



Eliminates the special case

delete

```
Node *prev, *cur;
for (prev = head, cur=prev->next;
    (cur != NULL) && (newValue > cur->item);
    prev = cur, cur = cur->next;)
if (cur != NULL)
{ prev->next = cur->next
  cur->next = NULL;
  delete cur;
  cur = NULL; }
```

insert

```
for (prev = head, cur = prev->next;
    (cur != NULL) && (newValue > cur->item);
    prev = cur, cur = cur->next;)
newPtr->next = cur;
prev->next = newPtr;
```


Habit is habit, and not to be flung out of the window by any man, but coaxed downstairs a step a time. —Mark Twain

Ch3 鏈結串列

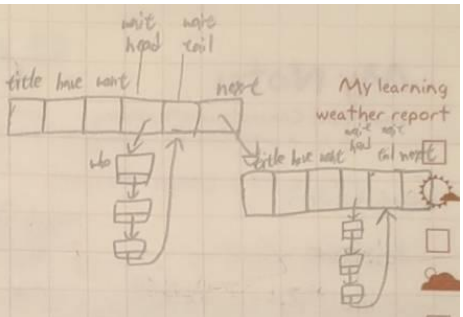
My Questions

Problems & Difficulties needing exploration

DVD store

```
struct waitNode
{ string  who;
  WaitNode *next;
}
```

```
struct StackNode{
{ string  title;
  int     have, want;
  WaitNode *waitHead, *waitTail;
  StackNode *next;
}
```



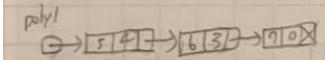
=> Commands : I L A M D O R S

I	L	A	M	D	O	R	S
查詢	列表	新增	修改	進貨	退貨	售出	

=> Operation : List (L) 列表
Find (I, M, D, O, S) 搜尋
Replace (M, D, R, S) 置換
Insert (A, D) 插入

P3.2 Polynomial addition

$5x^4+6x^3+7$




```
struct Node
{ double c;
  int    p;
  Node  *next;
}
```

```
Node * addPoly (Node *x, Node *y)
{ Node *z = NULL, *w = NULL;
  if ① x or y is empty
  else{
    ② a dummy head node
    do{
      ③ create node
      ④ copy p & c, then find next
    }while (x!=NULL) && (y!=NULL);
    ⑤ remaining x or y }
  return z;
}
```

① if (x = NULL)
  z = copyList(y);
else if (y = NULL)
  z = copyList(x);

② w = new Node
  z = w

③ z → next = new Node;
  z = z → next;

④
```
If (x→p >= y→p){
  z→p = x→p
  if (x→p = y→p){
    z→c = x→c + y→c;
    y = y→next;
  } else z→c = x→c
  x = x→next;
}ELSE{
  z→p = y→p;
  z→c = y→c;
  y = y→next;
}
```

⑤
```
If (x = NULL)
  z→next = copyList(y);
Else if (y = NULL)
  z→next = copyList(x);
z = w→next;
w→next = NULL;
delete w;
w = NULL;
```

My Opinions

Thoughts, inspirations, and suggestions

密碼 cipher key

習慣就是習慣，
誰也不能將其扔出窗外，
只能一步一步地引下樓。
—馬克・吐溫

My Notes

Important Concepts worth keeping

(Ch3 鏈結串列)

Today: / /

Conclusion:

Pointer

int *p; 創指標

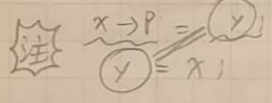
p = &x; 被創x位置

p = new int 申請空間

delete p; 歸還空間

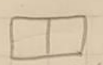
p = NULL; 刪指標

不可對調否則 memory leak!



不可對調
↑左右有相同表示

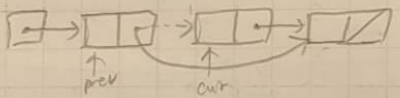
struct Node
{ int item;
Node *next; }



Add

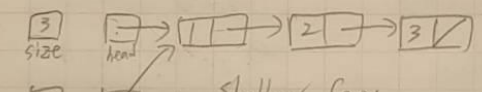
newPtr->next = cur; prev->next = newPtr;

Delete

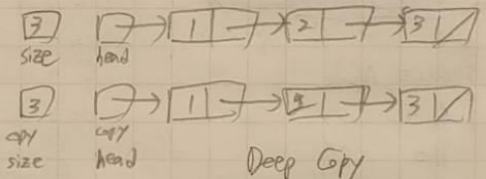


prev->next = cur->next;

Shallow Copy vs Deep Copy



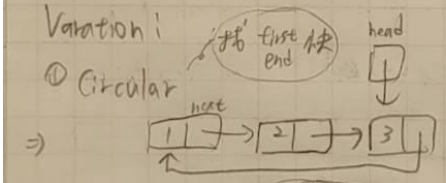
Shallow Copy (靜態)



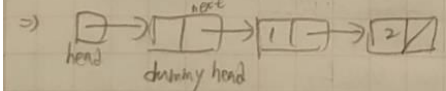
Deep Copy (動態)

Variation:

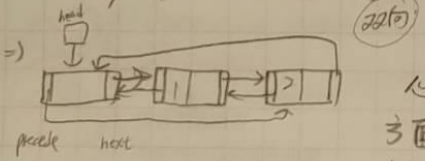
① Circular 找 first end 快



② Dummy head insert delete 方便

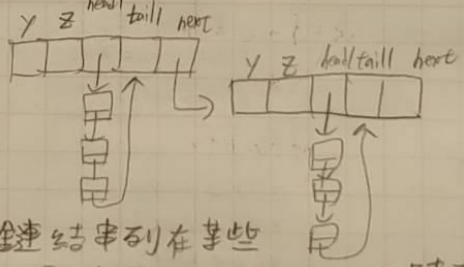


③ Doubly (可結合以上兩者)




```
struct Node1{
  string x;
  Node1 *next;
}
```

```
struct Node2{
  string y;
  int z;
  Node1 *head *tail;
  Node2 *next;
}
```



心得: 鏈結串列在某些方面特別好用, 像是新增, 刪除就比陣列好用但缺點是每次都從頭讀, 是一個新鮮的儲存方式。

Ch4 遍迴解題

My Questions
Problems & Difficulties needing exploration

My learning weather report

Defining Languages

(L) 語言: a set of string of symbols.

(G) 文法: rule for forming the string in a Language.

basics = x|y (x or y) 或
- xy or x·y (x followed by y) 緊接著

ex. C++ programes = {string w, w is a syntactically correct C++ program}

or 代換

<word> any instance of word

ex. <number> = <digit><number> | <digit>

<addition> = <number> + <addition> | <number>

C++ identifier begin with a letter & followed by 0 or ∞ letters + digits

<letter> = a|b|...|A|B|...|Z

<digit> = 0|1|...|9

(L) C++ Ids = {w: w is a legal C++ identifier}

(G) <identifier> = base ① <letter> | <identifier> ② <letter> | <identifier> ③ <digit>

ex. A2C3 → ③3 → ②C → ①2 → ①A

L ↓ G ↓ R

recognition algorithm (辨識演算法)

isId (in w: string): boolean

if (w is a letter)
  return true;
else return false;

else if (the last character of w is letter or digit)
  return isId (w minus its last character);
else return false

③ ◦ recognition algorithm

isPal (in w: string): boolean

if (w = empty or w length = 1)
  return true;
else if (w.st = w.end)
  return isPal (-w-)
else
  return false;

-w- = 去頭去尾

遞迴

My Opinions
Thoughts, inspirations, and suggestions

Two Simple Languages: Palindromes

迴文: anna, Bob, 121, 4884, 我治人人.人人治我

① ◦ Language

Palindromes = {w: w reads the same left to right as right to left}

② ◦ Grammer 偶數 base 奇數 base

<pal> = empty string | <ch> |
a<pal>a | b<pal>b | ... | Z<pal>Z

<ch> = a|b|...|A|B|...|Z

Ch4 遍迴解題

Today: / /

A^nB^n

$L = \{w : w \text{ is the of the form } A^nB^n \text{ for some } n>0\}$

$G = \langle legal\text{-}word \rangle = empty\ string \mid A\langle legal\text{-}word \rangle B$

```
R => isAnBn (in w: string): boolean
    if ( w.length =0)
      reeurn true;
    else if ( AwB )
      reeurn (isAnBn(_w_))
    else
      return false;
```

AwB = w start with A, end with B

_w_ = 去頭去尾

遍迴

P4.1

G: <S> = <L> | <D><S><S>
<L> = A | B
<D> = 1 | 2

(a) 三个字元字串
1AA 1AB 1BA 1BB
2AA 2AB 2BA 2BB

(b) 三个以上 => ∞
ex. 12AAB

P4.2

寫一遍迴之法(至少1字母, 1st大寫, 其餘小寫)

<S> = <U> | <S><L>
<U> = A|B|...|Z
<L> = a|b|...|z

Algebraic Expressions

- Infix expressions 中序運算式
  ex, a+b
- Prefix expression 前序、、
  ex, +ab
- Postfix expression 後序、、
  ex, ab+

須括弧

((a+b)x c)/d

+ a b

x (+ a b) c d

生

((a b + c) x d) /

生

- 前, 後序
  (优) 不用優先 可判讀
  不用結合 求解
  不用括弧

- G <infix> = <identifier> | <infix><operator><infix>
  <operator> = +|-|x|/
  <identifier> = a|b|...|z

中 → 前
最裡往最左放

中 → 後
最裡往右一格放

The amount of time you have been performing a habit is not as important as the number of the times you have performed it. «Atomic Habits»

ch4 課題解題

My Questions
Problems & Difficulties needing exploration

My learning weather report

CFG = <prefix> = <identifier> |
<operator> <prefix> <prefix>
<operator> = + | - | * | /
<identifier> = a | b | ... | z

{ Base Case letter is prefix (exp)
Recursive <operator> <prefix> <prefix> } 終止條件

擷查: endPre(in first: int): int
last = str.Exp.length() - 1;
if (first < 0) or (last < first)
return -1;
ch = strExp[first];
if (ch is identifier) ← base case
return first;
else if (ch is operator)
{ firstEnd = endPre(first+1)
if (firstEnd > -1)
return endPre(firstEnd+1);
else return -1;
}
else return -1;

重要特性
prefix + string ≠ NULL
一定非 prefix

ex. (+ *ab/cd - fg) / z ⇒ 非prefix
x y w

R = isPre(): boolean
lastChar = endPre(0);
return (lastChar >= 0 &
lastChar = strExp.length() - 1;

+ x ab / cd
① → ⊗ ← fE=a return b
1.op 2.op
→ ① ← fE=c return d
3.op
first = last

My Opinions
Thoughts, inspirations, and suggestions

評估
evaluatePrefix(inout strExp: string): float
ch = strExp.first;
- strExp;
if (ch = identifier)
return (identifier)
else if (ch = operator (op))
{ op1 = evaluatePrefix(strExp)
op2 = evaluatePrefix(strExp)
return op1 op op2
}

Ex a=2, b=3, c=4
* + ab c
↳ op1 = + = 5 → op1 = a = 2, op2 = b = 3
op2 = c = 4 ⇒ 20

+ a * bc
↳ op1 = a = 2
op2 = * (12) → op1 = b = 3, op2 = c = 4
⇒ 14

My Notes

Important Concepts worth keeping

Ch4 遞迴解題

Today: / /

G = <postfix> = <identifier> |
<postfix> <postfix> <operator>
<operator> = +|-|*|/
<identifier> = a|b|...|z

前序 → 後序

postfix(exp) = postfix(prefix1)
+postfix(prefix2) + <operator>

前轉後

```
convert (inout pre:string, out post:string)
  ch = pre.first
  -pre;
  if (ch is letter)
    post = post + ch
  else
  { convert (pre, pose)
    convert (pre, post)
    post = post + ch
  }
```

Ex ⊛ + abc

⇒ (+ab, ⊛) abc

(c, c) c

⇒ ab+c*

Backtracking

can be combined with recursion

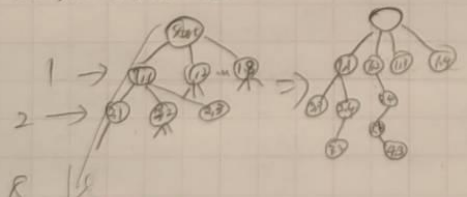
- Base case 8個皇后
- Recursive step
  if (success) continue
  if (false) backtrack

Ex. eight queen problem

4,426,165,368 種方法

但每列/行之有一皇后 ⇒ 8! = 40320 種

由左至右每欄一個



1 →
2 →
8

- board (棋盤)
  存位置
- queen (皇后)
  存位置
  判斷衝突

Ex Maze, Number Maze

暫停吸口氣，
靜思問自己，
關鍵 10 秒鐘，
Ho 住十年功。

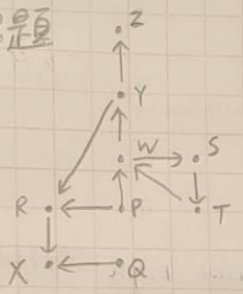
Ch4 遞迴解題

My Questions
Problems & Difficulties needing exploration

Application : A search Problem
- 航班路徑
- 航線地圖

data

- base case 抵達
- 沒航班 backtrack
- 重複



My learning weather report

```
+ searchR (in originCity: City , in destinationCity: City): boolean
   Mark originCity as visited
   if (originCity is destinationCity)
      Terminate ┐ the destination is reached 抵達
   else
      for (each unvisited city ( adjacent to originCity)
         searchR (c, destinationCity)

bool Map:: isPath (int originCity, int destinationCity)
  { int nextCity;
    bool success, done;
    markVisited(originCity);
    if(originCity = destinationCity)
       return true;
    else { done = false;
           success = getNextCity(originCity, nextCity);
           while (success && !done)
           { done = isPath (nextCity, destinationCity)
             if(!done)
                success = getNextCity (originCity, nextCity);
           }
    return done;
  }
```

```
while (success && !done)
{ done = isPath (nextCity, destinationCity)
  if (!done)
     success = getNextCity (originCity
                            nextCity);
}
}
}
```

My Opinions
Thoughts, inspirations, and suggestions

My Notes

Important Concepts worth keeping

ch4 遞迴解題

Today: / /

數學歸納法

① Prove recursion right (特性)

② Prove recursion time (工作量)

①

```
if (n is 0)
    return 1
else
    return n * fact(n-1);
```

- fact(0) = 0! = 1 最簡假設
- fact(n) = n! = n×(n-1)×...×1 if n>0
- fact(n+1) = (n+1) × fact(n) 歸納
  = (n+1)×n! = (n+1)×n×(n-1)×...×1

If E is prefix, then EY is not prefix $\forall(Y \neq \phi)$

最簡 (Basis) $|E|=1$: E = single letter → E not op → EY NOT prefix

假設 (Inductive hypothesis) $1<|E|<n$: if E is prefix, then EY is not prefix

歸納 (Inductive Step) $|E|=n$: $E = op, P_1, P_2$, P_1 = prefix, P_2 = prefix, $|P_1|<n$, $|P_2|<n$

反證法 (Assume) EY is prefix

then $EY = op\ W_1\ W_2$, W_1 = Prefix, W_2 = Prefix

$P_1 = W_1$

$P_2Y = W_2$ → contradiction → EY is not prefix.

② Case of Towers of Hanoi

```
solveTowers( count, source, destination, auxiliary)
                 n       s            d            a
  if(count=1)
    move disk from s to d
  else
    { solveTowers(count-1, s, a, d)
      solveTowers( 1 , s , d , a)
      solveTowers(count-1 , a , d , s)
    }
```

工作量

when N=1 - moves(1) = 1

when N>1 - moves(N) = moves(N-1) + moves(1) + moves(N-1)

$$\Rightarrow \begin{cases} -\ moves(1)=1 \\ -\ moves(N)=2\times moves(N-1)+1 \text{ , if } N>1 \end{cases}$$

⇒公式 $moves(N)=2^N-1$, for all $N \geq 1$

⇒推導

moves(1) = 1

moves(2) = 2 × moves(1) + 1

moves(N-1) = 2 × moves(N-2) + 1

moves(N) = 2 × moves(N-1) + 1

$= 2\times 2\ moves(N-2)+1+1$

$= 2^2 \times moves(N-2)+2^1+1$

$= 2^{N-1} \times moves(1) + 2^{N-2}+2^1+2^0$

$= 2^{N-1}+2^{N-2}+2^{N-3}\cdots+1$

$= 2^N-1$

⇒歸納: basis $N=1$: moves(1) = 1

inductive hypothesis : $1<N<k$ $moves(N)=2^N-1$

inductive Step $N=k$: $moves(k) = 2^* moves(k-1)+1$

$= 2^* 2^{k-1}-1+1$

$= 2^k-1$

一步一步再一步，
山窮水盡疑無路，
一寸一寸又一寸，
柳暗花明又一村。

My Questions
Problems & Difficulties needing exploration

ch4 遍迴解題

My learning weather report

Conclusion:

語言(L) ⇒ 字串的集合 (輸入) (輸入是否符合條件)

語法(G) ⇒ 可以處理語言 (理解程式用法)

辨識演算法(R) ⇒ 簡化語法

只適用遍迴

Infix 中序 ← 須掃描

Prefix 前序 } 不用優先 不用結合 不用掃描 → 可辨識 我解

Postfix 後序 }

中→前 往左掃5放

中→後 往右掃5放

prefix 特別車

prefix + string = prefix

Backtrack

當答案有限，搜尋範圍大 ⇒ 遍迴 + 回朔 到錯的時

My Opinions
Thoughts, inspirations, and suggestions

數學歸納法 可驗證

① 特性

② 工作量

心得:

我覺得語法就像解題的題示，看懂後就會更容易描由程式完成想達成的事，還學到了不同的運算式去適應不同的程式，還有驗證的方法，非常有用。