臺灣二 OO 三年國際科學展覽會

科　　別：電腦科學科

作品名稱：AI 演化技術

得獎獎項：電腦科學科佳作

學　　校：臺北市立建國高級中學

作　　者：莊偉赳、徐茂芳

作 者 簡 介

< 莊偉赳 >

目前就讀北市建國高級中學三年級，從小研讀電腦科學相關領域資料，較感興趣的部分在電腦網路管理、網路網頁程式撰寫以及人工智慧演算法。

就讀國中時曾負責維護北市明德國中學校網頁。曾任建國中學電子計算機研習社副社長，並曾擔任社刊主編。曾任建國中學網路管理人員管理校園網路。

去年(2002)參加台灣區國際科學展覽會電腦科學類，獲得一等獎，並獲推薦參加當年度紐西蘭科學展覽會。

在紐西蘭科學展覽會中獲得 3rd place International Exhibit of Excellence

< 徐茂芳 >

目前就讀於北市建國中學二年級，平常無其他嗜好，打球、電腦就是一整天的生活，常埋身於網路的世界裡，曾與同好共同創立一個 MUD，現已卸職，平常多寫演算法還有網路應用程式，曾入選北市資訊能力競賽。對於架設伺服器還有各種不同新奇伺服器都有濃厚的興趣，目前擔任建國中學網管人員。

摘要

此研究之重點在於如何建構一套人工智慧方法，人工智慧含多種進行方式，例如以類神經網路訓練近似於人腦之結構，而專家系統係於不同的科學領域內以自己之所知判斷。我們先以 John Holland 的 Genetic Algorithm （暫譯基因演算法，以下暫稱 GA）的理論來實作出一套人工智慧系統之方法。

Genetic Programming 係以G A爲基礎之實作方法，主要的內容不出基本的演化定義；在這次的試驗中，首要爲先定出程式欲演化成何種類型，在此我們先定義爲排序型的演算法，經過分析實際程式之結構後定出適合基因元件的資料結構，分化爲兩個部份進行，一爲產生器，亦爲突變，交配器，一爲評審程式，亦爲執行器，兩者使用相同之基本元件，再以不斷的交配和突變以達到全域最佳化。我們將兩種部份完成後，加上現有的 PVM 分散運算函式庫來增加演算的速度。

Abstract

The research mainly discusses AI. AI, involves in several types, for example, neural network(NN), which adopts human-like training method; Expert systems determine and make decision by what it has known. We will use John Koza's Genetic Programming theory to implement an AI system.

Genetic Programming theory is based on GA. In our experiments, we have to define what kind of applications we want : a sorting application, which is divided into two parts - a producer (along with crossover and mutation operations) and an executor (along with judging operations), is an easy-implemented algorithm. Our program, also with PVM, will approaches global optimal after evolutions.

壹、前言

一、動機：

遺傳演算法是生物科技應用於資訊科技之一種實例。期盼以加強之遺傳演算法產生更有效率的演算法，用以增進程式效率；遺傳演算法在程式演化之方面確實非常吸引人，因爲如何能讓一個死的程式可以自行演化並且達到我們想要之結果相當不容易，亦由於前人之研究中尚有未完臻之處，希望能由本研究中完成改進。

二、目的：

利用基因遺傳演算法來，將已預設好之基本程式模型演化成第一種基本類型(排序演算法)，並將其效率調校至可以達到最佳化範圍之最好狀態。並且期待能再以此種模式爲架構，做出具棋奕思考能力之程式。

貳、研究方法及過程

一、理論基礎：

本實驗建構於基因演算法（Genetic Algorithm）之相關理論上，它是由 John Holland 在 1975 年出版的著作“Adoption in Nature and Artificial Systems” 上所提出的想法：基因演算法是一種模仿大自然界中，物競天擇法則和基因交配之演算方式。一些傳統人工智慧方法無法有效解決之計算問題，都可以很快速地找出答案來。且基因演算法是一種特殊之搜尋技巧，適合處理多變數非線性問題。

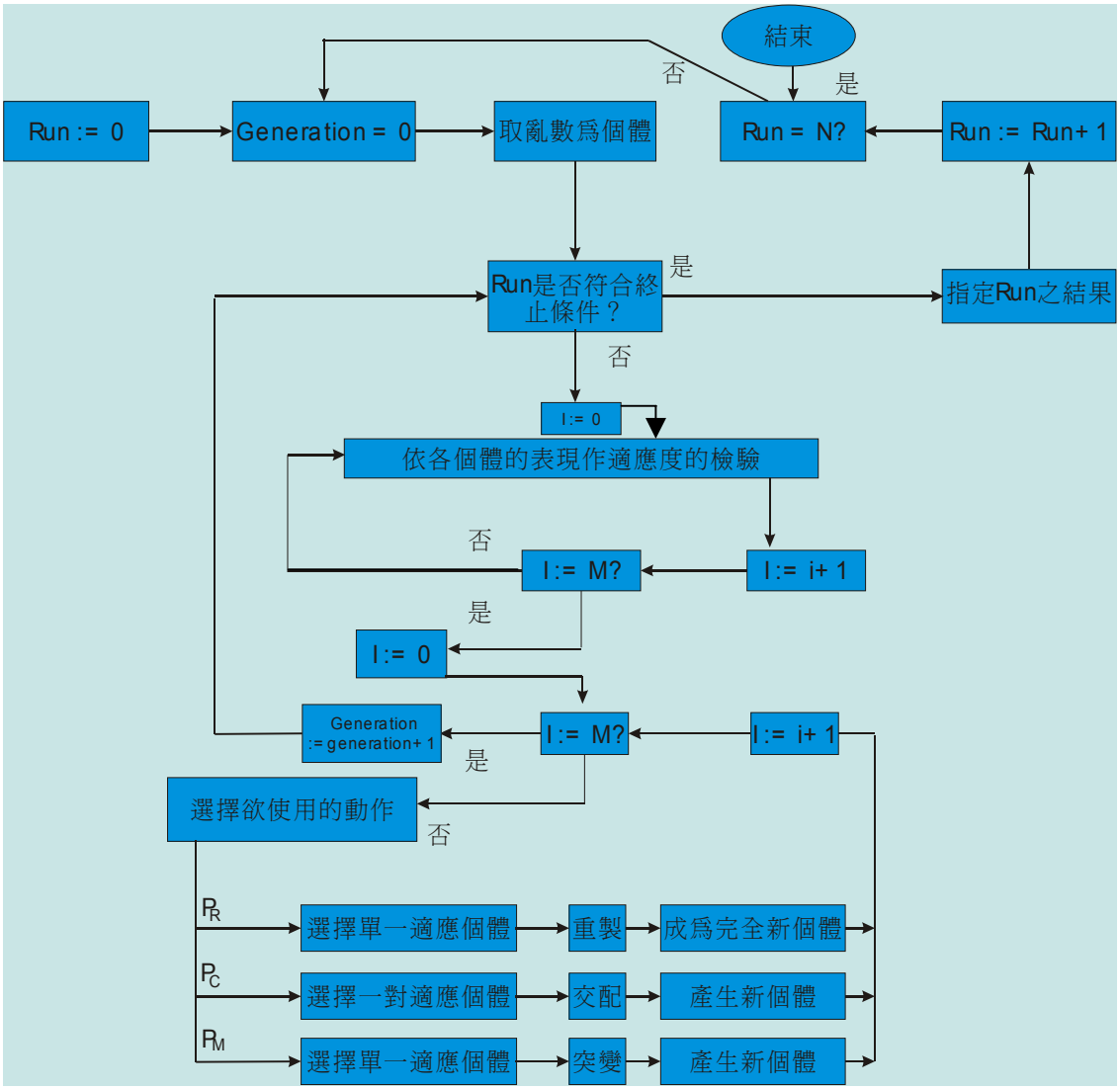
實作之基本動作是由交配（Crossover）、突變（Mutation）、重製（Reproduction）所組成，依序所做的動作分別是

1.由亂數產生一個新個體，定義爲世代 0。

2.計算每一個世代中所有個體之適應度。

3.任選二或一個適應度達標準之個體做基因階層的動作，也就是交配、突變或重製，其中，交配又可以稱做重合(Recombination) 。

4.進行下一子代之動作。



圖表一、基礎 GA 的流程

二、程式架構：

由於 GA 理論可以平行化（PGA, Parallel Genetic Algorithm），所以程式中應用到 PVM 函式庫分散計算量。以 PVM 函式庫連結各個節點形成小型叢集（cluster），並利用基因演算法，整個程式更具彈性，若要增加運算速度，僅需將機器加到 PVM host list 中分攤運算量。

另外，實作中將程式分爲幾個部份:

judge()	執行產生出來程式的部份，然後依據執行這個程式產生的輸出給分數
generator()	利用 GA 產生程式碼出來，經一次次的交換變異選擇後產生新子程式

程式中採用 server-client 模式，將 generator 視爲 server；將產生出來之資料傳送給 judge 程式—client 群運算，最後將結果傳回 generator 做總整理。

三、基因語言語法的定義／實作：

一個好的資料結構在這裡是很重要的，我們完成的部份包括產生語法之基本單位，以及一個 Virtual memory—姑且稱之爲 Vmem 好了，用在程式中記錄變數之使用，並且可以增加擴充性。總而言之，就是想要達到一般程式語言可以做到的事，但又要避免許多可以去掉之負擔，所以使用 structure linklist 來實作出基本語言

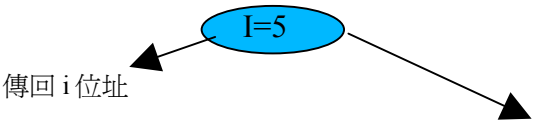
之語法，一個單位結構中包含了函式指標、各個基本語法單位需要之資料；在執行期間會直接由 generator 亂數產生內容，不致限制了程式之架構。

```
struct abc {
  int nElement;
  int element[3];
  struct abc (*pfunc)();
`
```

先將要實作之演算法做個簡單分析，程式的架構不出流程控制（if、switch 等等）、巢狀迴圈（while、for），因此我們將爲演化所需要之架構單位，實作出 for、compare（也就是 if 條件式的功能）、swap 和 break 功能，但是爲了在實際寫程式中之便利性，加上了 next，用在表示一個巢狀迴圈之結束；它的位置並不一定會伴隨著 for、compare 出現，也是爲了讓隨機產生之架構可以有更大的彈性使用空間，架構完成之後，緊接著就是其他細微部份之補充。

四、Vmem 的使用：

在這個程式產生器中，爲了使程式中使用之變數等更加靈活而且可以增加功能，使用了一個自己寫的虛擬記憶體部份。當程式執行時要求一個變數紀錄資料（例如計數器）就會傳回這個變數在 Vmem 中之位址。



2003	1923	0	…….	記憶體內容
used	used	empty	……	記憶體狀態

圖表二. 虛擬記憶體

五、模擬 compiler 動作：

用基本單位產生小巨集，再利用產生出來的小巨集構成一個完整之程式，或者是由小巨集結合成大巨集，也就是副程式之功能。

六、目標：

先列出目標，然後根據這個目標來進行。第一個當然是正確性，其次是時間之消耗。正確性很容易判斷，但是時間的消耗卻是個問題；產生出來的程式最後要產生排序資料來做比對，選擇出結果較正確者。

自行編寫一個直譯器之好處在於速度較快。若否，假如是產生程式語言之原始碼再呼叫編譯器編譯，不但很可能編不過，而且還得注意一些語法之邏輯問題，最重要的是速度較慢，用直譯器可以明確的表示自己產生出來之程式，程式之 exception 問題也比較容易偵測出來。

要寫一個 GP 程式，必須明確指出輸入以及輸出，在這裡，輸入（評審程式）就是隨意產生之一串數字，輸出就是將它排序完之順序；它的生死決定於輸出之正確性。

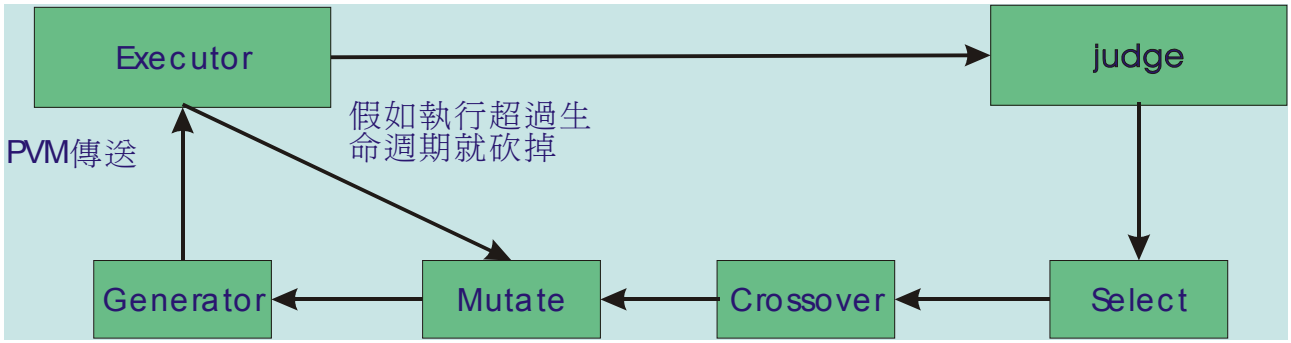
參、研究結果與討論

```
(( do_start->do_next->(ERROR!!)do_next->(ERROR!!)do_for->do_swap->do_swap->do_for-
>do_compare->do_compare->do_swap->do_swap->do_compare->do_swap->do_next->do_for-
>do_compare->do_compare->do_swap->do_compare->do_for->do_for->do_stop ))
(( do_start->do_for->do_compare->do_compare->do_next->do_compare->do_swap->do_compare-
>do_swap->do_swap->do_next->do_swap->do_compare->do_for->do_for->do_next->do_next-
>do_compare->do_next->do_for->do_swap->do_stop ))
(( do_start->do_next->(ERROR!!)do_compare->do_compare->do_compare->do_next->do_for-
>do_swap->do_swap->do_for->do_compare->do_for->do_for->do_compare->do_for->do_for-
>do_for->do_for->do_for->do_for->do_next->do_stop ))
(( do_start->do_compare->do_swap->do_swap->do_for->do_next->do_compare->do_compare-
>do_swap->do_compare->do_compare->do_next->do_for->do_swap->do_next->do_swap-
>do_compare->do_for->do_swap->do_compare->do_swap->do_stop ))
```

圖表三、是我們的偵錯過程，產生出來之順序很多都是不符邏輯結構，Judge 執行時將直接拋棄有問題的順序。

```
pvm> conf
conf
4 hosts, 1 data format
        HOST       DTID       ARCH    SPEED          DSIG
      Dwarf      40000      LINUX     1000 0x00408841
       lab1      80000      LINUX     1000 0x00408841
       lab2      c0000      LINUX     1000 0x00408841
      koala     100000      LINUX     1000 0x00408841
```

圖表四、在 PVM 中各連結起的 hosts PVM 在我們的程式中大量使用，因爲它可以做到產生新子程序、送出 POSIX signal 等等



圖表五、執行步驟

我們的研究是想用 GP 的強大搜尋能力找出一個最好的結果，目前已經有了答案，但經更進一步的研究，發現還有很多發展空間。

GA 理論應用到大量的亂數，但電腦中的亂數是假亂數，必要時需寫出更亂的亂數產生器來加強此套系統。

我們的研究在於加強之前的研究成果，並且加上其他較爲複雜之部份。

前人之研究已指出『要找出「產生最佳解、及避免陷入無窮迴圈」的方法是無解問題』，不過似乎可以透過讓程式”自然死亡”以改善情況，以下是我們的構想：

爲了防止產生的程式執行出無窮迴圈之情況，有些時候必須在超過時間限制時砍掉跑太久之程式，然而又不知道怎麼樣的限制是適當的，於是嘗試著讓每個產生出來的『程式』都有自己不定之生命週期，時間一到就 terminate 掉，也就是結束程式。雖然就機率而言，最佳解很有可能會因爲生命週期太短而無法完成工作，被判定爲不成功，但就整體而言，由於無窮迴圈之程式會在生命週期完成後被關閉，所以將產生無窮迴圈執行的時間拿來產生其他較可能解，反而是較有利的。

肆、結論與應用

本實驗之目的是想利用 GA 強大搜尋能力發展出一套可以應用之演算法，但是根據我們現在所完成的結果，發現若要做的更好，仍需要許多其他演算法配合。根據文獻上記載，NN、fuzzy、GA 這幾種理論是可以互相組合使用，透過這些 AI 相關領域之演算法或許可以加快執行速度。經研究後，GA 的確能加快進行速度，然而並沒有辦法讓電腦完完全全的將 CPU 整個使用率投入，所以此方面還有發展空間。

本專題牽涉到許多專業領域之探討，而由於此部份資料尚有發展空間，故未來我們將持續進行本計畫領域相關問題之研究，以期能用更好的方法來提升此研究主題。

伍、參考文獻

一、吳身澗，「人工智慧程式設計」， 台灣， 維科圖書有限公司， 2002

二、潘正君、康立山、陳毓屏，「演化計算」， 中華人民共和國，清華大學出版社，2000

三、蘇木春、張孝德，
「機器學習-類神經網路、模糊系統以及基因演算法則」，二版，台灣，全華科技圖書股份有限公司，2000

四、Al Geist & Adam Beguelin & Jack Dongarra & Weicheng Jiang & Robert Manchek & Vaidy Sundream,PVM A User's Guide and Tutorial for Networked Parallel Computing,USA , Massachusetts Institute of Technology , 1994.

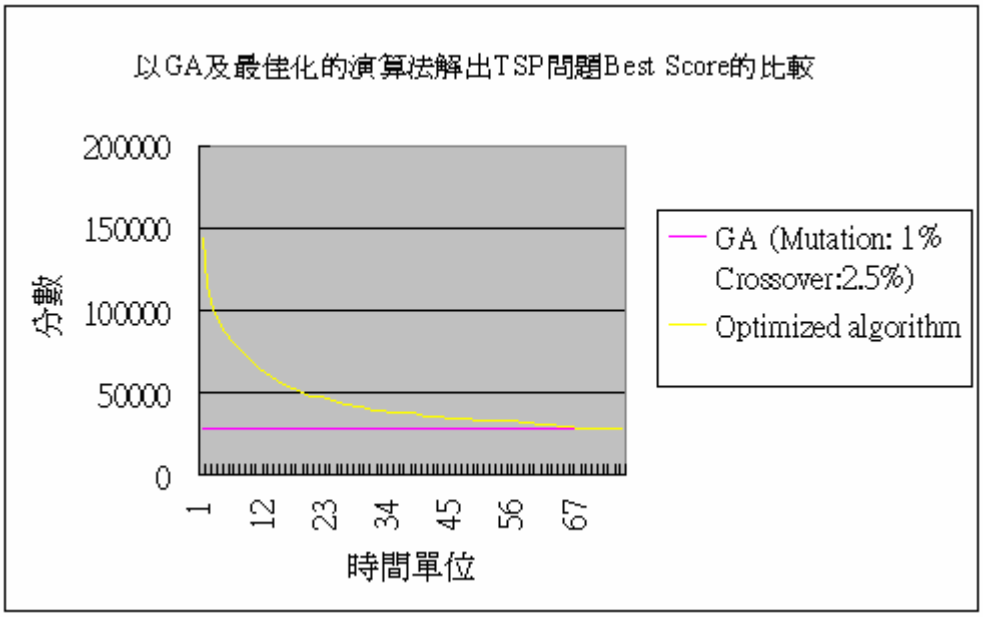
五、George F Luger & William A Stubblefield., Artificial Intelligence 3Rd, America, Addison-Wesley, 1999.

六、Kai Kwang & Zhiwei Xu,Scalable Paralel Computing, Singapore, McGraw-Hill Companies Inc, 1998.

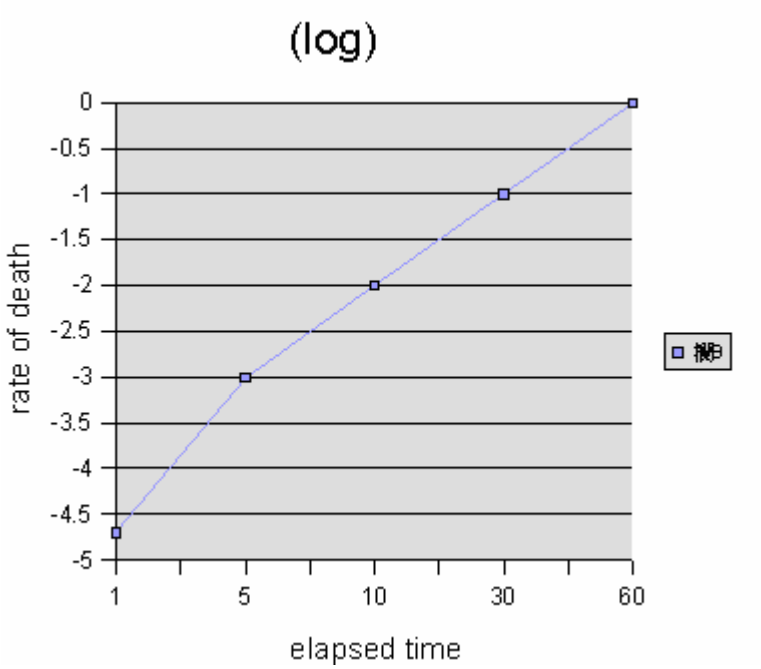
七、Nils J. Nilsson, Artificial Intelligence A New Synthesis, America, Morgan Kaufmann, 1998.

附件:內容包含如下資料

1. 在正文中提到之運算結果所顯示的圖表
   - 以 GA 及最佳化的演算法解出 TSP 問題 Best Score 的比較
   - 死亡率與時間關係
   - 最小值示意圖
   - pseudo random
2. GP 程式範例 GPBubble； 用 GP 演算法寫 bubble sort
3. TTT_ab； 以 alpha-beta search 寫 TicTacToe 遊戲
4. GP 程式範例 GPTicTacToe；發展井字遊戲的程式



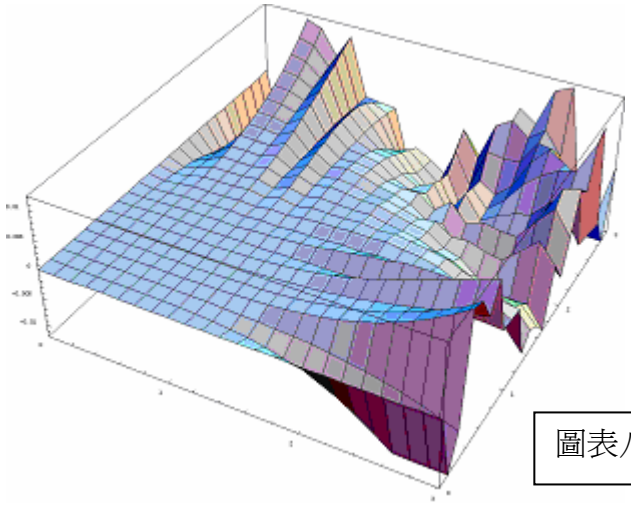
圖表六、解 TSP 問題



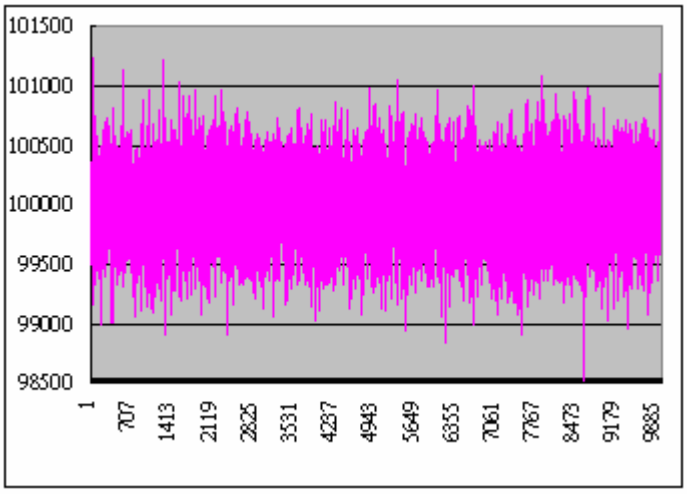
圖表七、死亡率與時間關係

程式中將需使用
1 秒鐘的基因程式之死亡率設爲 5/10000
5 秒鐘的基因程式之死亡率設爲 1/1000
10 秒鐘的基因程式之死亡率設爲 1/100
30 秒鐘的基因程式之死亡率設爲 1/10
60 秒鐘的基因程式之死亡率設爲 1
左圖縱座標以 log10 來表示死亡率

尋找最小值時常常僅會搜到區域間最小值，然而 GA 可以跳越區域最小值達到全域最大值



圖表八、最小值示意圖



圖表九、random 不平均現象

電腦系統用的 pseudo random 不很隨機，特別是在大量使用後更可能會重複出現

```
/* ########################
 *  警告:
 *  本附件程式碼僅爲參考用, 如因執行本程式造成系統毀損, 概不負責
 *  如無法編譯, 或者無法成功執行, 請自行解決
 *  聯絡作者:
 *     莊偉赳  Wei-Chiu Chuang     wcchuang@ck.tp.edu.tw
 *     徐茂芳  Mau Fang  Shiu     mfshiu@ck.tp.edu.tw
 *   Warning:
 *      The source code appended is provided only for test. We don't guarantee
   any responsibility over any system crash if you execute these programs.
   *  Facing any problem with compile or can't successfully execution? Please resolve it by yourself ,
or contact authors over mails:
   *     Wei-Chiu Chuang   wcchuang@ck.tp.edu.tw
   *    Mau Fang  Shiu              mfshiu@ck.tp.edu.tw
  ############################*/
/* GPBubble
   co-created by Wei-Chiu Chuang , Mau Fang  Shiu
   製作人: 徐茂芳、莊偉赳

*/
/////config.h
#include <pvm3.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <math.h>
#include <limits.h>

/*  user defined */
#include <struct.h>
//#include <vmem.h>


#define ARG_ERROR -1    /*  input argument error */
#define OFFSPRINGS      10
#define FUNCTIONS 5
#define MARK_COMPLETE   2<<31-1
#define GROUP           "GPBUBBLE"
#define PATHPROGRAM     "/home/cominging/SF/gpbubble"

#define ONE_SECOND       1000000
#define FIVE_SECOND      5000000
#define TEN_SECOND      10000000
#define HALF_MINUTE     30000000
#define ONE_MINUTE      60000000
```

```
#define RUNNING         1<<1
#define EXIT            1<<2
#define TERMINATED      1<<3

extern void vinit(void);                    /* 初始化虛擬記憶體 */
extern int  v_alloc(int);                   /* 分配虛擬記憶體空間 */
extern int  vread(int);                 /* 讀取虛擬記憶體位址 */
extern int  vwrite(int, int);               /* 寫入資料 */
extern int  vfree(int);

extern struct stacks (*pEssential[])(struct stacks *);
/*                      對於 do_compare 的比較   */
#define  CMP_BIGGER             0  /* a >  b */
#define      CMP_BIGGEREQUAL    1  /* a >= b */
#define  CMP_EQUAL              2  /* a == b */
#define  CMP_SMALLEREQUAL       3  /* a <= b */
#define  CMP_SMALLER            4  /* a <  b */
#define      CMP_NOTEQUAL           5  /* a != b */
#define  CMP_COMPARE            6  /* 比較的種類 */

//char *pfunction_name[] = { "do_swap", "do_for", "do_next", "do_compare",
"do_start", "do_stop" };
extern struct stacks do_swap(struct stacks *);
extern struct stacks do_for(struct stacks *);
extern struct stacks do_next(struct stacks *);
extern struct stacks do_compare(struct stacks *);
extern struct stacks do_start(struct stacks *);
extern struct stacks do_stop(struct stacks *);

///////crossover.c
#include <config.h>
/*-------------------------------------------------------------------------
 * void crossover( struct stacks *)
 * 參數說明:
 * % *p
 *   是這個程式 structure 的起始位址
 *   要決定變異的機率 ex. 99 %
 *-------------------------------------------------------------------------*/
#define CROSSOVER 0.9
void crossover(struct stacks *p){
      srandom(time(0));
      srand(time(0));
#ifdef DEBUG
      printf("Entering crossover(*)\n");
#endif //DEBUG
```

```
      struct stacks *f1,*f2,*sw,*t=p;
      int swap1,swap2;
      int npos1,npos2;
      int nPlen=0;
/*  先取得總長度  */
#ifdef DEBUG
      printf("get total length\n");
#endif //DEBUG
      while(1){
          if( t->next_depth != NULL ){
                t = t->next_depth;
                nPlen ++;
          }else if( t->next != NULL ){
                // go across <for>,<compare>
                if( t->upper_depth == t->next  )t = t->next->next;
                else t = t->next;
                nPlen ++;
          }else break;
      }
#ifdef DEBUG
      printf("do swap!\n");
#endif //DEBUG
/*  任取兩者交換
 *  抑或是找兩塊來交換 ?
 *                      */
      while(/* swap1 != swap2*/ random() < (int)RAND_MAX*CROSSOVER ){  // 到底要
跑幾次呢?
            do{
                  swap1 =(int) ( (float)nPlen*rand()/(RAND_MAX));
                  swap2 =(int) ( (float)nPlen*rand()/(RAND_MAX));
            }while( swap1 == 0 || swap2 == 0 );
#ifdef RUN//DEBUG
            printf("swap1:%d swap2:%d nPlen:%d\n",swap1,swap2,nPlen);
#endif // DEBUG
            f1=f2=p;
            npos1=npos2=0;
            /*  走訪到要交換的位址 */
            while(npos1 < swap1){
                if( f1->next_depth != NULL ){
                  f1 = f1->next_depth;
                  npos1++;
                }else if( f1->next != NULL ){
                  if( f1->upper_depth == f1->next  )f1 = f1->next->next;
                  else f1 = f1->next;
                  npos1++;
                }else break;
```

```
            }
#ifdef RUN//DEBUG
      printf("finish traveling swap1\n");
#endif  // DEBUG
            /*  走訪到要交換的位址 */
            while(npos2 < swap2){
                if( f2->next_depth != NULL ){
                  f2 = f2->next_depth;
                  npos2++;
                }else if( f2->next != NULL ){
                  if( f2->upper_depth == f2->next  )f2 = f2->next->next;
                  else f2 = f2->next;
                  npos2++;
                }else break;

            }
#ifdef RUN//DEBUG
      printf("finish traveling\n");
#endif  // DEBUG
            /*  交換  */
            struct stacks *f1_next,*f1_former;  /*   保持本身不變動  */
            sw = f1;           /*   只移動前後指標  */
            f1_next = f1->next;
            //printf("1\n");
            f1_former = f1->former;
            //printf("2\n");

            if( f1->former != NULL)
                  f1->former->next = f2;  /* 把 f1 的前一個的後面改爲 f2 */
            //printf("3\n");
            if( f1->next != NULL )
                  f1->next->former = f2;  /* 把 f1 的後面(已改爲 f2 的後面)的前面
改成 f1 以方便倒朔  */
            //printf("4\n");
            f1->next = f2->next;    /* 把 f1 的後面改爲 f2 的後面  */
            //printf("5\n");
            f1->former = f2->former;/* 把 f1 的前面改爲 f2 的前面  */
            //printf("6\n");

#ifdef RUN//DEBUG
      printf("f1 done\n");
#endif // DEBUG
            if( f2->former != NULL )
                  f2->former->next = f1;  /* 把 f2 的前一個的後面改爲 f1 */
            if( f2->next != NULL )
```

```
                f2->next->former = f2;  /* 把 f2 的後一個(原 f1 的後一個)的前面
改爲 f2 */
            f2->next = f1_next;     /* 把 f2 的後面改爲原 f1 的後面*/
            f2->former = f1_former; /* 把 f2 的前一個改爲原 f1 的前一個  */
#ifdef RUN//DEBUG
      printf("f2 done\n");
#endif // DEBUG
      }
#ifdef DEBUG
      printf("finish crossover...\n");
#endif
}

//////do_macro.c
#include <config.h>

struct stacks (*pEssential[])(struct stacks *) = {
/*    do_max,
      do_min,*/
      do_swap,
      do_for,
      do_next,
      do_compare,
      do_stop,
      do_start,

};

struct stacks do_swap(struct stacks *InData)
{
      struct stacks return_stack;
      int a = InData->element[0];   // 此指位置
      int b = InData->element[1];   // 此指位置

      if( InData->nElement != 2 )
      {
            return_stack.arg_error = InData->arg_error = 1;
      }
      else if( (a = vread(a)) == ARG_ERROR || (b = vread(b)) == ARG_ERROR )
      {
            return_stack.arg_error = InData->arg_error = 1;
      }
      else
      {
            return_stack            = *InData;
            return_stack.intdata[a] = InData->intdata[b];
```

```
            return_stack.intdata[b] = InData->intdata[a];
            *InData                 = return_stack;
      }
      return return_stack;
}

struct stacks do_for(struct stacks *InData)
{
      struct stacks return_stack;
      int start;  /* 先不在前面抓出 start 的資料,
                     在 #1 的地方會對 start 做完整的檢查. */
      int stop;
      int step = InData->element[2];

      /*--------------- #1 ---------------*
       * 若 nElement 爲 4 就表示已經 allocate 過了, 可以直接 vread 出來
       * 不然就試著去 allocate 一塊位置; 若成功, 則 nElement 則改爲 4,
       * 如果失敗, arg_error 就會設成 True (1);
       * 在這裡我先前面直接存取 InData 的內容, 因爲對 for 而言, 不需要什麼
       * 特殊的判斷.
       */

      if( InData->nElement == 4 )
      {
            start = vread(InData->element[3]);
      }
      else
      {
            InData->element[1] %= InData->nData;
            InData->element[0] %= InData->nData;
            start = InData->element[0];
            InData->nElement = 4;
            InData->element[3] = v_alloc(start);
            if( (InData->element[3]) == -1 )
            {
                  InData->arg_error = 1;
            }
      }
      stop = InData->element[1];

      /* 在這裡判斷 start 是不是有不符條件的行爲, 會依據 step 來判斷,
       * 若成功就直接 vwrite +1 的值進去, 然後將 return_stack 內容改爲
       * InData 的內容傳回. */
      if( InData->nElement == 4 )
      {
            switch(step)
```

```
		{
			case 1:
				if( start > stop )
					InData->nElement = 1;
				else
					vwrite(InData->element[3], start+step);
				break;
			case 0:
				if( start == stop )
					InData->nElement = 1;
				break;
			case -1:
				if( start < stop )
					InData->nElement = 1;
				else
					vwrite(InData->element[3], start+step);
				break;
		}
	}
	return_stack = *InData;
	return return_stack;
}

/* 廢渣 ><" 像我一樣... */
struct stacks do_next(struct stacks *InData)
{
	struct stacks return_stack = *InData;
	return return_stack;
}

struct stacks do_compare(struct stacks *InData)
{
	struct stacks return_stack;

	return_stack = *InData;

	/* 若 InData 中的 nElement 爲 3 (Default) 則代表之前沒有判斷過這個
	 * compare, 而 nElement 只可能爲 3 或 1. 若判斷式成功, 則 element[0]
	 * 的值會改成 1 否則的就爲 0 */
	if( InData->nElement == 3 )
	{
		int a = vread(InData->element[0]%InData->nData);
		int b = vread(InData->element[1]%InData->nData);
		if( a == ARG_ERROR || b == ARG_ERROR )
		{
			return_stack.arg_error = InData->arg_error = 1;
```

```
            }
            else
            {
                  return_stack.arg_error = InData->arg_error = 0;
                  InData->nElement = 1;
                  switch(InData->element[2])
                  {
                        case CMP_BIGGER:
                              if( InData->intdata[InData->element[0]] > InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                        case CMP_BIGGEREQUAL:
                              if( InData->intdata[InData->element[0]] >= InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                        case CMP_EQUAL:
                              if( InData->intdata[InData->element[0]] == InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                        case CMP_SMALLEREQUAL:
                              if( InData->intdata[InData->element[0]] <= InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                        case CMP_SMALLER:
                              if( InData->intdata[InData->element[0]] < InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                        case CMP_NOTEQUAL:
                              if( InData->intdata[InData->element[0]] != InData-
>intdata[InData->element[1]] )
                                    return_stack.element[0] = 1;
                              else return_stack.element[0] = 0;
                              break;
                  }
//                InData->element[0] = return_stack.element[0];
            }
```

```
	}
	else
	{
		/* 若判斷過, 則直接將 element[0] 改爲 0 */
		InData->element[0] = return_stack.element[0] = 0;
	}
	return return_stack;
}

struct stacks do_start(struct stacks *InData)
{
	struct stacks return_stack = *InData;
	return return_stack;
}

struct stacks do_stop(struct stacks *InData)
{
	struct stacks return_stack = *InData;
	return return_stack;
}
```

//////exec_judge.c

```
/*-------------------------------------------------------------------------
 * executor:
 * $ generator & executor 寫在同一個程式, 然後 fork() 產生其子程序之後, 由 pvm
 *   傳送產生的資料, 然後執行.
 * $ 從 pvm_recv 中取得由 generator 產生所要執行的程式.
 * $ 在結束時 pvm_send 送出最後一個 stack 中的資料
 *   % 那麼如果不是跑到最後的話, 則設一個變數 sigCaught, 若 sig_catcher 接收到
 *     生命週期函數 TimeSensor() 所送出的終結訊號, 則會使 sigCaught 之值爲真,
 *     則 execute 檢查到這個變數時, 就會自動跳出, 並且呼叫函數 coda 將資料
 *     傳給 Judge.
 * $ Judge 程式
 * $ Virtual Memory
 *   % int vread(addr)
 *   % int vwrite(addr, info)
 *   % int v_alloc(info)
 *   % int vfree(addr)
 *-------------------------------------------------------------------------*/

// Q1: 是不是所有送來的資料都要有不同的測資? 看得出優劣高下?

#include "config.h"

#ifdef DEBUG
#include <string.h>
```

```
char *mirror[] = { "do_swap", "do_for", "do_next", "do_compare", "do_stop",
"do_start" };
#endif

int sigCaught = 0;

void sig_catcher(int);              /* 當接收到 SIG_USR1 之反應 */

void catch_data(void);
void Judge(int *, int);
void send_main_data(int);
void coda(struct stacks *, struct stacks *);
void int_init(void);

int   defaultData[MAX_INT_DATA];    /* testing data array */
int   nDefaultData;                 /* length of defaultData */
int   mytid;                        /* Task ID of this process */
int   mygid;                        /* Group ID of this process */
int   seed;                   /* */
struct      stacks *start;              /* */

#ifdef DEBUG
int   steps;
void ECHO(const char *str)
{
      char buf[512], buf2[24];
      sprintf(buf2, "%d-%03d\t", mytid, steps++);
      strcpy(buf, "echo '");
      strcpy(buf+strlen(buf), buf2);
      strcpy(buf+strlen(buf), str);
      strcpy(buf+strlen(buf), "' >> ~/SF/test");
      system(buf);
}
#endif

void executor()
{
      struct stacks *session, temp;
      int x, Halt = 0;

      signal(SIGUSR1, sig_catcher);
      mytid = pvm_mytid();
      mygid = pvm_joingroup(GROUP);
      char buf[400];
      sprintf(buf, "mytid%d mygid%d", mytid, mygid);
      ECHO(buf);
```

```
	// Catching data from parent.
	catch_data();

	// Initialize ``srand'' function's seed
	srand(seed);

	// Generate the ints what we want the program to sort.
	int_init();

	// Initialize the agency ``session''
	session = start;

	// Initialize the virtual memory
	vinit();

	temp.nData = nDefaultData;
	for(x=0; x<nDefaultData; x++)
	{
		temp.intdata[x] = defaultData[x];
	}

	while( !sigCaught && !Halt )
	{
		// Copy the last testing data to the next.
		for(session->nData=temp.nData, x=0; x<temp.nData; x++)
		{
			session->intdata[x] = temp.intdata[x];
		}

#ifdef DEBUG
/*		char buf[400];
		for(x=0; x<6; x++)
			if( pEssential[x] == session->pfunc )
				break;
		sprintf(buf, "pfunc(%-11s); %d %d", mirror[x], session->nData,
session->nElement);
		ECHO(buf);*/
#endif
		temp = (*session->pfunc)(session);
		// If argument is wrong, then halt this one.
		if( temp.arg_error)
		{
			Halt = 1;
		}
		else if( (session->next_depth) != NULL )
```

```
            {
                  if( (session->pfunc) == do_compare )
                  {
                        // element[0] controlled if this node is ran.
                        if( temp.element[0] )
                              session = session->next_depth;
                        // If element[0] is not true, then skipped it.
                        else if( (session->next) != NULL )
                              session = session->next;
                        // Else.. Halt.
                        else Halt = 1;
                  }
                  else if( session->pfunc == do_for )
                  {
                        // It means this cycle have done.
                        if( temp.nElement == 1 )
                        {
                              vfree(temp.element[3]);
                              if( session->next != NULL )
                                    session = session->next;
                              else Halt = 1;
                        }
                        else session = session->next_depth;
                  }
                  else printf("(executor) Unexpection Error: Unknown for
next_depth.\n");
            }
            else if( session->next != NULL )
                  session = session->next;
            else Halt = 1;
      }
      coda(start, session);
}

/*-------------------------------------------------------------------------
 * void catch_data()
 * $ 從 parent 抓 data.
 *-------------------------------------------------------------------------*/
void catch_data(void)
{
      int nElement, element[MAX_ELEMENT], pfunc, x;
      struct stacks *now, *last;
#ifdef DEBUG
      int count = 0;
      char buf[256], buf2[512];
#endif
```

```
	now = NULL;
	last = NULL;

	pvm_recv(-1, 1);
	pvm_upkint(&nDefaultData, 1, 1);
	pvm_upkint(defaultData, nDefaultData, 1);
	ECHO("Caught first Integer datum.");
	while(1)
	{
		pvm_recv(-1, 1);
		pvm_upkint(&nElement, 1, 1);
		pvm_upkint(element, nElement, 1);
		pvm_upkint(&pfunc, 1, 1);
		ECHO("Caught 1");
#ifdef DEBUG
/*		sprintf(buf2, "(%03d) pfunc=%11s;%2d, %1d; seed: %10d", count++,
mirror[pfunc], pfunc, nElement, seed);
		for(x=0; x<nElement; x++)
		{
			sprintf(buf, "\t%12d", element[x]);
			strcpy(buf2+strlen(buf2), buf);
		}
		ECHO(buf2);*/
#endif
		// do_start
		if( pfunc == 5 )
		{
			ECHO("yeah! do start");
			start = (struct stacks *)malloc(sizeof(struct stacks));
			now = start;
			now->arg_error     = 0;
			now->former  = NULL;
			now->nElement      = 0;
			now->next_depth  = NULL;
			now->pfunc   = do_start;
			now->upper_depth = NULL;
			now->next    = (struct stacks *)malloc(sizeof(struct stacks));
			(now->next)->former      = now;
			(now->next)->upper_depth = NULL;
			now          = now->next;
		}
		// do_stop
		else if( pfunc == 4 )
		{
			ECHO("haha... Do Stop.");
			now->arg_error    = 0;
```

```
                now->nElement     = 0;
                now->next   = NULL;
                now->next_depth   = NULL;
                now->pfunc  = do_stop;
                break;
          }
          else
          {
                now->arg_error    = 0;
                now->pfunc  = pEssential[pfunc];
                now->nElement     = nElement;
                for(x=0; x<nElement; x++)
                      now->element[x] = element[x];
                if( now->pfunc == do_for || now->pfunc == do_compare )
                {
                      now->next   = NULL;
                      now->upper_depth= (now->former)->upper_depth;
                      now->next_depth   = (struct stacks
*)malloc(sizeof(struct stacks));
                      (now->next_depth)->former     = now;
                      (now->next_depth)->upper_depth      = now;
                      now = now->next_depth;
                }
                else if( now->pfunc == do_next )
                {
                      now->next_depth = NULL;
                      if( (now->former)->pfunc == do_for || (now->former)-
>pfunc == do_compare )
                      {
                            now->next = now->former;
                            now->upper_depth = now->former;
                            now = now->former;
                      }
                      else if( (now->former)->upper_depth != NULL )
                      {
                            now->next = (now->former)->upper_depth;
                            now->upper_depth = (now->former)->upper_depth;
                            now = (now->former)->upper_depth;
                      }
                      now->next = (struct stacks *)malloc(sizeof(struct
stacks));
                      (now->next)->former = now;
                      (now->next)->upper_depth = now->upper_depth;
                      now = now->next;
                }
                else
```

```
                {
                        now->next_depth = NULL;
                        now->next = (struct stacks *)malloc(sizeof(struct
stacks));
                        (now->next)->upper_depth = now->upper_depth;
                        (now->next)->former = now;
                        now = now->next;
                }
            }
      }
#ifdef      DEBUG
      ECHO("DATA catching is finished.");
#endif
}

/*-------------------------------------------------------------------------
 * void sig_catcher(int signal)
 * $ It would be called by the ``TimeSensor()''
 * $ Set the variable ``sigCaught'' true.
 *-------------------------------------------------------------------------*/
void sig_catcher(int handle)
{
      signal(SIGUSR1, sig_catcher);
      sigCaught = handle;
      ECHO("\033Signal caught...");
}

/*-------------------------------------------------------------------------
 * void coda()
 * $ 釋放所佔據的記憶體空間.
 * $ 將最後取得的 intdata 送至 Judge()
 *-------------------------------------------------------------------------*/
void coda(struct stacks *start, struct stacks *node)
{
      int intdata[MAX_INT_DATA], x, nData;
      struct stacks *session, *back;
      extern void release(struct stacks *p);
//    ECHO("\tCoda executed.");
      nData = node->nData;
      for(x=0; x<MAX_INT_DATA; x++)
            intdata[x] = node->intdata[x];
//    ECHO("\t\t\tKilling tree.");
      release(start);
#ifdef DEBUG
/*    char buf[1400];
      buf[0] = '\0';
```

```
	for(x=0; x<nData; x++)
	{
		sprintf(buf+strlen(buf), "%d ", intdata[x]);
	}
	ECHO(buf);*/
#endif
//	ECHO("Judging..");
	Judge(intdata, nData);
}

void Judge(int *intdata, int nData)
{
	  int x, y, score;
//	ECHO("Entered Judging.");
	for(score=0, x=0; x<nData; x++)
	{
		for(y=x+1; y<nData; y++)
		{
			if( intdata[x] <= intdata[y] )
				score++;
			else
				score--;
		}
	}
#ifdef DEBUG
//	char buf[400];
//	sprintf(buf, "\tScore: %4d %4d", score, nData);
//	ECHO(buf);
#endif
	  if( !sigCaught )
		send_main_data(score);
	else
		return;
}

void send_main_data(int score)
{
	int tid, tt = score;
	tid = pvm_parent();
	pvm_initsend(PvmDataDefault);
	pvm_pkint(&tt, 1, 1);
	pvm_send(tid, 1);
	ECHO("\t\tScore Sent..");
}
```

```
//////generator.c
#include <config.h>
extern void assign_data(struct stacks *);
extern void crossover(struct stacks *);
extern void mutation(struct stacks *);

extern struct stacks do_swap(struct stacks *);
extern struct stacks do_for(struct stacks *);
extern struct stacks do_next(struct stacks *);
extern struct stacks do_compare(struct stacks *);
extern struct stacks do_start(struct stacks *);
extern struct stacks do_stop(struct stacks *);

#define DEFAULTLENGTH          2
#define     DO_START           5
#define DO_STOP                4
#define MAX_MEMORY_UNIT          100
/*到底要幾個選項?
 * #define FUNCTIONS   5
 *
 *************************/

/*
 *重大改變!
 *   使用自訂的虛擬記憶體
 *   Valloc()
 *   Vwrite()
 *   Vinit()
 *   Vread()
 * */


/*-------------------------------------------------------------------------
 * Comment by JoJo 02/11/23
 * 這個部份是產生GP程式的地方,
 * 想要採用現成的GA Library: GAlib made by MIT,
 * some examples also employ PVM^^
 *
 * 嗯, 隨機產生不定長度, 不定內容的東西
 * 假如 status == INIT:
 * 一次產生數個程式offsprings
 * 第一步:  產生DEFAULT_LENGTH長度之程式
 * 進入迴圈->跑到 random==FINISH
 *    執行不定次crossover
 *    執行不定次mutation
 * 結束迴圈
```

```
 * 回到 main()--> 執行 executor(); --> judge()--> 送回 generator()重新
 *
 * 假如 status == REPEAT:
 * 第一步: 附加 append 不定長度 LENGTH 程式
 * 進入迴圈->跑到 random==FINISH
 *    執行不定次 crossover
 *    執行不定次 mutation
 * 結束迴圈
 * 回到 main()--> 執行 executor(); --> judge()--> 送回 generator()重新
 *
 *-------------------------------------------------------------------------*/
#define INIT            0
char *pfunction_name[] = { "do_swap", "do_for", "do_next", "do_compare",
"do_stop","do_start"};
//extern char *pfunction_name[];
void genNew(struct stacks *);
void genAppend(struct stacks *);

void generator(int status, struct stacks **prog, int nInstances)
{
      srand(time(0));
      srandom(time(0));
      int i,j,tmp;
      struct stacks *p;
      printf("(generator) enter me!\n");
      if( status == INIT )
      {
            printf("(generator) status == INIT\n");
            for(i=0; i<nInstances; i++)
            {
                  *(prog+i) = (struct stacks *)malloc(sizeof(struct stacks));
                  genNew(*(prog+i));
            }
      }
      else
      {
            printf("(generator) status != INIT\n");
            for(i=0; i<nInstances; i++)
            {
                  if( *(prog+i) == NULL )
                  {
                        *(prog+i) = (struct stacks *)malloc(sizeof(struct
stacks));
                        printf("(generator) node@%d is NULL (%d)\n", i,
*(prog+i));
                        genNew(*(prog+i));
```

```
                }
                else
                {
                        printf("(generator) appending to %d\n", i);
                        genAppend(*(prog+i));
                }
        }
    }
}

void genNew(struct stacks *prog)
{
    int plength=0, j, tmp;
    struct stacks *p;
    printf("(generator::genNew()) caught %d\n", prog);
    p=prog;
    p->former   = NULL;
    p->upper_depth    = NULL;
    p->next_depth     = NULL;
    p->pfunc    = pEssential[DO_START];
    p->nElement = 1;
    p->element[0]     = 0;
    assign_data(p);
    p->next = (struct stacks *)malloc( sizeof(struct stacks) );
    (p->next)->former = p;
    (p->next)->upper_depth  = NULL;
    p = p->next; //  向下一個移動

    // remember clearing data
    for(j=0; j<DEFAULTLENGTH; j++)
    {
        do {
            tmp = (int) ((float)FUNCTIONS*rand()/(RAND_MAX));
            if( pEssential[tmp] != do_start && pEssential[tmp] != do_stop)
                break;
        } while( 1 );
        p->pfunc = pEssential[tmp];
        if( p->pfunc == do_for )
        {   // 這可就麻煩了
            p->next           = NULL;
            // 目前還不知道
            // 假如這是第二層的 for 就重要了
            p->upper_depth    = p->former->upper_depth;
            assign_data(p);
            p->next_depth = (struct stacks *)malloc( sizeof(struct
stacks));
```

```
            (p->next_depth)->former = p;
            (p->next_depth)->upper_depth = p;
            p = p->next_depth; // 進入下一層
        }
        else if( p->pfunc == do_next )
        {     //結束這個 for
        /*
         * 不用向下建了
         * 找出<for>的位址
         * 幫 for 建下一個
         ************************/
            struct stacks *f;
            assign_data(p);
            if( (p->former)->pfunc == do_for
                || (p->former)->pfunc == do_compare )     /* added by
Cominging */
            {
                f = p->former;
                p->next     = f;        /* 指向 for */
                p->upper_depth    = f;
                p->next_depth     = NULL;
                p = f;                  /* 爲<for, compare>建造下一個資
料 */
            }
            else if( (p->former)->upper_depth != NULL )
            {
                f = (p->former)->upper_depth; /* 有問題, 萬一上一個就是
for?  */
                p->next     = f;        /* 指向 for */
                p->upper_depth    = f;
                p->next_depth     = NULL;
                p = f;                  /* 爲<for>建造下一個資料 */
            }
            else
            {
                p->upper_depth    = NULL;
                p->next_depth     = NULL;
                printf("\033[1;30m(ERROR!!)\033[m");
            }
            p->next = (struct stacks *)malloc( sizeof(struct stacks) );
            (p->next)->former = p;
            (p->next)->upper_depth = p->upper_depth;
            p = p->next;
        }
        /*********************************
         * do_compare 與 do_for 共用同一種
```

```
             * 結尾 do_next
             *********************************/
            else if( p->pfunc == do_compare )
            {
                  p->next           = NULL; // 目前還不知道
                  p->upper_depth    = p->former->upper_depth; // 假如這是第二層
的 compare 就重要了
                  assign_data(p);
                  p->next_depth = (struct stacks *)malloc( sizeof(struct
stacks));
                  (p->next_depth)->former = p;
                  (p->next_depth)->upper_depth = p;
                  p = p->next_depth; // 進入下一層
            }
            else
            {
                  p->next_depth     = NULL;
                  // 清除資料
                  // 指定資料
                  assign_data(p);
                  p->next = (struct stacks *)malloc( sizeof(struct stacks) );
                  (p->next)->upper_depth = p->upper_depth;
                  (p->next)->former = p;
                  p = p->next; //  向下一個移動
            }
      }// end of for(j)
      // ad tail do_stop
      p->pfunc = do_stop;
      p->next = p->next_depth = NULL;
      assign_data(p);
      while(p->pfunc != do_start)
      {
            p = p->former;  // 走訪到最原始位址
      }
#ifdef RUN
      crossover(p);
      mutation(p);
#endif
}

void genAppend(struct stacks *prog)
{
      int tmp;
      struct stacks *d, *where_stop;

      printf("(generator::genAppend()) caught %d\n", prog);
```

```
/*************************************************************
 *
 * Comment by JoJo
 *   After our first generation,
 *   the second generations is to append variable length
 *   to the program
 * Warning:
 *   some of the offsprings have already killed ....
 *   If find some killed "body", then copy existed one to it.
 *
 *************************************************************/

      d = prog;
      if( d == NULL )
      {
            printf("dEAD..\n");
            return;
      }
      /*
      while( d->next->pfunc != do_stop )
      {
            d = d->next;
            if( d == NULL ) printf("ERROR!!!\n");
      }
      */
      while(d->next != NULL || d->next_depth != NULL )
      {
            if( d->next != NULL) d=d->next;
            else if( d->next_depth != NULL) d=d->next_depth;
      }
      where_stop = d;
      if( where_stop->pfunc != do_stop ) printf("\n\t\033[0;32mBIGERROR lost
do_stop\033[0m\n");
      d = d->former;
#ifdef DEBUG
//    printf("do_stop found ..\n");
#endif
      tmp=(int)((float)FUNCTIONS*rand()/(RAND_MAX));
      if( pEssential[tmp] == do_stop ) { //啥事也別做了
            printf("1st use do_stop? do nothing..exit \n");
      }
      else
      {
            d->next = (struct stacks *)malloc( sizeof ( struct stacks ) );
            d->next->former = d;
            d->next->upper_depth = d->upper_depth;
```

```
            d->next->pfunc = pEssential[tmp];
            //assign_data( d->next);
            d = d->next;

            while(1)
            {     //執行到他成爲 do_stop 的上面
                  //while( d->next->pfunc != do_stop )d = d->next;

                  tmp=(int) ( (float)FUNCTIONS*rand()/(RAND_MAX));
                  if( pEssential[tmp] == do_stop )
                  {
#ifdef DEBUG
//                      printf(">>do_stop...finish");
#endif
                      /* 找到 where_stop, 把 d->former 的 next 設成 where_stop ,
free (d) */
                      //assign_data(where_stop);
                      d->former->next = where_stop;
                      where_stop->former = d->former;
                      free( d);
                      where_stop->next=NULL;
                      where_stop->next_depth=NULL;
                      break;
                  }// 如果等於 STOP 那就別做了
                  else {
                      where_stop->former = d;
#ifdef DEBUG
//                      printf("set where_stop->former = d= %d\n",(int)d);
#endif
                  }

                  d->pfunc = pEssential[tmp];
                  if( d->pfunc == do_for )
                  {     // 這可就麻煩了
#ifdef DEBUG
//                      printf("do->pfunc=do_for\n");
#endif
//                      printf("former= %d\n",d->former);
                      d->next           = NULL;           // 目前還不知道
                      d->upper_depth    = d->former->upper_depth;// 假如這是第
二層的 for 就重要了
                      assign_data(d);
                      d->next_depth = (struct stacks *)malloc( sizeof(struct
stacks));
                      d->next_depth->former = d;
                      d->next_depth->upper_depth = d;
```

```
                        d = d->next_depth;              // 進入下一層
#ifdef DEBUG
//                      printf("finish construct do_for\n");
#endif
                }
                else if( d->pfunc == do_next )
                {       //結束這個 for
                /**********************
                 * 不用向下建了
                 * 找出<for>的位址
                 * 幫 for 建下一個
                 **********************/
#ifdef DEBUG
//                      printf("do->pfunc=do_next\n");
#endif
                        struct stacks *f;
                        if( d->former->pfunc == do_for )
                              f = d->former;
                        else
                              f = d->former->upper_depth;   /*  有問題  萬一上一
個就是 for?  */
                        d->next     = f;                    /** 指向 for **/
                        d->upper_depth    = f;
                        d->next_depth     = NULL;
                        assign_data(d);
                        if( f != NULL)
                              d = f;                        /*  成爲<for>建造下
一個資料 */
                        d->next = (struct stacks *)malloc( sizeof(struct
stacks) );
                        d->next->former = d;
                        d->next->upper_depth = d->upper_depth;
                        d = d->next;
#ifdef DEBUG
//                      printf("finish construct do_next\n");
#endif
                }
                /**********************************
                 * do_compare 與 do_for 共用同一種
                 * 結尾 do_next
                 **********************************/
                else if( d->pfunc == do_compare )
                {
                        d->next           = NULL; // 目前還不知道
                        d->upper_depth    = d->former->upper_depth; // 假如這是
第二層的 compare 就重要了
```

```
                        assign_data(d);
                        d->next_depth = (struct stacks *)malloc( sizeof(struct
stacks));
                        d->next_depth->former = d;
                        d->next_depth->upper_depth = d;
                        d = d->next_depth; // 進入下一層
                  }
                  else
                  {
                        d->next_depth     = NULL;
                        // 清除資料
                        // 指定資料
                        d->next = (struct stacks *)malloc( sizeof(struct
stacks) );
                        assign_data(d);
                        d->next->upper_depth = d->upper_depth;
                        d->next->former=d;
                        d = d->next; //  向下一個移動
                  }
            }// while(1)
            /*  already done before */
            while(d->former != NULL)
                  d = d->former;  // 走訪到最原始位址
#ifdef RUN
            // 交換
            crossover(d);
            // 變異
            mutation(d);
#endif
      }// end of if(do_stop)
}

/*
 * MAX_MEMORY_UNIT      100
 *
 * **********/
/*-------------------------------------------------------------------------
 * void assign_data(struct stacks *)
 *
 *-------------------------------------------------------------------------*/
/*                      對於 do_compare 的比較  */
#define  CMP_BIGGER           0  /* a >  b */
#define      CMP_BIGGEREQUAL  1  /* a >= b */
#define  CMP_EQUAL            2  /* a == b */
#define  CMP_SMALLEREQUAL     3  /* a <= b */
```

```
#define  CMP_SMALLER          4  /* a <  b */
#define      CMP_NOTEQUAL           5  /* a != b */
#define  CMP_COMPARE          6  /* 比較的種類 */

void assign_data(struct stacks *p)
{
#ifdef DEBUG
//    printf("assign_data..\n");
#endif
      p->nElement = 1;
      int addr1, addr2;
      if( p->pfunc == do_swap )
      {
            do {
                  addr1 = (int) ((float)MAX_MEMORY_UNIT*rand()/RAND_MAX);
                  addr2 = (int) ((float)MAX_MEMORY_UNIT*rand()/RAND_MAX);
            } while( addr2 == addr1 );
            p->nElement = 2;
            p->element[0] = addr1;
            p->element[1] = addr2;
#ifdef DEBUG
            printf("do_swap->");
#endif
      }
      else if( p->pfunc == do_start )
      {
            p->nElement = 0;
#ifdef DEBUG
            printf("(( do_start->");
#endif
      }
      else if( p->pfunc == do_compare )
      {
            do
            {     // addr2 不能與 addr1 一樣 不然等於浪費時間 by jojo
                  addr1 = (int) ((float)MAX_MEMORY_UNIT*rand()/RAND_MAX);
                  addr2 = (int) ((float)MAX_MEMORY_UNIT*rand()/RAND_MAX);
            }
            while( addr2 == addr1 );
            p->nElement = 3;
            p->element[0] = addr1;
            p->element[1] = addr2;
            p->element[2] = (int)((float)CMP_COMPARE*rand()/RAND_MAX);
#ifdef DEBUG
            printf("do_compare->");
#endif
```

```
	}
	else if( p->pfunc == do_stop )
	{
		p->nElement = 0;
#ifdef DEBUG
		printf("\033[0;33mdo_stop\033[0m ))\n");
#endif
	}
	else if( p->pfunc == do_for )
	{
		//  define starter, goal, steper
		//  FOR i=starter->goal step stper;
		/*
		 * FOR_ARG_START = specific[0]
		 * FOR_ARG_GOAL  = specific[1]
		 * FOR_ARG_STEP  = specific[2]
		 * FOR_ARG_COUNTER   = specific[3];
		 * ************************************/
		  p->element[0] = random();
		  p->element[1] = random();
		  // Cominging changed.
		  p->element[2] = ((int)((float)rand()/RAND_MAX)*INT_MAX)%3-1;
		  p->nElement = 3;
#ifdef DEBUG
		printf("do_for->");
#endif
	}
	else if( p->pfunc == do_next )
	{
		// Cominging changed.
		p->nElement = 0;
#ifdef DEBUG
		printf("do_next->");
#endif
	}
	else
	{
#ifdef DEBUG
		printf("\033[1;33m(assign_data) ERROR!!\033[m can't assign what p-
>pfunc is...\n");
#endif
	}
}

//////main.c
#include <stdio.h>
```

```
#include "config.h"
#include <stdlib.h>
#include <time.h>

/*  implemented by JoJo  */
extern void mututation();
extern void crossover();
extern void generator(int, struct stacks **, int);
extern void send_data(struct stacks **, int *, int );
extern void assign_data(struct stacks *);
extern void selectbetter(struct stacks **, int*, int, int*);
/*  implemented by Koala */
extern void executor();
extern void Judge(int *, int);
extern void sig_catcher();
/*  ****************    */

int determineToDie(float);
int complete();
void nano(int *, int);

int judge_data[OFFSPRINGS];
#define DIFFTIME( x, y )       (y.tv_sec - x.tv_sec)*1000000 + y.tv_usec-
x.tv_usec

/*-------------------------------------------------------------------------
 * int determineToDie(float)
 *    依照 rate 產生一個亂數以決定是否判爲死亡
 *-------------------------------------------------------------------------*/
int determineToDie(float rate){
      if( random() < (int)(RAND_MAX * rate) )
            return 1; /* 哈哈 去死吧 */
      else return 0;                          /* 喔 罪不致死 */
}
/*-------------------------------------------------------------------------
 * int complete()
 *   判斷這些程式中是否已經有完全成功者？
 *-------------------------------------------------------------------------*/
int complete()
{
      int i;
      for(i=0;i<OFFSPRINGS;i++)
            if( judge_data[i]  == MARK_COMPLETE )return 1;
      return 0;
}
```

```
int main(void)
{
      /*  randomly run n instruments */
      printf("Start Genetic Programming.....\n");
       /* ********************************************************
        * 步驟:
        *   先跑一次 generator()->產生 DEFAULTLENGTH 長度程式
        *   送到 executor()執行  送到 judge 得到評估分數
        *   從中選出較好的 送回 generator()-->
        *   直到 judge()分數 reach MARK_COMPLETE  // 完全排序成功
        *
        * ********************************************************/
      srand(time(NULL));
      if( pvm_parent() != PvmNoParent )
      {
            /* Execute the queue */
            executor();
      }
      else
      {     /*   I'm parent  */
            struct timeval tv_before,tv_after;
            struct timezone tz;
            int timediff;
            /* record programs status */
            int status[OFFSPRINGS];
            int do_penalty[] = {0,0,0,0,0,0};
            /* PVM control */
            int bufid, bytes, msgtag, tid, where, gnum, score;
            int tids[OFFSPRINGS];
            int i, count=0;
            int dead;
            int is_start=0;
            struct stacks *prog[OFFSPRINGS];
            float rate;

            /*  do initialization */
#ifdef  DEBUG
            printf("(main) joingroup %s\n", GROUP);
#endif
            if( pvm_joingroup(GROUP) < 0 )
                  printf("\033[1;31mJoin Group Error!!!\033[m\n");
            while( !complete() )
            {
                  printf("not complete ..\n");
                  /*  do initialization */
                  count = 0;
```

```
                for(i=0; i<OFFSPRINGS; i++)
                {
                        status[i] = RUNNING;
                        judge_data[i] = 0;
                }
                tz.tz_dsttime = 0;      tz.tz_minuteswest = 0;
                // 取得執行前時間
                gettimeofday(&tv_before, &tz);
                /* ********************** *
                 * 每產生一代子代(用 generator() )
                 * 就呼叫一次 pvm_spawn()
                 * 用以執行這些產生出來的程式
                 * ********************** */
                generator(is_start++, prog, OFFSPRINGS);
#ifdef DEBUG
                printf("start pvm_spawn...\n");
#endif
                for(dead=0, i=0; i<OFFSPRINGS; i++)
                {
                        printf("%d\n", prog[i]);
                        if( prog[i] == NULL )
                        {
                                status[i] = TERMINATED;
                                dead++;
                        }
                        else if( prog[i]->pfunc == pEssential[4] )
                        {
                                status[i] = TERMINATED;
                                dead++;
                        }
                }
                gnum = pvm_spawn(PATHPROGRAM, (char **)0, PvmTaskDefault,
(char *)0, OFFSPRINGS-dead, tids);
                send_data(prog, tids, gnum);
#ifdef DEBUG
                printf("TIDS:");
                for(i=0; i<gnum; i++)
                        printf("[%d]%d  ", i, tids[i]);
                printf("\n");
                printf("wait until %d child(ren) leaving or terminated..\n",
gnum);
#endif
                while( count < gnum )
                {       // wait until all child leaving or terminated
                        while( (bufid = pvm_nrecv(-1, -1)) != 0 )
                        {       // Receive data
```

```
                        pvm_bufinfo( bufid , &bytes, &msgtag, &tid );
                    // identify where the message sent
                    // packet format:
                    //   1. serial number of the process
                    //   2. scores
                        pvm_upkint(&score, 1, 1);

                        for(i=-1, where=0; where<gnum; where++)
                            if( tids[where] == tid )
                            {
                                int j, t;
                                for(j=0, t=0; j<OFFSPRINGS && where >
t; j++)
                                {
                                    if( prog[j] != NULL )
                                    {
                                        if( prog[j]->pfunc !=
do_stop )
                                        {
                                            t++;
                                        }
                                    }
                                }
                                if( j >= OFFSPRINGS )
                                    printf("\033[1;31mWARNING: Can't
find tid%d's position\033[m", tid);
                                status[j] = EXIT;
                                judge_data[j] = score;
                                printf("{recv: @%2d %3d (%6d)}\n", j,
score, tid);
                                i = 1;
                                count++;
                                /*if( count++ %5 == 4 )
                                    printf("\n");*/
                                break;
                            }
                        if( i == -1 )
                        {
                            printf("\033[1;31mCan't find correct
tid..\033[m (tid:%d score:%d)", tid, score);
                        }
                    } // end of while( bufid )
//                  printf("Outta (%d)\n", count);
                    gettimeofday( &tv_after, &tz);
                    timediff = DIFFTIME( tv_before,tv_after);
                    if(     timediff > ONE_MINUTE  && !do_penalty[0] )
```

```
                        { rate = 0.9;   do_penalty[0] = 1;}
                    else if( timediff > HALF_MINUTE  && !do_penalty[1])
                        { rate = 0.5;   do_penalty[1] = 1;}
                    else if( timediff > TEN_SECOND   && !do_penalty[2])
                        { rate = 0.1;   do_penalty[2] = 1;}
                    else if( timediff > FIVE_SECOND  && !do_penalty[3])
                        { rate = 0.01;  do_penalty[3] = 1;}
                    else if( timediff > ONE_SECOND   && !do_penalty[4])
                        { rate = 0.001; do_penalty[4] = 1;}
                    else if( !do_penalty[5] )
                        { rate = 0.0005;do_penalty[5] = 1;}

                    for(i=0; i<OFFSPRINGS; i++)
                    {
                        if( status[i] != TERMINATED && status[i] != EXIT )
                        {
                            if( determineToDie(rate) )
                            {
                                /*  用 pvm_sensig(TID, SIGNUM); */
                                int x, j;
                                for(x=0, j=0; x<i; x++)
                                {
                                    if( prog[x] != NULL && prog[x]-
>pfunc != do_stop )
                                    {
                                        j++;
                                    }
                                }
                                printf(":: It is a good day to die,
%d(%d,%d)\n", tids[j], j, i);
                                pvm_sendsig(tids[j], SIGUSR1);
                                status[i] = TERMINATED;
                                judge_data[i] = -1;
                                count++;
                            }// end of if(determineToDie())
                        }
                    }// end of for(i)
                    usleep(1000);
                }// end of while(count)
                nano(judge_data, gnum);
                selectbetter(prog, judge_data, OFFSPRINGS, status);
//                getchar();
                printf("aaaa\n");
            }// end of while (!complete())
        }//end of if(fork())
}
```

```
void nano(int *judge_data, int nInstance)
{
      int i, ave = 0;
      for(i=0; i<nInstance; i++)
            ave += judge_data[i];
      ave /= nInstance;
      printf("==========> 平均:: %d (不準)\n", ave);
}

/////mutation.c
#include <config.h>
extern void assign_data(struct stacks *);
/*-------------------------------------------------------------------------------
 * void mutation( struct stacks *)
 * 參數說明:
 * % *p
 *   是這個程式 structure 的起始位址
 *   執行變異動作
 *   要決定變異的機率 ex. 1 %
 *------------------------------------------------------------------------------*/
#define  MUTATION 0.10
void mutation(struct stacks *p){
      srandom(time(0));
      srand(time(0));
#ifdef DEBUG
      printf("Entering mutation(*)\n");
#endif //DEBUG
      struct stacks *t = p;
      int mtt,tmp,npos=0,nPlen=0;

#ifdef DEBUG
      printf("get total length\n");
#endif //DEBUG
      // 取得總長度
      while(1){
          if( t!= NULL && t->next_depth != NULL ){
                printf("going to next depth\n");
                t = t->next_depth;
                nPlen ++;
          }else if(t != NULL & t->next != NULL ){
                printf("going to next\n");
                // go across <for>,<compare>
                if( t->upper_depth == t->next  )t = t->next->next;
                else t = t->next;
                nPlen ++;
```

```
          }else break;
      }
      printf("aaaa\n");
      while( random() < (int)RAND_MAX*MUTATION /*mtt != 0 || mtt !=
nPlen*/ ){  // 到底要跑幾次呢?
#ifdef DEBUG
      printf("do mutation...\n");
#endif //DEBUG
            do{
                  mtt =(int) ( (float)nPlen*rand()/(RAND_MAX));
            }while( mtt == 0);
#ifdef DEBUG//RUN//DEBUG
            printf("mtt:%d nPlen:%d\n",mtt,nPlen);
#endif // DEBUG
                while(npos < mtt){
                  if( t->next_depth != NULL ){
                      t = t->next_depth;
                      npos++;
                  }else if( t->next != NULL ){
                        if( t->upper_depth == t->next  )t = t->next->next;
                        else t = t->next;
                      npos++;
                  }else break;
            }
            tmp = (int)((float)FUNCTIONS*rand()/(RAND_MAX));
            t->pfunc = pEssential[tmp];
            assign_data(t);
      }
#ifdef DEBUG
      printf("finish mutation...\n");
#endif
}

//////select.c
#include <config.h>

#define BETTER          0.40
/*struct _SORT{
      int judge;
      int order;
}_sorted;
*/
void release(struct stacks *);

void selectbetter(struct stacks **p, int *judge_data, int nInstances, int
*status)
```

```
{
	/*  from nInstances select better k % */
	int i, j, tmp=0, count, max=0;
	int sorted[nInstances];
	int betterInstances;
#ifdef DEBUG
	printf("Entering selectbeter()...\n");
#endif
	// sort rank from higher to lowest
	// judge_data then won't be used, so we can change its content.
	for(count=0, i=0; i<nInstances; i++)
	{
		if( *(status+i) == EXIT )
			count++;
	}

	for(i=0; i<count; i++)
	{
		for(max=-INT_MAX, j=0; j<nInstances; j++)
		{
			if( *(judge_data+j) > max && *(status+j) == EXIT )
			{
				max = *(judge_data+j);
				tmp = j;
			}
		}
		*(judge_data+tmp) = -INT_MAX;
		sorted[i] = tmp;
		printf("sorted[%d]: %d (%d)\n", i, tmp, *(judge_data+tmp));
	}

	for(max=0; max<nInstances; max++)
		if( *(status+max) != EXIT )
		{
			sorted[i++] = max;
			printf("o-sorted[%d]: %d\n", i-1, max);
		}

	betterInstances = (int)(nInstances*BETTER);
	printf("betterInstance..%d\n", betterInstances);
	for(i=betterInstances; i<nInstances; i++)
	{
		/*  delete what we don't wnat */
#ifdef DEBUG
		printf("(select) Cleaning %d(%d)\n", i, *(p+sorted[i]));
#endif
```

```
            release(*(p+sorted[i]));
            *(p+sorted[i]) = NULL;
#ifdef DEBUG
            printf("(select) Cleaned.\n");
#endif
      }
}

void release(struct stacks *p)
{
/*  free memory */
/*  recursively */
#ifdef DEBUG
      printf("(select) Release...\n");
#endif
      struct stacks *t= p;
      struct stacks *nexts;
      while(t != NULL ){
            nexts = t->next;
            if( t->pfunc == do_for || t->pfunc == do_compare ){
#ifdef DEBUG
            printf("release do_for or do_compare\n");
#endif
                  release( t->next_depth );
            }
            if( ( t->pfunc == do_next && t->upper_depth != NULL ) || t->pfunc ==
do_stop ) {
                  free(t);
#ifdef DEBUG
                  printf("bumping do_next or do_stop! return\n");
#endif
                  return ;
            }
            else {
                  free(t);
                  t = nexts;
            }
      }
}

//// #include "config.h"

int nData;
int dData[MAX_INT_DATA];

extern char *pfunction_name[];
```

```
void int_init()
{
	int x;
	nData = ((int) ((float)MAX_INT_DATA*rand()/RAND_MAX))%MAX_INT_DATA+1;
	for(x=0; x<nData; x++)
		dData[x] = ((int)((float)INT_MAX*rand()/RAND_MAX));
}

void send_data(struct stacks **prog, int *tids, int nInstances)
{
	int i, j, *t;
	struct stacks *p;
	int a, tmp=0;
	int seed;

	srand(time(NULL));
	int_init();

	printf("start send_data...\n");
	for(i=0, j=0; i<OFFSPRINGS; i++)
	{
		if( *(prog+i) == NULL || prog[i]->pfunc == pEssential[4] )
			continue;
#ifdef DEBUG
/*		if( *(prog+i) != NULL )
		    printf("@%d instance..(%x) seed: %d\n", i, *(prog+i), seed);
		else
		    printf("instance(%d) is dead\n", i);
*/
#endif
		p = *(prog + i);
		printf("[%d %d|%d] ", *(tids+j), j, *(prog+i));

		pvm_initsend(PvmDataDefault);
		pvm_pkint(&nData, 1, 1);
		pvm_pkint((int *)dData, nData, 1);
		pvm_send(*(tids+j), 1);

		// 從頭到 do_stop 傳一遍
		while(p != NULL)
		{
			pvm_initsend(PvmDataDefault);
//			printf(" %d&%d",p->nElement, tmp);
#ifdef DEBUG
			if( p->pfunc == do_start )
				printf("(STT)");
```

```
                else if( p->pfunc == do_stop )
                      printf("(STP)");
                else if( p->pfunc == do_swap )
                      printf("(SWP)");
                else if( p->pfunc == do_for )
                      printf("(FOR)");
                else if( p->pfunc == do_next )
                      printf("(NXT)");
                else if( p->pfunc == do_compare )
                      printf("(CMP)");
                else
#endif
//                if( p->pfunc != do_start && p->pfunc != do_stop && p->pfunc !=
do_swap && p->pfunc != do_for && p->pfunc != do_next && p->pfunc != do_compare )
                {
                    p->pfunc = do_stop;
                    p->nElement = 0;
                    p->next_depth = NULL;
                    p->next = NULL;
                    p->upper_depth = NULL;
                    tmp = 0;
#ifdef DEBUG
                    printf("Exception.. (STP)");
#endif
                }
                for(a=0;a<6;a++)
                      if( pEssential[a] == p->pfunc )
                            tmp = a;
                pvm_pkint( &(p->nElement), 1 ,1 );
                pvm_pkint( p->element, p->nElement, 1);
                pvm_pkint( &tmp, 1, 1);
                pvm_send(*(tids+j), 1   /*SENDTAG*/);

                if ( p->next_depth != NULL )
                {
                      p = p->next_depth;
                }
                else
                {
                      if( p->upper_depth == p->next && p->upper_depth !=
NULL)p = p->next->next;
                      else p = p->next;
                }
            }
            printf("\n");
            j++;
```

```
      }
}
```

////struct.h

```
/*struct stacks {
  int element[MAX_ELEMENT];
  int nElement;

   int intdata[100];
   int nData;

   struct stacks  *upper_depth; // 指向上一層
   struct stacks  *next;
   struct stacks  *former;
   // 針對 for 特別設計的
   struct stacks  *next_depth;

   struct stacks (*pfunc)(struct stacks *);

   int stack_size;
};*/

/* OTHER LIMITS DEFINATION 上限的定義 */
#define MAX_ELEMENT           4
#define MAX_INT_DATA          10
struct stacks
{
      int element[MAX_ELEMENT];     /* 暫存的元素 */
      int nElement;                 /* 暫存的元素數 */

      int intdata[MAX_INT_DATA];    /* 要 sorting 的資料 */
      int nData;                    /* 要 sorting 的資料有多大 */

      int arg_error;                /* 是否有發生參數錯誤 */

      struct stacks *upper_depth;   /* (巢狀迴圈) 上一層 */
      struct stacks *next_depth;    /* (巢狀迴圈) 下一層 */
      struct stacks *next;          /* 對於同一層的下一個節點 */
      struct stacks *former;        /* */
      struct stacks (*pfunc)(struct stacks *);  /* 欲執行的函數指標(function
pointer) */
};
```

////vmem.c

```
#define MAX_MEMORY_UNIT       100   /* 最大虛擬記憶體的 UNIT 數 */
```

```
int vmem[MAX_MEMORY_UNIT][2];        /* 宣告虛擬記憶體 */
int vmem_stack;                      /* 虛擬記憶體的上限紀錄 */
void vinit()
{
      int x;
      for(x=0; x<MAX_MEMORY_UNIT; x++)
      {
            vmem[x][0] = 0;
            vmem[x][1] = -1;
      }
      return;
}

int v_alloc(int info)
{
      if( vmem_stack >= MAX_MEMORY_UNIT )
            return -1;
      vmem[vmem_stack][0] = info;
      vmem[vmem_stack][1] = 0;
      return vmem_stack++;
}

int vread(int addr)
{
      // 該記憶體位置不存在, 未使用
      if( vmem[addr][1] == -1 )
            return -1;
      else
            return vmem[addr][0];
}

int vwrite(int addr, int info)
{
      // 未使用之位置..
      if( vmem[addr][1] == -1 )
            return -1;
      // 不可寫入..
      else if( vmem[addr][1] == 1 )
            return 0;
      else
      {
            vmem[addr][0] = info;
            return 1;
      }
}
```

```
int vfree(int addr)
{
      if( vmem[addr][1] == -1 )
            return -1;
      else
      {
            vmem[addr][1] = -1;
            vmem[addr][0] = 0;
            return 1;
      }
}
```

```
/* TTToe_ab
  created by Mau Fang  Shiu
  製作人: 徐茂芳
  alpha-beta search
*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>

#define MAX_SEARCH_DEPTH      3
#define MAP_WIDTH       3
#define MAX_PLAYERS           2

float doABSearch (int *, int, int);
int doResult (int *);
void doOccupy(int);
void mapDump(int *map);

int nowPlay;
int People[MAX_PLAYERS];

int mainMap[MAP_WIDTH*MAP_WIDTH] = {0};
char icon[] = {' ', 'O', 'X', 'A', 'B', 'C'};

int main (void)
{
      int Winner;
      People[0] = 1;
      srand(time(NULL));
      nowPlay = (int)rand()%MAX_PLAYERS+1;
      while( !(Winner=doResult(mainMap)) ) {
            int x;
            nowPlay++;
#ifdef DEBUG
            printf("Turns on Player %d(", nowPlay%MAX_PLAYERS);
            for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
                  printf("%d", mainMap[x]);
            }
            printf(")\n");
#endif
            doOccupy(nowPlay%MAX_PLAYERS);
      }
      mapDump(mainMap);
      printf("Player: %d won.\n", Winner);
      return 0;
}
```

```
void doOccupy(int player)
{
	int x, candidate[MAP_WIDTH*MAP_WIDTH], canMax=0, realCan;
	float toChoose[MAP_WIDTH*MAP_WIDTH], max;
	if( !People[player] ) {
		for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
			if( !mainMap[x] ) {
				toChoose[x] = doABSearch(mainMap, MAX_SEARCH_DEPTH,
player);
			}
			else {
				toChoose[x] = -2;
			}
		}
#ifdef DEBUG
		printf("toChoose dump:");
		for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
			printf(" %f", toChoose[x]);
		}
		printf("\n");
#endif
		for(max=-10, x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
			if( toChoose[x] > max )
				max = toChoose[x];
		}
		for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
			if( toChoose[x] == max )
			{
				candidate[canMax++] = x;
			}
		}
#ifdef DEBUG
		printf("Candidates:");
		for(x=0; x<canMax; x++)
			printf("%d", candidate[x]);
		printf("\n");
#endif
		realCan = rand()%canMax;
		mainMap[candidate[realCan]] = player+1;
		mapDump(mainMap);
	}
	else {
		int position;
		while(1) {
			mapDump(mainMap);
```

```
                printf("Please enter what position would you like: ");
                scanf("%d", &position);
                if( position < 0 || position > MAP_WIDTH*MAP_WIDTH ||
mainMap[position] ) {
                      printf("Wrong Position!! Agian...\n");
                }
                else {
                      mainMap[position] = player+1;
                      break;
                }
          }
     }
}

float doABSearch (int *map, int depth, int flag)
{
     int x, y, cntX, count;
     float ttlChance;
     float chanceWin[MAP_WIDTH*MAP_WIDTH];

     // Init
     if( depth <= 0 ) {
          return 0;
     }

     // Debug
#ifdef DEBUG
     printf("p%d dep%d (", flag, depth);
     for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
          printf("%d", map[x]);
     }
     printf(")\n");
#endif
     // End of Debug

     for(cntX=0, x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
          chanceWin[x] = 0;
          if( !map[x] ) {
                cntX++;
          }
     }

     if( (x=doResult(map)) ) {
          if( x == flag )
                return cntX*(cntX-1);
          else if( x == MAX_PLAYERS+1 )
```

```
                return 0;
            else
                return -1*cntX*(cntX-1);
        }

        if( cntX < MAX_PLAYERS ) {
#ifdef DEBUG
            printf("xxxxxxxxxxxxxxxxxxxx HERE xxxxxxxxxxxxxxxxxxxx\n");
#endif
            // 暫時做成 for 2 players
            return 0;
        }
        // end of Init

        // Search
        for(x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
            if( !map[x] ) {
                map[x] = flag;
                for(y=0; y<(MAP_WIDTH*MAP_WIDTH); y++) {
                    if( map[y] )
                        continue;
                    if( flag == 1 )
                        map[y] = 2;
                    else
                        map[y] = 1;
                    chanceWin[x] += doABSearch(map, depth-1, flag);
                    map[y] = 0;
                }
                map[x] = 0;
            }
        }
        // end of Search

        // Return
        for(ttlChance=0, x=0; x<(MAP_WIDTH*MAP_WIDTH); x++) {
            if( chanceWin[x] > ttlChance ) {
                ttlChance += chanceWin[x];
            }
        }
        ttlChance = ttlChance/(cntX*(cntX-1));
        return ttlChance;
        // end of Return
}

// Return value List:
//    0: Not finished
```

```
//    1: Player 1 won
//    2: Player 2 won
//    3: Equal
int doResult (int *map)
{
      int x, y, first, count, win;

      win = 0;

      for(count=0, x=0; x<(MAP_WIDTH*MAP_WIDTH); x++)
            if( map[x] == 0 )
                  count++;
      // 1 0 0 0
      // 0 1 0 0
      // 0 0 1 0
      // 0 0 0 1
      if( !win ) {
#ifdef DEBUG
//          printf("\\");
#endif
            for(first=map[0], x=0; x<MAP_WIDTH ; x++) {
                  if( map[x*MAP_WIDTH+x] == first ) {
                        win = first;
                  }
                  else {
                        win = 0;
                        break;
                  }
            }
#ifdef DEBUG
            if( win )
                  printf("(%d Win)", win+1);
#endif
      }
      // 0 0 0 1
      // 0 0 1 0
      // 0 1 0 0
      // 1 0 0 0
      if( !win ) {
#ifdef DEBUG
//          printf("/");
#endif
            for(first=map[MAP_WIDTH-1], x=0; x<MAP_WIDTH; x++) {
                  if( map[(x+1)*(MAP_WIDTH-1)] == first ) {
                        win = first;
                  }
```

```
                else {
                        win = 0;
                        break;
                }
            }
#ifdef DEBUG
            if( win )
                printf("(%d Win)", win+1);
#endif
        }
        // 1 1 1 1
        // 2 2 2 2
        // 3 3 3 3
        // 4 4 4 4
        if( !win ) {
#ifdef DEBUG
//          printf("-");
#endif
            for(x=0; x<MAP_WIDTH && !win; x++) {
                for(first=map[x*MAP_WIDTH], y=0; y<MAP_WIDTH; y++) {
                    if( map[x*MAP_WIDTH+y] == first ) {
                        win = first;
                    }
                    else {
                        win = 0;
                        break;
                    }
                }
            }
#ifdef DEBUG
            if( win )
                printf("(%d Win)", win+1);
#endif
        }
        // 1 2 3 4
        // 1 2 3 4
        // 1 2 3 4
        // 1 2 3 4
        if( !win ) {
#ifdef DEBUG
//          printf("|");
#endif
            for(x=0; x<MAP_WIDTH && !win; x++) {
                for(first=map[x], y=1; y<MAP_WIDTH; y++) {
                    if( map[x+y*MAP_WIDTH] == first ) {
                        win = first;
```

```
                        }
                        else {
                              win = 0;
                              break;
                        }
                  }
            }
#ifdef DEBUG
            if( win )
                  printf("(%d Win)", win+1);
#endif
      }
#ifdef DEBUG
      printf("\n");
#endif

      if( win )
            return win;
      // Equal
      else if( count == 0 ) {
            return 3;
      }
      // Not finished.
      else {
            return 0;
      }
}

void mapDump(int *map)
{
      int x, y;
      printf("<---------------------->\n");
      for(x=0; x<MAP_WIDTH; x++) {
            for(y=0; y<MAP_WIDTH-1; y++) {
                  printf("%c|", icon[map[y+x*MAP_WIDTH]]);
            }
            printf("%c\n", icon[map[y+x*MAP_WIDTH]]);
            if( x < MAP_WIDTH-1 ) {
                  for(y=0; y<MAP_WIDTH-1; y++) {
                        printf("-+");
                  }
                  printf("-\n");
            }
      }
      printf("<---------------------->\n");
}
```

```
/* GPTicTacToe
  created by Wei-Chiu Chuang
  製作人: 莊偉赳

*/

////// config.h
#include <stdio.h>
#include <pvm3.h>

#define USED      1
#define UNUSED    2
#define POPULATION      5
#define POPPICKUP 2
#define MAXLENGTH 100

#define MAXLOOP         100

#define FUNC_NONE       0
#define FUNC_IF                 1
#define FUNC_ELSE               2
#define FUNC_ENDIF              3
#define FUNC_WHILE              4
#define FUNC_OCCUPY             5
#define FUNC_IS_VACANCY         6
#define FUNC_IS_OCCUPY          7
#define FUNC_IS_FINISHED        8


struct _DATA {
      int func_type;
      int parameter[2];
};

struct _PROG {
 // save these "product program" details
      int pLength;
      int pStatus; // USED represent used, UNUSED represent unused
};


/////Gpmonitor.cpp
#include <kapplication.h>
#include "my.h"
/***************
 * 這個程式是拿來監控 PVM 群組內的狀態, 需要各
```

```
 * child 配合, 在進入新的 function 時傳送給 monitor program
 * 訊息, 才知道目前狀態(進入那個 function)
 * ************/
int GPmonitoring(int,char **);

int main(int argc,char **argv){
      KApplication app(argc,argv,"GPPVMmonitor" );
      myKmonitor *my = new myKmonitor();
      my->show();
      app.exec();
      GPmonitoring(argc,argv);

      //pvm_lvgroup( argv[1] );
}
int GPmonitoring(int argc,char **argv ){
      // join group



}


/////Gpslave.cpp
#include "config.h"
#include <unistd.h>

#define  ROUND_TIMES    100


extern int MAP[9];
extern int do_if( struct _DATA *);
extern int do_else( struct _DATA * );
extern int do_endif( struct _DATA *);
extern int do_while( struct _DATA *);
extern int do_occupy( struct _DATA *);
extern int do_is_vacancy( struct _DATA *);
extern int do_is_occupy( struct _DATA *);
extern int do_is_finished( struct _DATA *);

extern int mypvmgid,mypvmtid,mtid;
extern struct _PROG progdata[POPULATION];
extern struct _DATA prog[POPULATION][MAXLENGTH];
int tellcount=0;
int TELL( char *);

int GPsend_statement(int);
```

```
int GPcompete_init();
int GPcompete_start();
int GPcombat();
int GPscore();

int GPslave(){
char pvmgroupname[32];
int seqtmp[MAXLENGTH*3],datalength;
int i;
int perf=0;
      TELL( "GPslave()" );
      pvm_recv( pvm_parent(), 1 );
      pvm_upkstr( pvmgroupname );
      TELL( "upk group name" );
      mypvmgid = pvm_joingroup( pvmgroupname );
// recv from GPjudgee
      // get data length
      pvm_recv( pvm_parent(), 2 );
      pvm_upkint( &datalength,1,1);
      TELL( "upk data length" );
      // get monitor's tid
      pvm_recv( pvm_parent(), 3 );
      pvm_upkint( &mtid , 1, 1);
      TELL( "upk monitor tid " );

      // get data , $length size long
      pvm_recv( pvm_parent(), 4 );
      pvm_upkint(seqtmp , datalength ,1  );
      TELL( "upk data" );

      char slength[32];
      sprintf(slength, "Data size->%d", datalength );
      TELL( slength );
      TELL( "OK!" );
      progdata[0].pLength = datalength/3;
      progdata[0].pStatus = USED;
      for(i=0;i<datalength;i+=3){
            prog[0][i/3 ].func_type   =seqtmp[i  ];
            prog[0][i/3 ].parameter[0]=seqtmp[i+1];
            prog[0][i/3 ].parameter[1]=seqtmp[i+2];

            GPsend_statement( seqtmp[i] );
      }
      /*******
       *  最難 coding 的部份來了
       *  要執行這個程式, 並且要再寫一個 AI 程式跟它 PK
```

```
        *  -->不知道要 PK 多少場??
        *  Vmem 長度設定爲 9
        * *****/
       for(i=0;i<ROUND_TIMES;i++){
             GPcompete_init();
             perf += GPcompete_start();
       }
       /*******
        *  完成比賽::
        *   送回評分資料
        *   準備離開
        * *****/
       pvm_initsend(PvmDataDefault);
       pvm_pkint(&mypvmgid,1,1);
       pvm_pkint(&perf, 1,1);
       pvm_send( pvm_parent(), 200 );


       pvm_lvgroup( pvmgroupname );
       pvm_exit();

       return 0;
}
int GPcombat(){
// use common AI algorithm to play
       // I can't spend time writing AI program statement, so
       // write ugly but instant one instead.
       if( MAP[4] == 0 ){
                   MAP[4] = 2;
             return 1;
       }
       if( MAP[0] == 0 ){
             MAP[0] = 2;
//           return 1;
       }
       if( MAP[2] == 0 ){
             MAP[2] = 2;
//           return 1;
       }
       if( MAP[6] == 0 ){
             MAP[6] = 2;
//           return 1;
       }
       if( MAP[8] == 0 ){
             MAP[8] = 2;
//           return 1;
```

```
	}
	if( MAP[1] == 0 ){
		MAP[1] = 2;
//		return 1;
	}
	if( MAP[3] == 0 ){
		MAP[3] = 2;
//		return 1;
	}
	if( MAP[5] == 0 ){
		MAP[5] = 2;
//		return 1;
	}
	if( MAP[7] == 0 ){
		MAP[7] = 2;
//		return 1;
	}
	return 0;
}

int GPcompete_init(){
int i;
	//TELL( "GPcompete_init()");
	for(i=0;i<9;i++) MAP[i] = 0;
	return 0;

}
int GPcompete_start(){
int pos=0;
	//TELL( "GPcompete_start()");
//#ifdef LALALA
	char aaa[32];
//	sprintf(aaa, "pLength=%d", progdata[0].pLength);
//	TELL(aaa);
	while(pos < progdata[0].pLength  ){
		switch ( prog[0][pos].func_type ){
			case ( FUNC_IF ):
			//	TELL("-->do_if()");
				int tmp;
				tmp = do_if( &prog[0][pos] );
				if( tmp > 0 ){  // true
					/*  Go to next statement */
					pos++;
				}else{
					/* Find tag 'ENDIF'  */
```

```
                              while(  (prog[0][pos].func_type != FUNC_ELSE &&
prog[0][pos].func_type != FUNC_ENDIF) && pos < progdata[0].pLength )
                                    pos++;
                        }
                        break;
                  case ( FUNC_ELSE ):
//                      TELL("-->do_else()");
                        /*  Only a tag 僅是一個標籤  */
                        pos++;
                        break;
                  case ( FUNC_ENDIF ):
//                      TELL("-->do_endif()");
                        /*  Only a tag 僅是一個標籤  */
                        pos++;
                        break;
                  case ( FUNC_WHILE ):
//                      TELL("-->do_while()");
                        pos++;
                        break;
                  case ( FUNC_OCCUPY ):
                        TELL("-->do_occupy()");
                        do_occupy( &prog[0][pos] );
      /*  **********************
       *  重頭戲來了!!! 當產生出來的基因程式下了這一步後
       *  (也就是在虛擬記憶體中的棋盤下了一子之後)
       *  就切換到對戰者 AI 系統來下~~~~~ 然後再將控制權轉
       *  回基因程式
       *  **********************/
                        if( GPcombat() == 1 || GPcombat() == 2 ){
                              // GP win or AI win
                              goto outter;
                        }
                        pos++;
                        break;
                  case ( FUNC_IS_VACANCY ):
//                      TELL("-->do_is_vacancy()");
                        //do_is_vacancy ( prog[0][pos] );
                        pos++;
                        break;
                  case ( FUNC_IS_OCCUPY ):
//                      TELL("-->do_is_occupy()");
                        //do_is_occupy  ( prog[0][pos] );
                        pos++;
                        break;
                  case ( FUNC_IS_FINISHED ):
                        //do_is_finished( prog[0][pos] );
```

```
//                      TELL("-->do_is_finished()");
                        if( do_is_finished( &prog[0][pos]) ) goto outter;
                        /*
                         *  檢查是不是已經走了九格全滿了
                         * */
                        pos++;
                        break;
                default:
                        TELL("ERROR!!!");
                        sleep(1);
                        break;
            }
      }
//#endif
      usleep( 500 );
      return 0;
outter:
      usleep( 500 );
      return GPscore();
}

int GPscore(){
  if(       (MAP[0] == MAP[1] && MAP[1] == MAP[2] && MAP[0] == 1)  ||
            (MAP[3] == MAP[4] && MAP[4] == MAP[5] && MAP[3] == 1)  ||
            (MAP[6] == MAP[7] && MAP[7] == MAP[8] && MAP[6] == 1)  ||
            (MAP[0] == MAP[3] && MAP[3] == MAP[6] && MAP[6] == 1)  ||
            (MAP[1] == MAP[4] && MAP[4] == MAP[7] && MAP[7] == 1)  ||
            (MAP[2] == MAP[5] && MAP[5] == MAP[8] && MAP[8] == 1)  ||
            (MAP[0] == MAP[4] && MAP[4] == MAP[8] && MAP[0] == 1)  ||
            (MAP[2] == MAP[4] && MAP[4] == MAP[6] && MAP[2] == 1)     ){

        return 10;
  }
  else if( (MAP[0] == MAP[1] && MAP[1] == MAP[2] && MAP[0] == 2)  ||
            (MAP[3] == MAP[4] && MAP[4] == MAP[5] && MAP[3] == 2)  ||
            (MAP[6] == MAP[7] && MAP[7] == MAP[8] && MAP[6] == 2)  ||
            (MAP[0] == MAP[3] && MAP[3] == MAP[6] && MAP[6] == 2)  ||
            (MAP[1] == MAP[4] && MAP[4] == MAP[7] && MAP[7] == 2)  ||
            (MAP[2] == MAP[5] && MAP[5] == MAP[8] && MAP[8] == 2)  ||
            (MAP[0] == MAP[4] && MAP[4] == MAP[8] && MAP[0] == 2)  ||
            (MAP[2] == MAP[4] && MAP[4] == MAP[6] && MAP[2] == 2)     ){

        return -10;
  }
  else{
        return 0;
```

```
  }

}
int GPcompete(){

      return 0;
}
int GPsend_statement( int s ){
char slength[64];
            switch ( s ){
                  case( FUNC_IF ):
                        TELL( "IF" );
                        break;
                  case( FUNC_ELSE ):
                        TELL( "ELSE" );
                        break;
                  case( FUNC_ENDIF ):
                        TELL( "ENDIF" );
                        break;
                  case( FUNC_WHILE ):
                        TELL( "WHILE" );
                        break;
                  case( FUNC_OCCUPY ):
                        TELL( "OCCUPY" );
                        break;
                  case( FUNC_IS_VACANCY ):
                        TELL( "IS_VACANCY" );
                        break;
                  case( FUNC_IS_OCCUPY ):
                        TELL( "IS_OCCUPY" );
                        break;
                  case( FUNC_IS_FINISHED ):
                        TELL( "IS_FINISHED" );
                        break;
                  default:
                        TELL( "ERROR!!" );
                        sprintf(slength, "id=>%d",s);
                        TELL( slength );
                        break;
            }

      return 0;
}

int TELL( char *string ){
char tmp[64];
```

```
	++tellcount;
	sprintf(tmp, "[%d].%s",tellcount,string);
	pvm_initsend( PvmDataDefault );
	pvm_pkint ( &mypvmgid ,1,1);
	pvm_pkstr( tmp );
	pvm_send( mtid , 100);

	return 0;
}

/////main.cpp
#include "config.h"
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
int MAP[9];
struct _PROG progdata[POPULATION], selecteddata[POPPICKUP];
struct _DATA prog[POPULATION][MAXLENGTH], selectedprog[POPPICKUP][MAXLENGTH];
int rank[POPULATION];

struct _JUDGEPERF{
	int i;
	int perf;
};
struct _JUDGEPERF judge_perf[POPULATION];

/* **********************
 *
 * among 20 instances, the best 4 instances are permitted creating
 * new offsprings(each produce 5)
 *   --> 1. copy these instances to temporialy memory
 *       2. clean the memory occupied before.
 *       3. mutate & crossover these 4 instances, and then copy to the tree mem.
 * , whereas the others are abandoned.
 *
 * ********************** */

extern int GPslave();

int GPinit();
int GPgenerate( int,int);
int GPselect(int );
int GPabandon(int, int );
int GPcrossover( );
```

```
int GPmutate();
int GPcopy();
int GPfinished();
int GPjudge();
int GPbenchmark();
int GPcheck();
int GPassign(int,int,int);
int GPqsortcmp( const void *, const void *);

extern int do_if();
extern int do_else();
extern int do_endif();
extern int do_while();
extern int do_occupy();
extern int do_is_vacancy();
extern int do_is_occupy();
extern int do_is_finished();

extern int TELL( char *);

int mypvmtid,mypvmgid;

char pvmgroupname[32];
int mtid;
int count=0;

int main( int argc, char *argv[] ){
int i;
int judge_result;
struct timeval nowtime;
struct timezone tz;

      printf("Enter GP Tic-Tac-Toe\n");
      mypvmtid = pvm_mytid();
      printf("Load PVM\n");
      srand( time( NULL ) );
      if( pvm_parent() == PvmNoParent ){
            tz.tz_dsttime = 0; tz.tz_minuteswest = 1;
            gettimeofday( &nowtime, &tz );
            sprintf( pvmgroupname, "GPTTT%d", (int)nowtime.tv_sec );
            printf("Join group ( %s )no. %d \n",pvmgroupname,mypvmgid);
            mypvmgid = pvm_joingroup( pvmgroupname );

            printf("Start monitoring program\n");
            pvm_spawn( "/home/jojochuang/SF/GPTicTacToe/monitor",(char **)NULL,
PvmTaskDefault,(char *)NULL, 1, &mtid );
```

```
            pvm_initsend( PvmDataDefault );
            pvm_pkstr( pvmgroupname );
            pvm_send( mtid,  1 );
            printf("Wait for monitor to activate\n");
            sleep(3);
            printf("initialize GP..\n");
            GPinit();
            for(i=0;i<POPULATION;i++)
                  GPgenerate( 0 ,i);
            while( GPfinished() == 0 ){
                  printf("loop (%d)\n", count );
                  judge_result = GPjudge();
                  if( judge_result < 0 ){// didn't find best solution
                        GPbenchmark(); // get,sort program performance;
                        /**  should be ignore.. */
                        GPselect( POPPICKUP );// preserve
                        /**  ignore above!!  */
                        GPabandon( POPULATION, POPPICKUP);//remove,clean,kill)
                        GPcrossover();// crossover, and also copy to child
                        GPmutate();   // modify produced child
                        GPcopy();     // directly copy parents to child
                  }// continue....
                  else{
                    // judge >=0 means got it
                        printf("Find answer...\n");
                        break;
                  }
            }
            pvm_lvgroup(pvmgroupname);
            printf("Leave group..\n");
      }else{
            //TELL("I'm slave");
            GPslave();
      }
}
int GPcopy(){
int i,j,parent;
      printf("Entering GPcopy()\n");

      for(i=0;i<POPULATION;i++){
            if( progdata[i].pStatus == UNUSED ){
                  parent = (int) (  (POPPICKUP-1)*rand()/(RAND_MAX+1.0));
                  progdata[i].pStatus = USED;
                  progdata[i].pLength = selecteddata[parent].pLength;

                  printf("[複製]從%d 到%d\n",parent,i);
```

```
                for( j=0;j< selecteddata[parent].pLength;j++){
                      prog[i][j].parameter[0] =
selectedprog[parent][j].parameter[0];
                      prog[i][j].parameter[1] =
selectedprog[parent][j].parameter[1];
                     prog[i][j].func_type = selectedprog[parent][j].func_type;
                }
           }
      }

      return 0;
}
int GPinit(){
int i,j;
      printf("Entering GPinit()\n");
      for(i=0;i<9;i++)MAP[i] = 0;
      for(i=0;i<POPULATION;i++){
           progdata[i].pLength = 0;
           for(j=0;j<MAXLENGTH;j++){
                prog[i][j].func_type=-1;

           }
      }
      for(i=0;i<POPPICKUP;i++){
           selecteddata[i].pLength = 0;
           for(j=0;j<MAXLENGTH;j++){
                prog[i][j].func_type=-1;
           }
      }
      for(i=0;i<POPULATION;i++)rank[i] = 0;

      return 0;
}

int GPfinished(){
      // verify if it is finished
      /*if ( GPcheck() ) return 1;
      int i,tmp;
      for(i=0;i<9;i++)if( MAP[i] != 0 )tmp++;
      if( tmp == 9 )return 1;// all vacancy are occupied

      return 0;// nothing happened, then, send 0 instead.
      */
      if( count++ > MAXLOOP ){
           printf("執行超過%d 次，中斷\n",MAXLOOP);
           return 1;
```

```
	}
	else return 0;
}

int GPqsortcmp(const void *a, const void *b){
	struct _JUDGEPERF x = *(struct _JUDGEPERF *)a;
	struct _JUDGEPERF y = *(struct _JUDGEPERF *)b;
	return x.perf - y.perf;

}
int GPbenchmark(){
int i,j;
	// sort performance
	printf("Entering GPbenchmark\n");
	qsort( judge_perf   ,POPULATION, sizeof(int)*2 ,GPqsortcmp  );
	printf("得分回傳資料排名：\n");
	printf("--------------------\n");
	for(i=0;i<POPPICKUP;i++){
		printf("第%d 名：( %d )得%d 分
\n",i,judge_perf[i].i,judge_perf[i].perf);
	}
	printf("--------------------\n");
	printf("複製%d 個到 selected[]\n",POPPICKUP);
	for(i=0;i<POPPICKUP;i++){
		selecteddata[i].pLength = progdata[ judge_perf[i].i ].pLength;
		selecteddata[i].pStatus = USED;
		for(j=0;j< selecteddata[i].pLength;j++){
			selectedprog[i][j].func_type    =
prog[ judge_perf[i].i ][j].func_type;
			selectedprog[i][j].parameter[0] =
prog[ judge_perf[i].i ][j].parameter[0];
			selectedprog[i][j].parameter[1] =
prog[ judge_perf[i].i ][j].parameter[1];
		}
	}
	return 0;

}
#define INITLENGTH      5
/***********************
 * generate INITLENGTH statement
 *
 ***********************/
int GPgenerate( int s,int instance){
	int i,j,tmp;
	printf("Entering GPgenerate()\n");
```

```
	//for(i=0;i<POPULATION;i++){
		progdata[instance].pLength= INITLENGTH;
		for( j=0;j<INITLENGTH;j++){
			tmp = 1+(int) (8.0*rand()/(RAND_MAX+1.0)); // randomly produce
statement
			prog[instance][j].func_type = tmp;
			//printf("-->%d\n", prog[instance][j].func_type );
			GPassign( instance,j,tmp );
		}
	//}

	return 0;
}
int GPassign( int i,int j, int function_type){
int tmp;
	switch ( function_type ){
		case FUNC_IF:
			tmp =(int) (2.0*rand()/(RAND_MAX+1.0));
			prog[i][j].parameter[0] = tmp;
			switch( tmp ){
				case 0:
					prog[i][j].parameter[1] =(int)
(9.0*rand()/(RAND_MAX+1.0));

					break;
				case 1:
					prog[i][j].parameter[1] =(int)
(9.0*rand()/(RAND_MAX+1.0));
					break;
				case 2:
					break;
			}
			break;
		case FUNC_ELSE:
			break;
		case FUNC_ENDIF:
			break;
		case FUNC_WHILE:
			break;
		case FUNC_OCCUPY:
			tmp =(int) (9.0*rand()/(RAND_MAX+1.0));
			prog[i][j].parameter[0]= tmp;
			break;
		case FUNC_IS_VACANCY:
			break;
		case FUNC_IS_OCCUPY:
```

```
                  break;
            case FUNC_IS_FINISHED:
                  break;
      }
}

int GPselect( int total){
//    printf("Entering GPselect()\n");
      return 0;
}

int GPabandon( int population, int remain){
int i,j;
      printf("Entering GPabandon()\n");
      for(i=0;i<POPPICKUP;i++){

      }
      for(i=POPPICKUP;i<POPULATION;i++){
            printf("清除%d 資料\n",judge_perf[i].i );
            progdata[ /*rank[i]*/ judge_perf[i].i  ].pStatus = UNUSED;
            progdata[ /*rank[i]*/ judge_perf[i].i  ].pLength = 0;
            for(j=0;j<MAXLENGTH;j++)
                  prog[ /*rank[i]*/judge_perf[i].i ][j].func_type = FUNC_NONE;
      }
      return 0;
}
int GPcrossover( ){ /*********  UNFINISHED ***********/
int i;
int mom,dad;
int pos1,pos2;// from where we crossover them
struct _DATA tempdata[MAXLENGTH];
      printf("Entering GPcrossover()\n");
      mom=(int) (4.0*rand()/(RAND_MAX+1.0));
      dad=(int) (4.0*rand()/(RAND_MAX+1.0));

      pos1 = (int) ((float)(selecteddata[mom].pLength-1)
*rand()/(RAND_MAX+1.0));
      pos2 = (int) ((float)(selecteddata[dad].pLength-1)
*rand()/(RAND_MAX+1.0));

      for(i=pos1;i<selecteddata[mom].pLength;i++)
            tempdata[i-pos1].func_type = selectedprog[mom][i].func_type;


      printf("[交配](未完成)\n");
```

```
      return 0;
}
int GPmutate(){/*********  UNFINISHED ***********/
int i,j;
int who;
int pos;
int total_to_mutate;
int become;
int total_unused=0;
int unused[POPULATION];
struct _DATA child;
      printf("Entering GPmutate()\n");
      // decide how many child we want to born&mutate
      for( i=0;i< POPULATION; i++)
            if( progdata[i].pStatus == UNUSED ){
                  unused[total_unused] = i;
                  total_unused++;
            }
      total_to_mutate = (int)( (float)total_unused*rand()/(RAND_MAX+1.0));
      for(i=0;i<total_to_mutate;i++){
            who    = (int) (4.0*rand()/(RAND_MAX+1.0));
            pos    = (int) ((float)( selecteddata[who].pLength-
1   )*rand()/(RAND_MAX+1.0));
            become = 1+(int) ( 8.0 *rand()/(RAND_MAX+1.0));
            for(j=0;j< selecteddata[who].pLength; j++){
                  prog[ unused[i] ][ j ].func_type =
selectedprog[who][j].func_type;
                  prog[ unused[i] ][ j ].parameter[0] =
selectedprog[who][j].parameter[0];
                  prog[ unused[i] ][ j ].parameter[1] =
selectedprog[who][j].parameter[1];
            }
            prog[ unused[i] ][ pos ].func_type = become;
            GPassign( unused[i], pos, become );
            printf("[變異複製]從 who=%d 到%d, pos=%d,
become=%d\n",who,unused[i],pos,become);
      }
      return 0;
}

int GPjudge(){
// play & judge
      // how to do it? Use PVM, distribute these statement, execute them
      // on the spawned processes.
int i,j,tmplength,nperf;
int tids[POPULATION];
```

```
	printf("Entering Judge stage. Spawn process\n");
	/// spawn..
	pvm_spawn( "/home/jojochuang/SF/GPTicTacToe/tttoe",(char **)NULL,
PvmTaskDefault,(char *)NULL, POPULATION, tids );
	printf("end spawn\n");
	pvm_initsend( PvmDataDefault );
	pvm_pkstr( pvmgroupname );

	for(i=0;i<POPULATION;i++)pvm_send( tids[i], 1 );

	int seqtmp[MAXLENGTH*3];
	for(i=0;i<POPULATION;i++){
		tmplength =progdata[i].pLength*3;
		pvm_initsend(PvmDataDefault);
		pvm_pkint(&tmplength, 1,1);
		pvm_send( tids[i], 2 );
		// save the data to a continuous temporaily place
		for( j=0;j<  progdata[i].pLength ; j++){
			seqtmp[ j*3 ] = prog[i][j].func_type;
			seqtmp[j*3+1] = prog[i][j].parameter[0];
			seqtmp[j*3+2] = prog[i][j].parameter[1];
		}
		pvm_initsend(PvmDataDefault);
		pvm_pkint( &mtid, 1,1 );
		pvm_send( tids[i], 3 );

		pvm_initsend(PvmDataDefault);
		pvm_pkint( seqtmp, progdata[i].pLength*3, 1 );
		pvm_send( tids[i], 4 );
		printf("send to child %d\n",i);
	}
	/* 開始等待資料輸入.......過程:
	 *   slave 建構程式段落
	 *   開始執行對家程式,進行 PK
	 *   多場過後, 評分
	 *   將分數送給 GPjudge()
	 * */
	printf("等待資料送回\n");
	for(i=0;i<POPULATION;i++){
		pvm_recv( -1, 200 );
		pvm_upkint( &nperf , 1, 1);
		pvm_upkint( &judge_perf[ nperf -1 ].perf, 1, 1);
		judge_perf[nperf-1].i = nperf-1;
		printf("收到來自%d，得分%d\n", nperf, judge_perf[ nperf-1 ].perf );
	}
```

```
      printf("exit GPjudge\n");
      return -1;
}


int GPcheck(){
  if( MAP[0] == MAP[1] && MAP[1] == MAP[2] && MAP[0] != 0 )
        return 1;
  if( MAP[3] == MAP[4] && MAP[4] == MAP[5] && MAP[3] != 0 )
        return 2;
  if( MAP[6] == MAP[7] && MAP[7] == MAP[8] && MAP[6] != 0 )
        return 3;

  if( MAP[0] == MAP[3] && MAP[3] == MAP[6] && MAP[6] != 0 )
        return 4;
  if( MAP[1] == MAP[4] && MAP[4] == MAP[7] && MAP[7] != 0 )
        return 5;
  if( MAP[2] == MAP[5] && MAP[5] == MAP[8] && MAP[8] != 0 )
        return 6;

  if( MAP[0] == MAP[4] && MAP[4] == MAP[8] && MAP[0] != 0 )
        return 7;
  if( MAP[2] == MAP[4] && MAP[4] == MAP[6] && MAP[2] != 0 )
        return 8;
  return 0;
}
/////vfunctions.cpp
#include "config.h"


#define TYPE_BOOLEAN          1
#define TYPE_INT        2
#define TYPE_VOID       3
extern int MAP[9];

extern int GPcheck();
extern int TELL( char * );
extern int GPcombat();


int do_if(struct _DATA *);
int do_else(struct _DATA *);
int do_endif(struct _DATA *);
int do_while(struct _DATA *);
int do_occupy(struct _DATA *);
int do_is_vacancy(struct _DATA *);
```

```
int do_is_occupy(struct _DATA *); //done...
int do_is_finished(struct _DATA *);


/* ****************************** */
int do_if( struct _DATA *data){
// 判斷式中也是一個小程式(形成一個小 function)?? >><<不要
// parameter[0] 是型態
      // ---> 0 indicates do_is_vacancy()
      // ---> 1 indicates do_is_occupy()
      // ---> 2 indicates do_is_finished();??
// parameter[1] 是內容
      // ---> 目前僅用在 do_is_occupy()
      // ---> 表示那個區域被佔用了
      switch ( data->parameter[0] ){
            case 0:
                  return do_is_vacancy( data );
                  break;
            case 1:
                  return do_is_occupy(data);
                  break;
            case 2:

                  return do_is_finished(data);
                  break;
      }
      return 0;
}

int do_else( struct  _DATA *data){

      /* 這僅是個標籤 */
      return 0;
}

int do_endif( struct  _DATA *data){

      /* 這僅是個標籤 */
      return 0;
}

/* ****************************** */
int do_occupy( struct  _DATA *data){
      // return 0: ok
      // return 1: already occupied by player1
      // return 2: already occupied by player2
```

```
	// data->parameter[0] is the position
	if( do_is_occupy(data) ==0 ){
		// the generated program is player 1
		MAP[data->parameter[1]] = 1;
	}
	return 0;
}

int do_is_vacancy( struct _DATA *data){
	if( MAP[data->parameter[1] ]==0 )
		return 1;
	else
		return 0;
	//return MAP[data->parameter[0]];
}

/* ****************************** */
int do_is_occupy( struct  _DATA *data){
 // return 0: not occupied.
	// 1: occupied by player1
	// 2: occupied by player2
	// data->parameter[0] is the position
	return MAP[data->parameter[1]];
}

int do_is_finished( struct  _DATA *data){
	/*if( GPcheck() > 0 )
		return 1;
	return 0;
	*/
int i,count=0;
	for(i=0;i<9;i++)if( MAP[i] != 0 ) count++;
	if( count == 9 ) return 1;
	return 0;
}
```

////my.h

```
#include<kmainwindow.h>
#include<keditcl.h>
#include<qstring.h>
#include<qtextstream.h>
#include<qpushbutton.h>
#include "config.h"
class myKmonitor:public KMainWindow
```

```
{
      Q_OBJECT
public:
      myKmonitor( QWidget *parent=0, const char *name=0 );
      ~myKmonitor();

public slots:
      void dstart();
private:
      void showEvent ( QShowEvent * );
      void timerEvent( QTimerEvent *);
      int init();
      KEdit *logedit[POPULATION];
      char *pvmgroupname;
};

///// my.cpp
#include"my.h"
//#include "config.h"

myKmonitor::myKmonitor( QWidget *parent, const char *name )
      :KMainWindow(parent, name )
{
      init();
}

myKmonitor::~myKmonitor(){
int i;
      for(i=0;i<POPULATION;i++)
            delete logedit[i];

      pvm_lvgroup( pvmgroupname );
      pvm_exit();
}
int myKmonitor::init(){
int i;
      this->resize( 1000,400);
      for(i=0;i<POPULATION;i++){
            logedit[i] = new KEdit( this );
            logedit[i]->move(200*(i%5) ,100*( i - i%5 )/5);
            logedit[i]->resize(200,400);
      }
      pvmgroupname = new char[32];
      /*qbutton = new QPushButton( this);
      qbutton->setText( "Test" );
      qbutton->setGeometry( 0,0,20,20 );
```

```
	connect( qbutton, SIGNAL( clicked() ), this, SLOT( dstart() ) );
	*/
	return 1;
}
void myKmonitor::showEvent( QShowEvent *e ){
//char pvmgroupname[32];
int tmp;
int mytid,mygid;

	QWidget::showEvent(e);
	mytid = pvm_mytid();
	//mygid = pvm_joingroup(argv[1]  );
	pvm_recv( pvm_parent() , -1 );
	pvm_upkint( &tmp, 1, 1);
	sprintf( pvmgroupname, "GPTTT%d", tmp );
	pvm_joingroup( pvmgroupname );
	startTimer( 100 );
}
void myKmonitor::timerEvent(QTimerEvent *e){
int bufid,msgtag=100;
int pvmwhere;
char pvmfunction[32];
char templog[40];
	QWidget::timerEvent(e);
	while(1){
		bufid = pvm_nrecv( -1, msgtag );
		if( bufid == 0 ) break;
		pvm_upkint( &pvmwhere, 1, 1);
		pvm_upkstr( pvmfunction );

		sprintf(templog, "%s\n",pvmfunction );

		QString     x = templog;
		QTextStream y( &x, IO_ReadWrite );

		if( pvmwhere >= 1 && pvmwhere <= 20 )
			logedit[pvmwhere-1]->insertText( &y  );
	}
}
void myKmonitor::dstart( ){
	/*
	QString a = "sss";
	QTextStream b( &a , IO_ReadWrite );
	logedit->insertText( &b  );
	*/
}
```

評 語

1.利用遺傳演算，以演化定義，模擬演算作資料搜尋上的應用，預期可以用於圍棋等智慧型應用。

2.程式與實作佳，較缺創意。