

通識計算機程式設計期中考

臺灣大學 鄭士康 4/22/2016

試題共 7 題，兩面印製 14 頁，滿分 100



本講義除另有註明外，採創用CC姓名標示-
非商業性-相同方式分享3.0臺灣版授權釋出

1. 撰寫一或數個C#敘述達成下列要求: (假設`using System;`敘述已經包含於程式中)
 - (a) 宣告`bool`變數`valid`，`double`變數`xd`，`float`變數`pr` (3%)
 - (b) 在螢幕顯示一行字，要求使用者輸入一個帶有小數點的實數 (3%)
 - (c) 自鍵盤讀入一個帶有小數點的實數，並將其值存入已宣告之`double`變數`xd` (3%)
 - (d) 將已宣告設值之`double`變數`xd`強制轉型為`float`數值，存入已宣告之`float`變數`pr` (3%)
 - (e) 將邏輯關係式`pr >= 0.0f && pr <= 1.0f`設定給已宣告之`bool`變數`valid` (3%)
2. 撰寫一或數個C#敘述達成下列要求: (假設`using System;`敘述已經包含於程式中)
 - (a) 將已宣告設值之`int`變數`n`，用遞增算子`++`加1 (3%)
 - (b) 令他處已宣告設值之`double`變數`chromosome`乘以2，取其積的整數部分，設定給他處已宣告之`int`變數`gene` (3%)
 - (c) 宣告`double`變數`result`，並設定其值為單一算式(expression) $1.0 - |x|$ ；亦即1.0 減去 `double`變數`x`的絕對值(`Math.Abs`)，假設`x`已於他處宣告設值 (3%)
 - (d) 利用三元運算子，使他處已宣告設值之`int`變數`bit`，原先數值大於0時，設定同一變數`bit`的新數值為0，反之則令其新數值為1 (3%)
 - (e) 宣告變數`c`為`char`型別，並令其值為水平位置控制字元(tabulation) (3%)
3. 撰寫一或數個C#敘述達成下列要求: (假設`using System;` 敘述已經包含於程式中)
 - (a) 先以一個敘述宣告一個亂數產生器物件`rand`，讓系統以時間及網路卡ID決定其種子數；再於另一敘述利用`rand.NextDouble()`產生一個0與1之間均勻分布的`double`亂數，設定給在此宣告為`double`的變數`chromosome`

(3%)

(b) 宣告一個**int**常數**POPULATION_SIZE**，設其值為**100** (3%)

(c) 寫一個**for**迴圈，整數迴圈控制變數**i**由**0**開始，每次加**1**，遞增至**POPULATION_SIZE-1**。迴圈中每個 iteration 自鍵盤讀入一個**double**數(螢幕上給使用者的提示可以省略)，將其值設定為陣列**fits**的第**i**個元素**fits[i]**。假定迴圈前已宣告設值**int**常數**POPULATION_SIZE**，以及長度**POPULATION_SIZE**的**double**數陣列**fits** (3%)

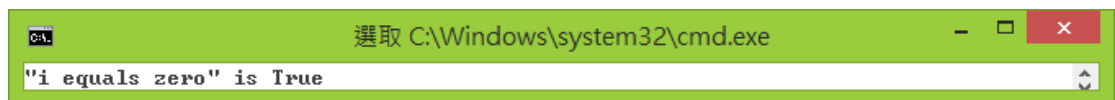
(d) 利用**Array.Sort()**將**double**數陣列**fits**由小而大排列。再利用**Array.Reverse()**倒排，而後取第一個元素，設定給他處已宣告之**double**變數**optimumValue** (3%)

(e) 寫一個**static void**函式**Initialize**，設其輸入參數為亂數產生器**rand**，輸出參數為**out double**參數**chromosome**；函式內利用**rand.NextDouble()**將**chromosome**設值為**0**與**1**之間的均勻分布亂數 (3%)

4. 找出以下程式片段之錯誤，並在盡量保持原先程式碼之前提下，予以更正。

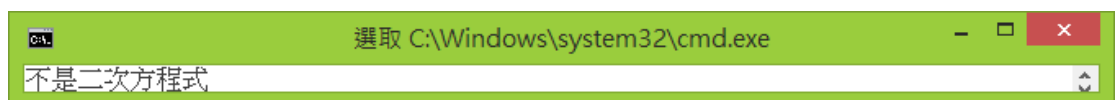
假設**using System**；敘述已經包含於程式中。

(a) (6%) (兩個語法錯誤) 執行時螢幕應顯示



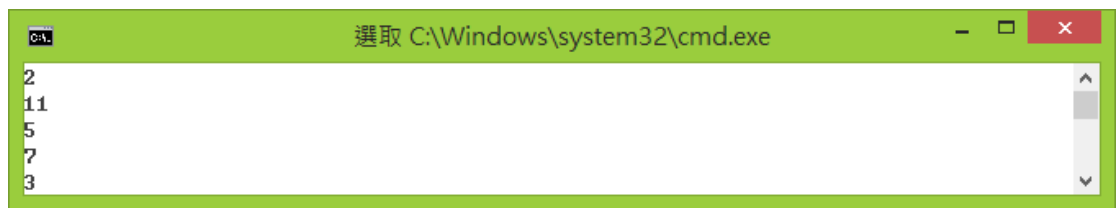
```
int i = 0;
bool iEqualsZero = (i = 0);
Console.WriteLine("i equals zero" is " + iEqualsZero);
```

(b) (3%) (一個語義錯誤) 執行時螢幕應顯示



```
int a = 0;
double delta = 4.0;
if (a != 0)
    if (delta > 0)
        Console.WriteLine("兩實根");
else
    Console.WriteLine("不是二次方程式");
```

(c) (3%) (一個語義錯誤) 執行時螢幕應顯示



```
C:\Windows\system32\cmd.exe
2
11
5
7
3
```

```
int[] p = { 2, 11, 5, 7, 3 };
int[] q = p;
Array.Sort(q);
for(int i = 0; i < p.Length; ++i)
{
    Console.WriteLine(p[i]);
}
```

(d) (3%) 數個同類型語法錯誤

```
static void Main(string[] args)
{
    double x = 1.0;
    double y = -1.0;
    double u;
    double v;
    Transform(x, y, ref u, ref v);
}

static void Transform(double x, double y,
    ref double u, ref double v)
{
    double a11 = 0.3;
    double a12 = 0.5;
    double a21 = 0.7;
    double a22 = 0.1;
    u = a11 * x + a12 * y;
    v = a21 * x + a22 * y;
}
```

5. 試寫出下列程式的螢幕輸出 (5 %)

```
using System;

namespace Problem5
{
    class Program
    {
        static void Main(string[] args)
        {
            const int N_GENES = 3;
            double x = 0.5;
            Console.WriteLine("x = " + x);
            int[] genes = Encode(x, N_GENES);
            Console.WriteLine("Encoded as:");
            Print(genes);

            Console.WriteLine();

            genes = new int[] {0, 1, 1};
            Console.WriteLine("genes are:");
            Print(genes);
            double y = Decode(genes);
            Console.WriteLine("Decoded as: " + y);
        }

        static int[] Encode(double x, int nGenes)
        {
            int[] results = new int[nGenes];
            double y;
            int m;
            for (int i = 0; i < nGenes; ++i)
            {
                y = 2.0 * x;
                m = (int)y;
                x = y - m;
                results[i] = m;
            }
        }
    }
}
```

```

        }
        return results;
    }

    static double Decode(int[] genes)
    {
        double result = 0.0;
        double weight = 0.5;
        for (int i = 0; i < genes.Length; ++i)
        {
            result += weight * genes[i];
            weight /= 2.0;
        }
        return result;
    }

    static void Print(int[] genes)
    {
        for(int i = 0; i < genes.Length; ++i)
        {
            Console.WriteLine("genes[{0}] = {1}",
                i, genes[i]);
        }
        Console.WriteLine();
    }
}

```

6. 試寫出下列程式的螢幕輸出 (10 %)

```

using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)

```

```

{
    double[] fits = { 28.0, 18.0, 14.0, 9.0, 26.0 };
    double[] chromosomes = { 1.0, 2.0, 3.0, 4.0, 5.0 };
    Console.WriteLine(
        "Before roulette wheel selection");
    Print(fits, chromosomes);
    RouletteWheelSelection(fits, chromosomes);
    Console.WriteLine(
        "After roulette wheel selection");
    Print(fits, chromosomes);
}

static void Print(double[] fits,
    double[] chromosomes)
{
    for(int i = 0; i < fits.Length; ++i)
    {
        Console.WriteLine(
            "fits[{0}] = {1}, chromosomes[{0}] = {2}",
            i, fits[i], chromosomes[i]);
    }
}

public static int Select(double r, double[] q)
{
    int i = 0;
    while (r > q[i] && i < q.Length) ++i;
    return i;
}

static void RouletteWheelSelection(
    double[] fits, double[] chromosomes)
{
    int n = fits.Length;
    double total = 0.0;
    int i;
    for (i = 0; i < n; ++i)
    {

```

```

        total += fits[i];
    }
    Array.Sort(fits, chromosomes);
    Array.Reverse(fits);
    Array.Reverse(chromosomes);
    double[] q = new double[n];
    double pi = fits[0] / total;
    q[0] = pi;
    for (i = 1; i < n; ++i)
    {
        pi = fits[i] / total;
        q[i] = q[i - 1] + pi;
    }
    int selected;
    double[] selectedFitness = new double[n];
    double[] selectedChromosomes = new double[n];
    double[] r = { 0.493, 0.111, 0.198, 0.416, 0.055 };
    for (i = 0; i < n; ++i)
    {
        selected = Select(r[i], q);
        selectedFitness[i] = fits[selected];
        selectedChromosomes[i] =
            chromosomes[selected];
    }
    Array.Copy(selectedFitness, fits, n);
    Array.Copy(selectedChromosomes, chromosomes, n);
}

}
}

```

7. 基因演算法(Genetic Algorithm, 簡稱 GA)是工程科學中，求最佳化(optimization, 或譯為「優化」)問題最佳解的常見方法。其概念源自 John H. Holland，模仿生物演化的歷程。每一代(generation)由許多「染色體」(chromosomes)構成族群(population)，每個染色體都可以決定一個「適配」值(fitness measure)。根據適配值大小，在族群中選擇(select)較能適應環境(亦即適配值較高)的染色體，增加其「交配」(mating)機會。同時，每個染色體也是由許多個 0 或 1 之「基因」(gene)

組成的字串，所以兩染色體交配時，二者部分基因「交換」(recombination 或稱 cross over)。而單一染色體的各基因，也都可能「突變」(mutation)，由 1 變為 0，或由 0 變為 1，稱為「位元反轉」(bit inversion)。交配及突變，產生子代(offspring)染色體。子代產生後，替換原先族群，再度進行基因重組與突變，反覆「演化」(evolution)。最後留存的染色體，其基因組合經過無數代的天擇演化，應該就會接近問題最佳解對應的基因字串。傳統的最佳化問題數學解法，如 gradient descent algorithm，得到的解答常常是區域最佳解(local solution)，而非全域最佳解(global solution)。相對而言，GA 所搜尋的區域可以遍及各處，比較可能找到全域解。關於 Holland 發明 GA 的傳奇故事以及 GA 的通俗解說，請參閱 M. Mitchell Waldrop 著，齊若蘭譯，《複雜：走在秩序與混沌邊緣》(Complexity – The emerging science at the edge of order and chaos)，台北市：天下文化，1994，頁 195~254。

基因演算法的流程如圖 1：起始族群包括多個染色體，每個染色體都是 0 或 1 的

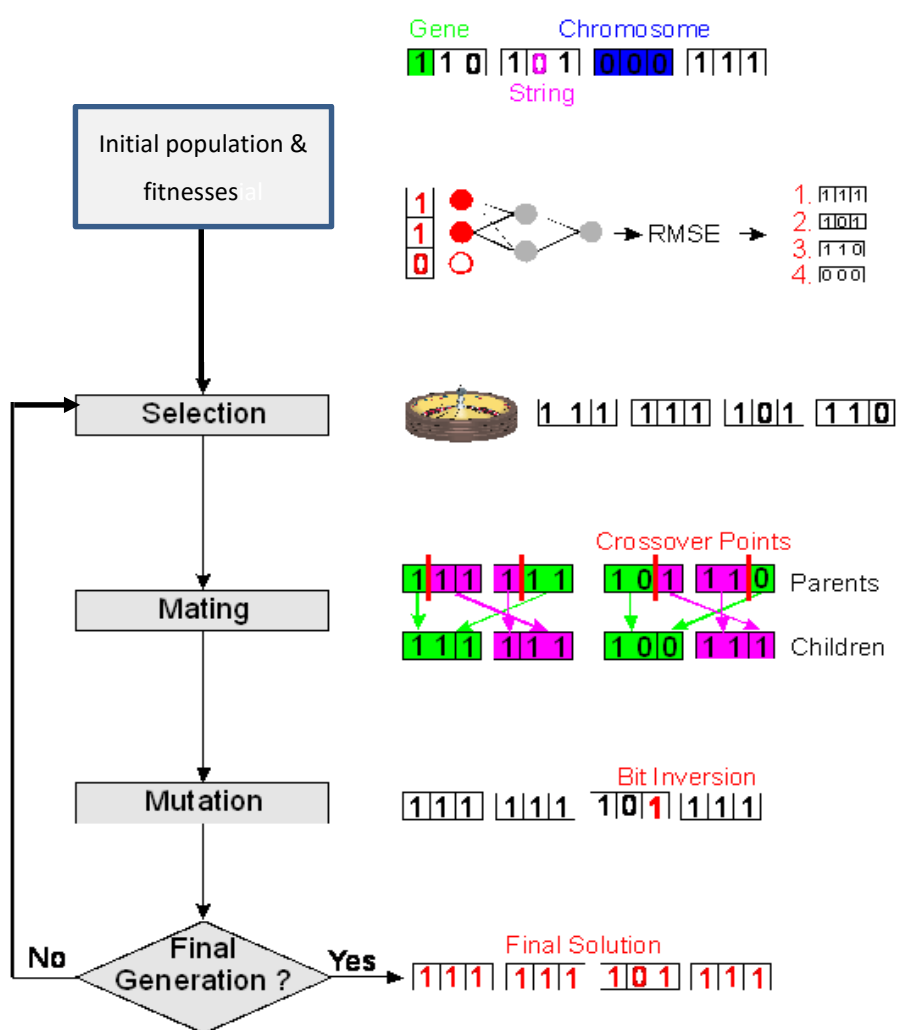


圖 1. 基因演算法流程 (修改自 http://www.frank-dieterle.de/phd/2_8_5.html)

基因組成的字串。每個染色體，可以依照要優化的目標函數(objective function)，算出一個對應的適配值。接著從族群中，依照適配值高低，分配各個染色體參加交配的機會。再其次便從挑選出來的染色體中，隨機配對，交換部分基因。交換基因有許多方法，圖 1 與圖 2 所示是最簡單的單點交叉(one-point crossover)方式：隨機決定一個位置 k ，染色體字串的前 k 個基因不動，第 $k+1$ 個以後的基因彼此交換，產生新一代。新世代的每個染色體，其中的每個基因，都依循設定好的突變機率，決定是否將基因位元反轉。至此檢驗表現最優的染色體適配值是否已經產生，或者演化的世代數已達上限。如果結束條件不成立，就回頭以新一代族群重新演化。

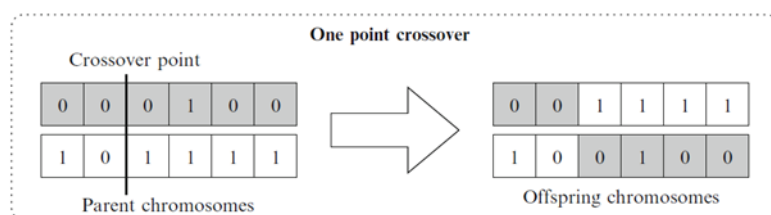


圖 2. 單點交叉 (取自 Kumara Sastry, David E. Goldberg, and Graham Kendall, "Genetic algorithms," Chapter 4, E.K. Burke and G. Kendall (eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, New York: Springer, 2014.)

本題請你依照如下說明及虛擬碼，撰寫一個基因演算法 C# 程式：

- 1) 主程式 **static void Main(string[] args)** 待完成。
- 2) 假設染色體是一個介於 0 和 1 之間的小數，可以利用後面附的函式 **Encode** 將小數轉成長度為 **N_GENES** 的位元陣列(實作為元素值 0 或 1 的整數陣列，方便計算)；也可以利用同樣附在本題題目後的函式 **Decode**，將長度為 **N_GENES** 的位元陣列，轉換為一個 0 與 1 間的小數。
- 3) 問題的目標函數定義為 $1 - |x - 0.5|$ ，其中 x 為染色體對應的小數。從這個適配值的定義，顯然我們的 GA 程式，應該跑出接近 0.5 的最佳解，對應的最佳適配值則要接近 1。函式 **Evaluate** 利用目標函數 **Fitness**，計算整個族群各染色體的適配值，二者的實作也附在後頭。
- 4) 使用俄羅斯輪盤(Roulette-Wheel)決定參與交配的親代染色體，其原理以例題說明如下：假設 5 個染色體對應的適配值 f_i 由大而小排列如表 1， f_0 代表最大的適配值。以適配值 f_i 除以適配值總和 $total = \sum_{i=0} f_i$ 決定適配值之染色體參與交配的機率 $p_i = f_i / total$ 。計算第 i 染色體的累積交配機率 $q_i = \sum_{j=0}^i p_j$ ，如表 1 的第五列所示。接著產生一個 0 與 1 之間，均勻分布的亂數 r 。如果

$q_{i-1} < r \leq q_i$, $q_{-1} = 0$ 就挑選 index 為 i 的染色體加入預定交配的親代。當例如，抽到的亂數是 $r = 0.458$ 時，就選擇 index 為 1 的染色體。挑選過的染色體可以重複被挑選，使其數量在預定交配的親代中佔有優勢。如此反覆 5 次，以維持親代染色體數目與原先族群染色體數目相同。顯然，適配值愈大的染色體，被挑選到的次數期望值愈高，它們的基因也更有可能傳遞到子代。後面列出的函式 **RouletteWheelSelection** 實作了上述的演算法。

表 1. 俄羅斯輪盤方法挑選染色體的範例

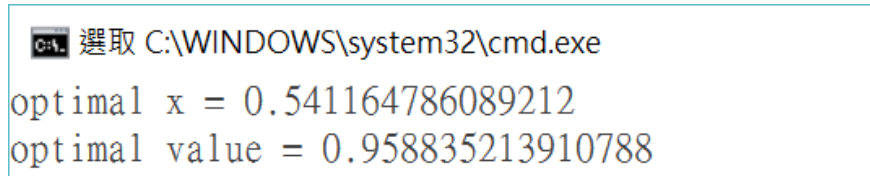
Index, i	0	1	2	3	4
Chromosome, x_i	1.0	5.0	2.0	3.0	4.0
Fitness, f_i	28	26	18	14	9
Probability, p_i	$28/95 \approx 0.295$	0.274	0.189	0.147	0.095
Cumulative probability, q_i	0.295	0.569	0.758	0.905	1.0

$$total = \sum_{i=0}^4 f_i = 95, \quad p_i = \frac{f_i}{total}, \quad \sum_{i=0} p_i = 1, \quad q_i = \sum_{j=0}^i p_j$$

- 5) 使用單點交叉交換基因 (待完成)。
- 6) 演化到親代染色體都很相似的時候，為避免適配值不再改變(可能陷入局部最佳解)，可以引入突變，增加變異機會，使未曾考慮過的染色體也能加入演化過程。假設突變機率為 p_m ，則某一染色體的突變模擬方式如下：對所有基因，以亂數產生器產生一個介於 0 與 1 之間的隨機數。如果此一隨機數小於 p_m ，該基因就要進行 bit inversion，否則基因值不變。此一步驟待完成。
- 7) 為簡化問題，令演化世代數固定為 **N_GENERATIONS = 1000**，族群的大小為 **POPULATION_SIZE = 100**，**N_GENES = 32**，突變機率 **0.15**。整個問題的虛擬碼如下：

1. 產生第一代的 chromosome 族群，並計算每個 chromosome 的適配值
2. Repeat N_GENERATIONS 個世代
 - {
 - 2.1 以俄羅斯輪盤選出參加交配的親代 parent chromosomes
 - 2.2 Repeat POPULATION_SIZE/2 次
 - {
 - 2.2.1 由 parent chromosomes 隨機挑選兩個 chromosome
 - 2.2.2 以單點交叉交換基因，產生兩個新的 offspring chromosome
 - }
 - 2.3 依照突變機率模擬每個 offspring chromosome 內的基因突變
 - 2.4 以 offspring chromosomes 取代原先的 chromosomes 族群
 - 2.5 計算新的 chromosomes 族群中每個 chromosome 的適配值
 - }
3. 以族群各 chromosome 的適配值為 key，對應染色體為 item，依照 key 由大而小排序
4. 輸出最大適配值(optimum value)及對應染色體(optimum solution)

程式專案執行畫面如圖 3。



```

C:\WINDOWS\system32\cmd.exe
optimal x = 0.541164786089212
optimal value = 0.958835213910788
  
```

圖 3. 第 7 題的專案執行螢幕畫面範例

本題滿分 25 分，全部待寫程式集中寫成一個大 **Main** 函式，不區分函式者，最高得 23 分；善用函式，乃至尚未教到的物件導向程式設計(object-oriented programming)者，最高得 25 分。(25%)

以下的函式可供參考利用(呼叫此處提供的函式，例如 **RouletteWheelSelection**，可不必在答案卷又抄寫一次)，如果覺得不合用，自己重新撰寫也可以)：

1). 函式 **Encode**

```

static int[] Encode(double x, int nGenes)
{
  
```

```

    int[] results = new int[nGenes];
    double y;
    int m;
    for (int i = 0; i < nGenes; ++i)
    {
        y = 2.0 * x;
        m = (int)y;
        x = y - m;
        results[i] = m;
    }
    return results;
}。

```

2). 函式 Decode

```

static double Decode(int[] genes)
{
    double result = 0.0;
    double weight = 0.5;
    for (int i = 0; i < genes.Length; ++i)
    {
        result += weight * genes[i];
        weight /= 2.0;
    }
    return result;
}

```

3). 函式 Evaluate 及 Fitness

```

static double[] Evaluate(double[] chromosomes)
{
    int populationSize = chromosomes.Length;
    double[] fits = new double[populationSize];
    for (int i = 0; i < populationSize; ++i)
    {
        fits[i] = Fitness(chromosomes[i]);
    }
    return fits;
}

```

```

    }

    static double Fitness(double x)
    {
        double fit = 1.0 - Math.Abs(x - 0.5);
        return fit;
    }

```

4). 函式 RouletteWheelSelection 及 Select

```

static void RouletteWheelSelection(
    double[] fits, double[] chromosomes, Random rand)
{
    int n = fits.Length;
    double total = 0.0;
    int i;
    for (i = 0; i < n; ++i)
    {
        total += fits[i];
    }
    Array.Sort(fits, chromosomes);
    Array.Reverse(fits);
    Array.Reverse(chromosomes);
    double[] q = new double[n];
    double pi = fits[0] / total;
    q[0] = pi;
    for (i = 1; i < n; ++i)
    {
        pi = fits[i] / total;
        q[i] = q[i - 1] + pi;
    }
    int selected;
    double[] selectedFitness = new double[n];
    double[] selectedChromosomes = new double[n];
    double r = 0.0;
    for (i = 0; i < n; ++i)
    {
        r = rand.NextDouble();

```

```

        selected = Select(r, q);
        selectedFitness[i] = fits[selected];
        selectedChromosomes[i] =
            chromosomes[selected];
    }
    Array.Copy(selectedFitness, fits, n);
    Array.Copy(selectedChromosomes, chromosomes, n);
}

static int Select(double r, double[] q)
{
    int i = 0;
    while (r > q[i] && i < q.Length) ++i;
    return i;
}

```