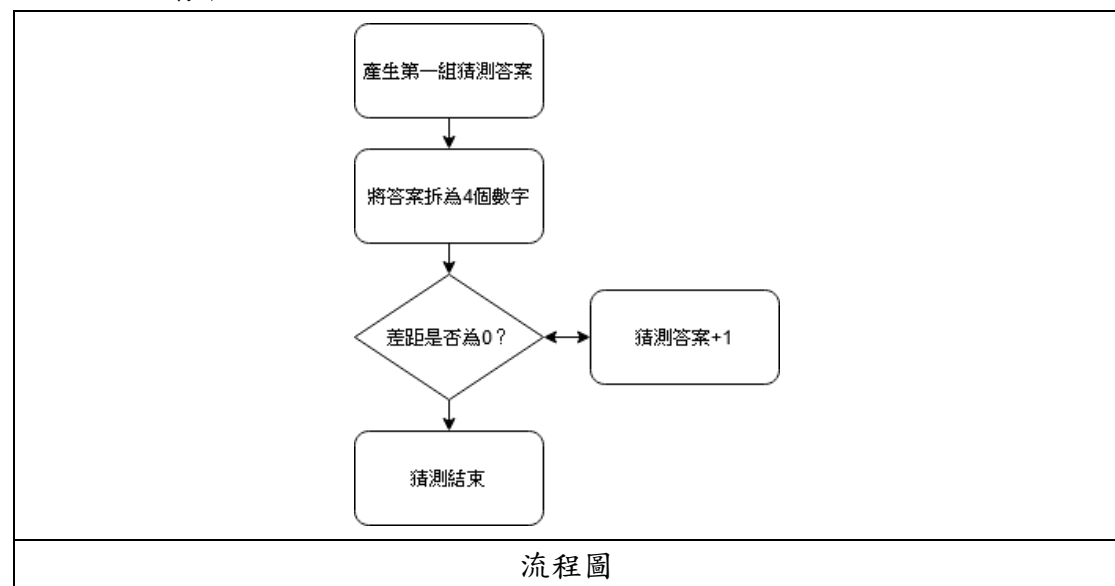


# 一、4 位數窮舉法：



## 1. 部分程式碼註解

```
scanf("%d",&a);
for(int b=3 ; b>=0;b--)
{
    answer[b]=a%10;
    a/=10;
}
```

輸入正解後，將其拆解成 4 個位數存入陣列

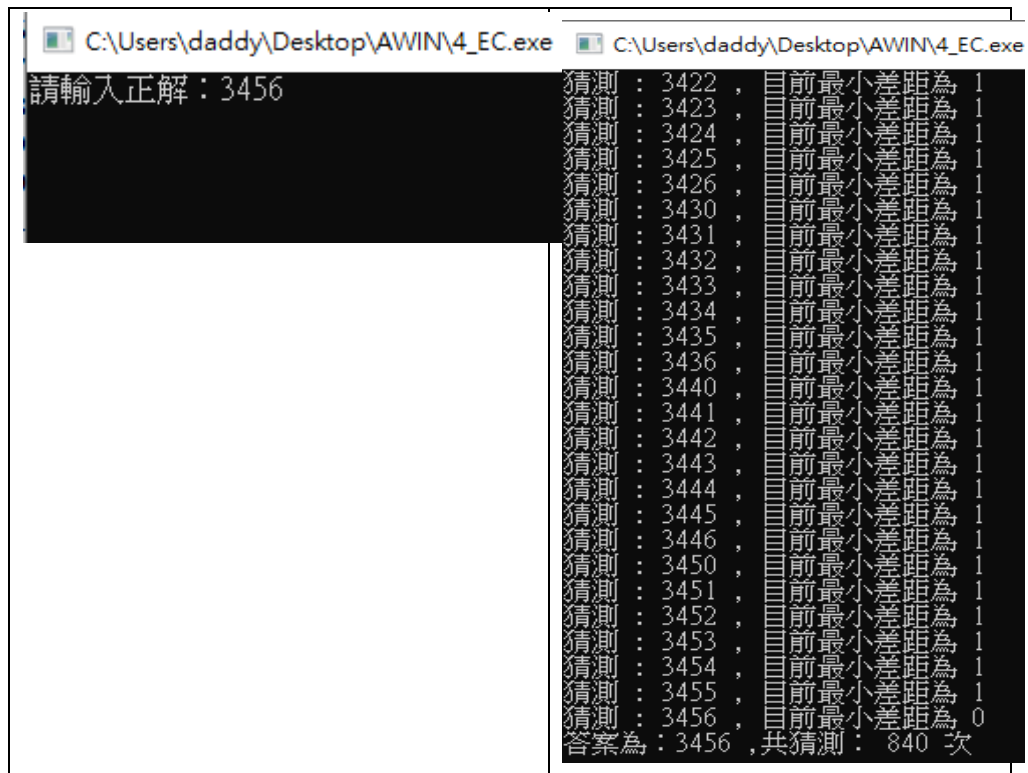
```

for(int i=0;i<10;i++)
{
    for(int j=0;j<10;j++)
    {
        for(int k=0;k<10;k++)
        {
            for(int l=0;l<10;l++)
            {
                value=abs(answer[0]-i)+abs(answer[1]-j)+abs(answer[2]-k)+abs(answer[3]-l);
                if(value<min)
                {
                    min=value;
                }
                times++;
                printf("猜測： %d%d%d%d , 目前最小差距為 %d \n",i,j,k,l,min);
                if(l==answer[3])
                {
                    ans=l;
                    break;
                }
            }
            if(k==answer[2])
            {
                ans=k*10+ans;
                break;
            }
        }
        if(j==answer[1])
        {
            ans=j*100+ans;
            break;
        }
    }
    if(i==answer[0])
    {
        ans=i*1000+ans;
        break;
    }
}
printf("答案為： %d , 共猜測： %d 次",ans,times);
  
```

從最第一位數開始猜測，若相符則往下一位數繼續猜測，最後算出答案

## 2. 執行方法

開啟程式後輸入正解，即會自動開始猜測。

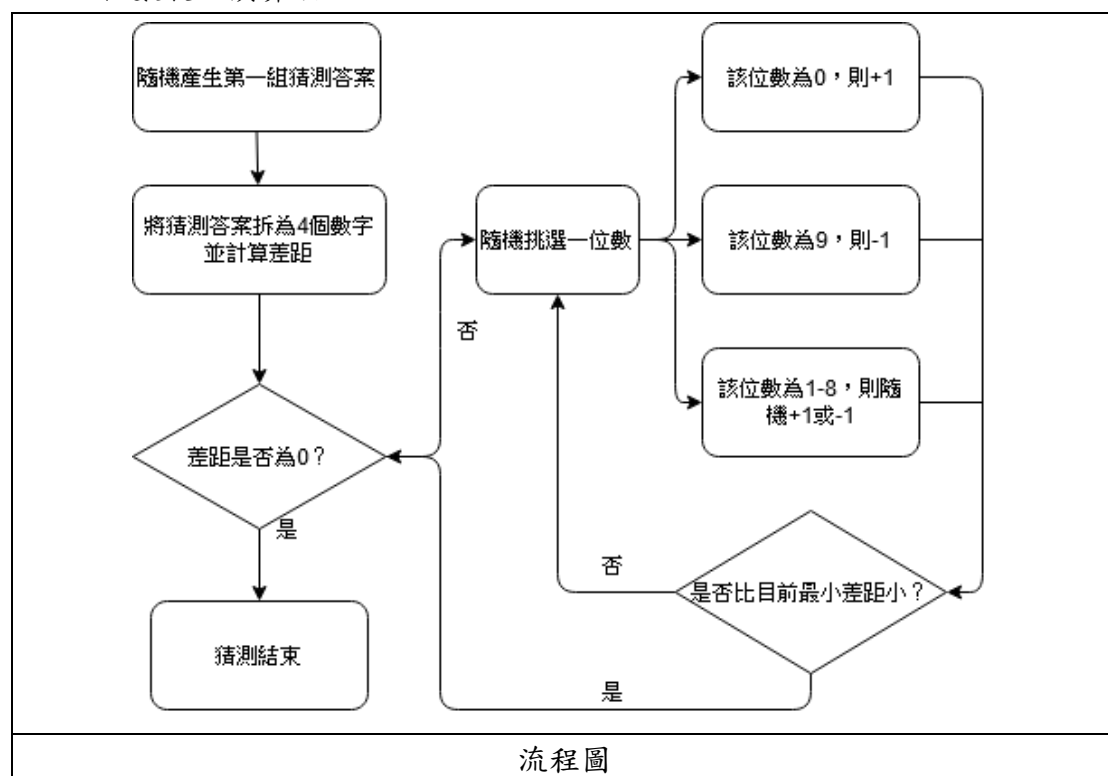


```
C:\Users\daddy\Desktop\AWIN\4_EC.exe C:\Users\daddy\Desktop\AWIN\4_EC.exe
請輸入正解：3456
猜測：3422，目前最小差距為1
猜測：3423，目前最小差距為1
猜測：3424，目前最小差距為1
猜測：3425，目前最小差距為1
猜測：3426，目前最小差距為1
猜測：3430，目前最小差距為1
猜測：3431，目前最小差距為1
猜測：3432，目前最小差距為1
猜測：3433，目前最小差距為1
猜測：3434，目前最小差距為1
猜測：3435，目前最小差距為1
猜測：3436，目前最小差距為1
猜測：3440，目前最小差距為1
猜測：3441，目前最小差距為1
猜測：3442，目前最小差距為1
猜測：3443，目前最小差距為1
猜測：3444，目前最小差距為1
猜測：3445，目前最小差距為1
猜測：3446，目前最小差距為1
猜測：3450，目前最小差距為1
猜測：3451，目前最小差距為1
猜測：3452，目前最小差距為1
猜測：3453，目前最小差距為1
猜測：3454，目前最小差距為1
猜測：3455，目前最小差距為1
猜測：3456，目前最小差距為0
答案為：3456，共猜測：840次
```

## 3. 遇到的困難

因為完全沒有寫過演算法的題目，所以剛開始遇到最大的困難就是看完題目後卻不知道題目要求的窮舉法是什麼，是後來在 Google 後找到了相關資料才知道原來就是暴力破解法，所以就很直覺地寫完了。

## 二、4 位數爬山演算法



### 1. 部分程式碼註解

<pre> printf("請輸入4位數正解："); scanf("%d",&amp;a);  for(int b=3 ; b&gt;=0;b--) {     answer[b]=a%10;     a/=10; } </pre>	<pre> srand( time(NULL) );  for(int i=0;i&lt;4;i++) {     guess[i] = rand() % (9 - 0 + 1) + 0; }  for(int j=0;j&lt;4;j++) {     bestcost=abs(guess[j]-answer[j]);     best_cost=bestcost+best_cost; } </pre>
輸入正解後拆解成 4 位數	隨機產生第一組猜測解，並將差距值當為最佳差距

<pre> guess_random=rand() % (3 - 0 + 1) + 0; count_random=rand() % (1 - 0 + 1) + 0; </pre>
隨機決定要改變的位數，以及隨機決定要+1 或-1

```

if(guess[guess_random]==0)
{
    times++;
    x=guess[guess_random];
    guess[guess_random]+=1;
    upcost=abs(guess[0]-answer[0])+abs(guess[1]-answer[1])+abs(guess[2]-answer[2])+abs(guess[3]-answer[3]);
    printf("差距為: %d ,第 %d 位數 +1 後差距: %d \n",best_cost,guess_random+1,upcost);
    if(upcost < best_cost)
    {
        best_cost = upcost;
    }
    else
    {
        guess[guess_random]=x;
    }
}

```

若選擇的位數為 0，則只能將其+1，接著以 x 紀錄原本數值，再計算差距值，最後將較小的差距值留下。

```

else if(guess[guess_random]==9)
{
    times++;
    y=guess[guess_random];
    guess[guess_random]=guess[guess_random]-1;
    downcost=abs(guess[0]-answer[0])+abs(guess[1]-answer[1])+abs(guess[2]-answer[2])+abs(guess[3]-answer[3]);
    printf("差距為: %d ,第 %d 位數 -1 後差距: %d \n",best_cost,guess_random+1,downcost);
    if(downcost < best_cost)
    {
        best_cost = downcost;
    }
    else
    {
        guess[guess_random]=y;
    }
}

```

若選擇的位數為 9，則只能將其-1，接著以 y 紀錄原本數值，再計算差距值，最後將較小的差距值留下。

```

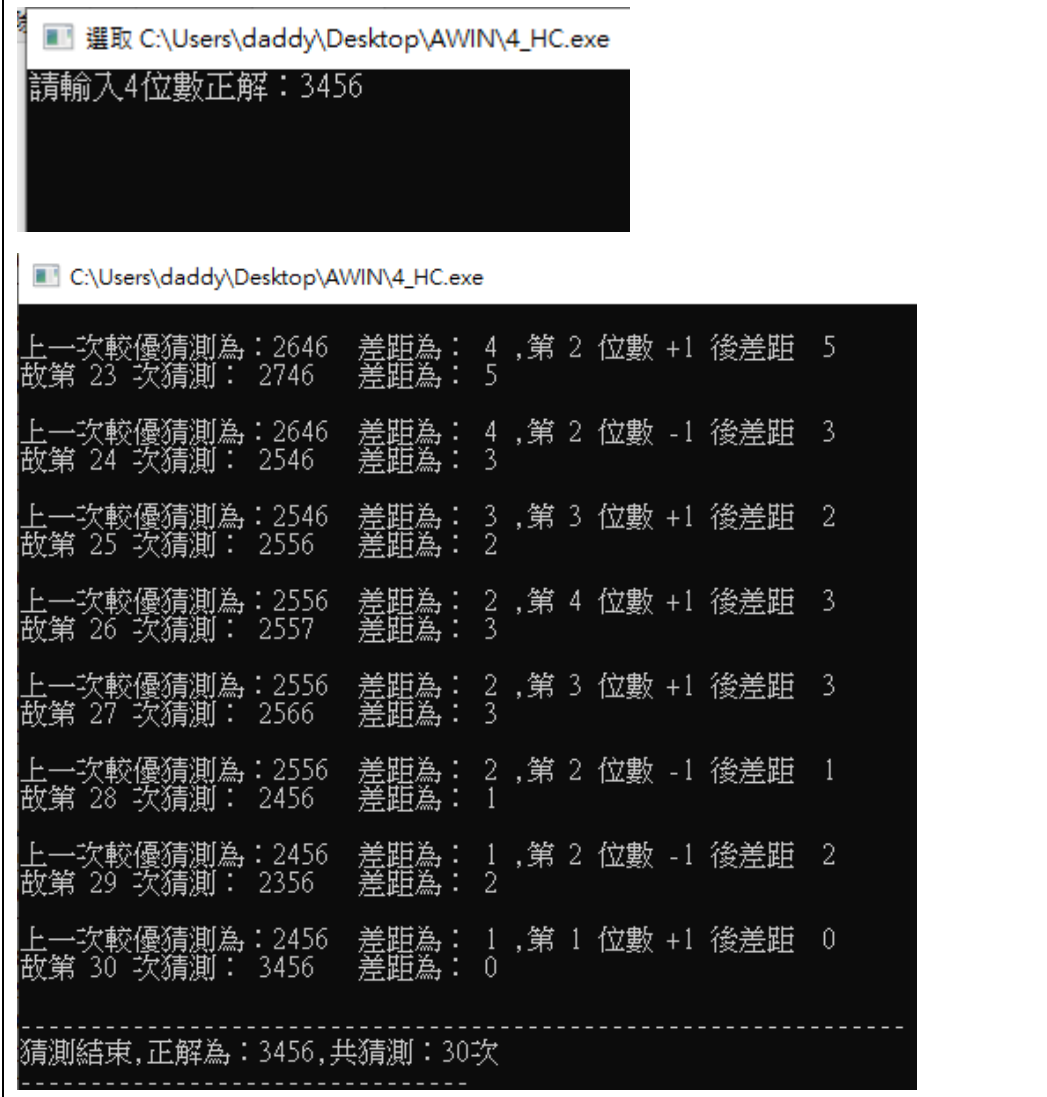
else
{
    if(count_random==0)
    {
        times++;
        x=guess[guess_random];
        guess[guess_random]=guess[guess_random]+1;
        upcost=abs(guess[0]-answer[0])+abs(guess[1]-answer[1])+abs(guess[2]-answer[2])+abs(guess[3]-answer[3]);
        printf("差距為: %d ,第 %d 位數 +1 後差距: %d \n",best_cost,guess_random+1,upcost);
        if(upcost < best_cost)
        {
            best_cost = upcost;
        }
        else
        {
            guess[guess_random]=x;
        }
    }
    else if(count_random ==1)
    {
        times++;
        y=guess[guess_random];
        guess[guess_random]=guess[guess_random]-1;
        downcost=abs(guess[0]-answer[0])+abs(guess[1]-answer[1])+abs(guess[2]-answer[2])+abs(guess[3]-answer[3]);
        printf("差距為: %d ,第 %d 位數 -1 後差距: %d \n",best_cost,guess_random+1,downcost);
        if(downcost < best_cost)
        {
            best_cost = downcost;
        }
        else
        {
            guess[guess_random]=y;
        }
    }
}

```

若為 1-8，則隨機將數值+1 或者-1，接著以變數紀錄原本數值，再計算差距值，最後將較小的差距值留下。

## 2. 執行方法

開啟程式後輸入正解，即會自動開始猜測。



```
選取 C:\Users\daddy\Desktop\AWIN\4_HC.exe
請輸入4位數正解：3456

C:\Users\daddy\Desktop\AWIN\4_HC.exe
上一次較優猜測為：2646 差距為： 4 ,第 2 位數 +1 後差距 5
故第 23 次猜測： 2746 差距為： 5
上一次較優猜測為：2646 差距為： 4 ,第 2 位數 -1 後差距 3
故第 24 次猜測： 2546 差距為： 3
上一次較優猜測為：2546 差距為： 3 ,第 3 位數 +1 後差距 2
故第 25 次猜測： 2556 差距為： 2
上一次較優猜測為：2556 差距為： 2 ,第 4 位數 +1 後差距 3
故第 26 次猜測： 2557 差距為： 3
上一次較優猜測為：2556 差距為： 2 ,第 3 位數 +1 後差距 3
故第 27 次猜測： 2566 差距為： 3
上一次較優猜測為：2556 差距為： 2 ,第 2 位數 -1 後差距 1
故第 28 次猜測： 2456 差距為： 1
上一次較優猜測為：2456 差距為： 1 ,第 2 位數 -1 後差距 2
故第 29 次猜測： 2356 差距為： 2
上一次較優猜測為：2456 差距為： 1 ,第 1 位數 +1 後差距 0
故第 30 次猜測： 3456 差距為： 0
-----
猜測結束,正解為:3456,共猜測:30次
-----
```

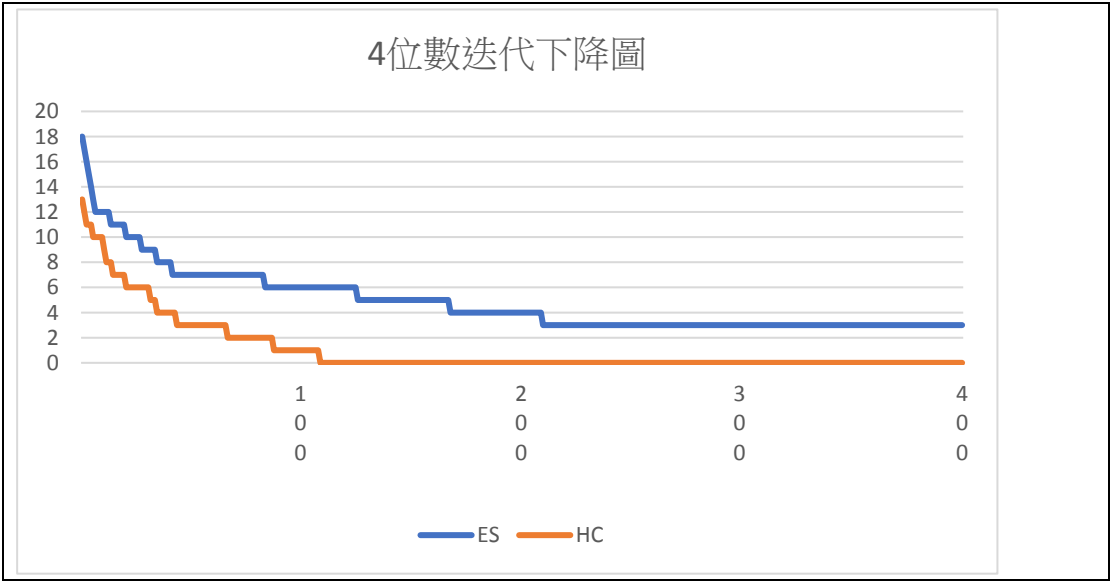
## 3. 遇到的困難

最一開始看到題目後沒有查資料就先靠著自己的感覺寫出的第一個解答，但當時因為不熟悉，所以不確定到底比較像哪種方法，查了很多資料、發文詢問，問了朋友、老師們得到的答案有人說像窮舉，有人說像爬山，讓我困擾了一陣子，後來是和朋友一起重新全部討論過後才覺得比較像是爬山，最後自己修改成現在的樣子。

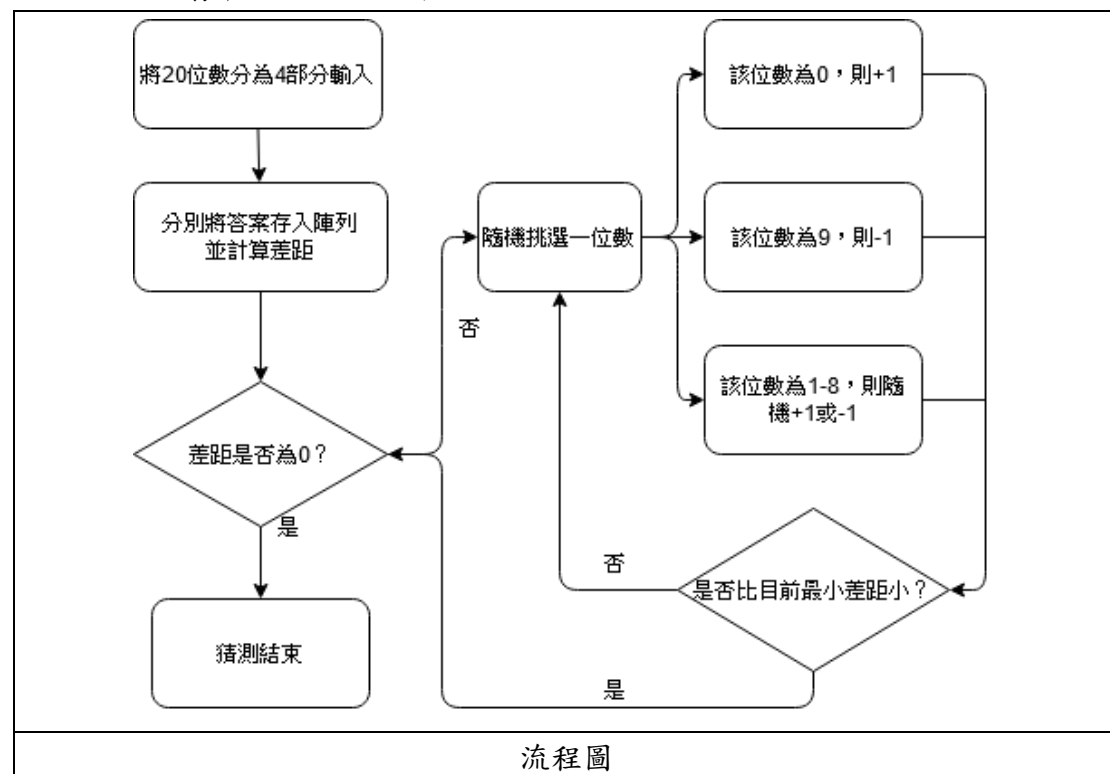
## 4. 參考資料

<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/502300/>  
<https://www.itread01.com/content/1544730876.html>

三、4 位數窮舉、爬山的迭代下降圖



#### 四、20 位數窮舉法、爬山演算法



#### 1. 部分程式碼註解

```

printf("請輸入20位數正解第 1 - %d 位數:", num3);
scanf("%d", &a3);

printf("請輸入20位數正解第 %d - %d 位數:", num3+1, num2);
scanf("%d", &a2);

printf("請輸入20位數正解第 %d - %d 位數:", num2+1, num1);
scanf("%d", &a1);

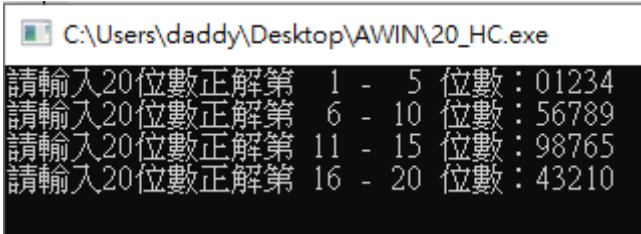
printf("請輸入20位數正解第 %d - %d 位數:", num1+1, num);
scanf("%d", &a);

for(int b=num-1 ; b>=0; b--){
    answer[b]=a%10;
    a/=10;
}
for(int b=num-6 ; b>=0; b--){
    answer[b]=a1%10;
    a1/=10;
}
for(int b=num-11 ; b>=0; b--){
    answer[b]=a2%10;
    a2/=10;
}
for(int b=num-16 ; b>=0; b--){
    answer[b]=a3%10;
    a3/=10;
}
  
```

因整數及長整數無法存取 20 位數，因此將正解分次輸入再存入陣列

2. 迭代下降圖放於下一題一同比較。

### 3. 執行方法



假設正確答案為 01234 56789 98765 43210 則分批輸入



輸入後 便會自動開始猜測

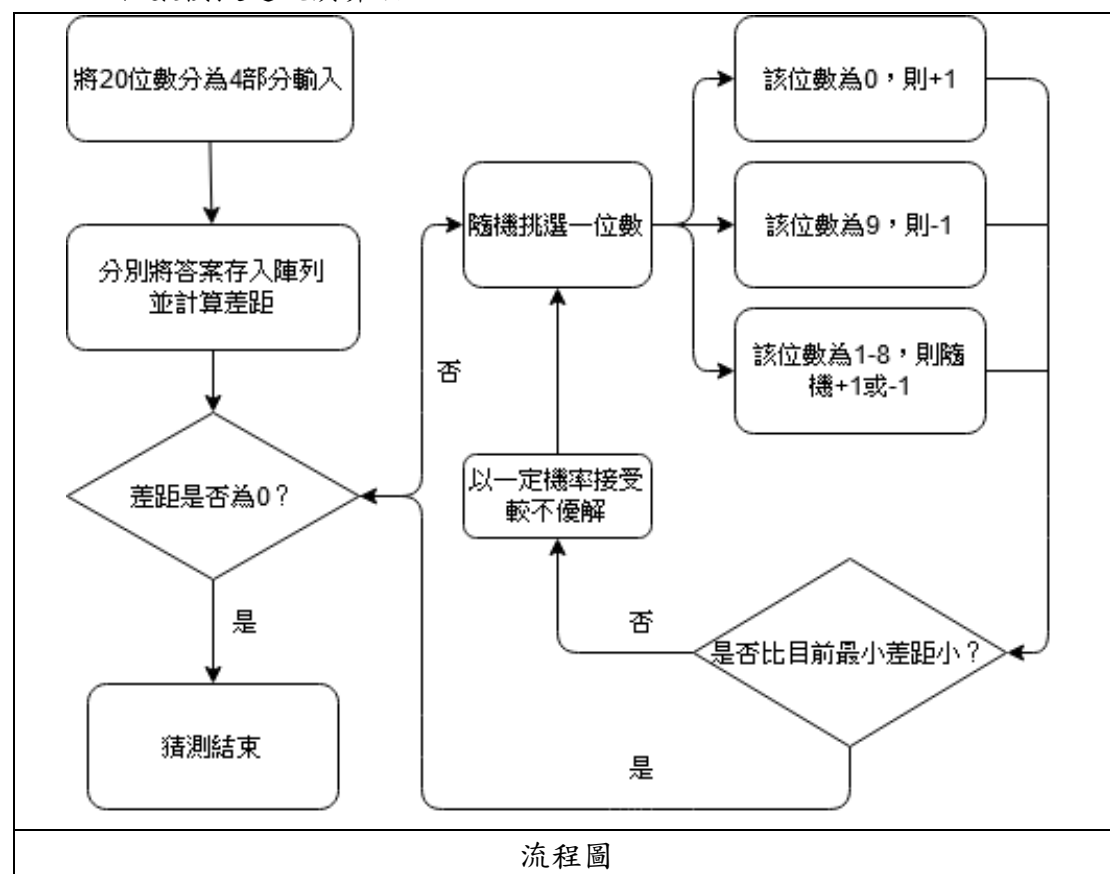
### 4. 遇到的困難

剛開始想說只要將正解位數擴大成 20 就可以解決了，但實際測試後不知道為什麼數值都會爆掉，後來嘗試縮小為 4 位數又變回正常，反覆測試後發現當位數 $\leq 10$  位數時都能順利執行，到第 11 位數時會跑出負數，才突然想到整數有可存取的範圍上限，接著原本想使用長整數來記錄，才發現長整數也無法完整存取 20 位數，最後才想到將正解分次輸入來記錄。

而 20 位數的窮舉法，當時想說要把時間把握在研究下面幾題的演算法上，所以就沒有特別簡化程式碼，而是直接將 4 位數程式碼擴大。



## 五、20 位數模擬退火演算法



### 1. 部分程式碼註解

```

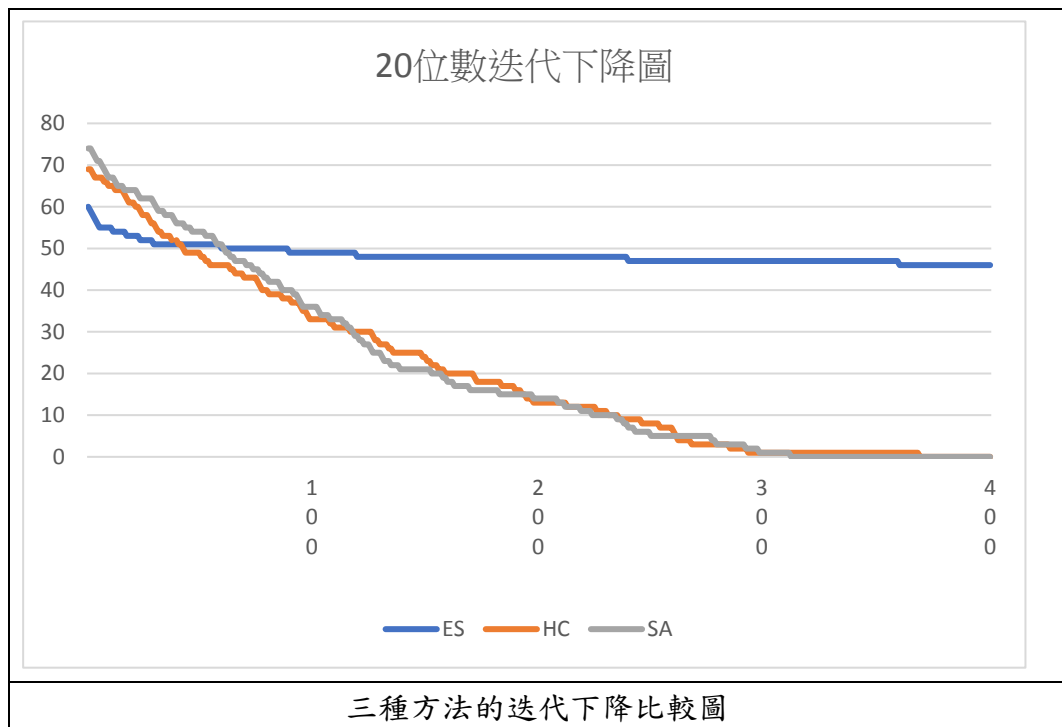
else
{
    double rd = rand() / (RAND_MAX + 1.0);
    if(exp(- cost / startT) > rd )
    {
        guess[guess_random]=x;

        P_L++;
    }
    if(P_L == limit)
        break;

    startT *= delta;
}
  
```

將爬山演算法找到的非最佳解以一定機率保存下來

## 2. 迭代下降圖



在位數小的時候各種演算法的差別其實感覺不太出來，但放大成 20 位數時窮舉法很明顯的次數提升很多，但爬山演算法及退火演算法在我上網查了資料後得知，兩者最大的差別是在尋找區域最佳解和全域最佳解時才會有較明顯差距，所以在這題好像不會太大的差異，但以速度來說，模擬退火還是小小的略勝一籌。

## 3. 執行方法

```
C:\Users\daddy\Desktop\AWIN\20_SA.exe
請輸入20位數正解第 1 - 5 位數：12345
請輸入20位數正解第 6 - 10 位數：67890
請輸入20位數正解第 11 - 15 位數：09876
請輸入20位數正解第 16 - 20 位數：54321
```

假設正確答案為 12345 67890 09876 54321 則分批輸入

```
C:\Users\daddy\Desktop\AWIN\20_SA.exe

上一次猜測為：12345677900987654321 差距為：1 ,第 15 位數 -1 後差距 2
故第 355 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 4 位數 +1 後差距 2
故第 356 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 7 位數 +1 後差距 2
故第 357 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 10 位數 +1 後差距：2
故第 358 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 15 位數 -1 後差距 2
故第 359 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 7 位數 -1 後差距 2
故第 360 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 17 位數 -1 後差距 2
故第 361 次猜測：12345677900987654321 差距為：1

上一次猜測為：12345677900987654321 差距為：1 ,第 8 位數 +1 後差距 0
故第 362 次猜測：12345678900987654321 差距為：0

-----
猜測結束,正解為：12345678900987654321,共猜了：362次
```

輸入後便會開始猜測

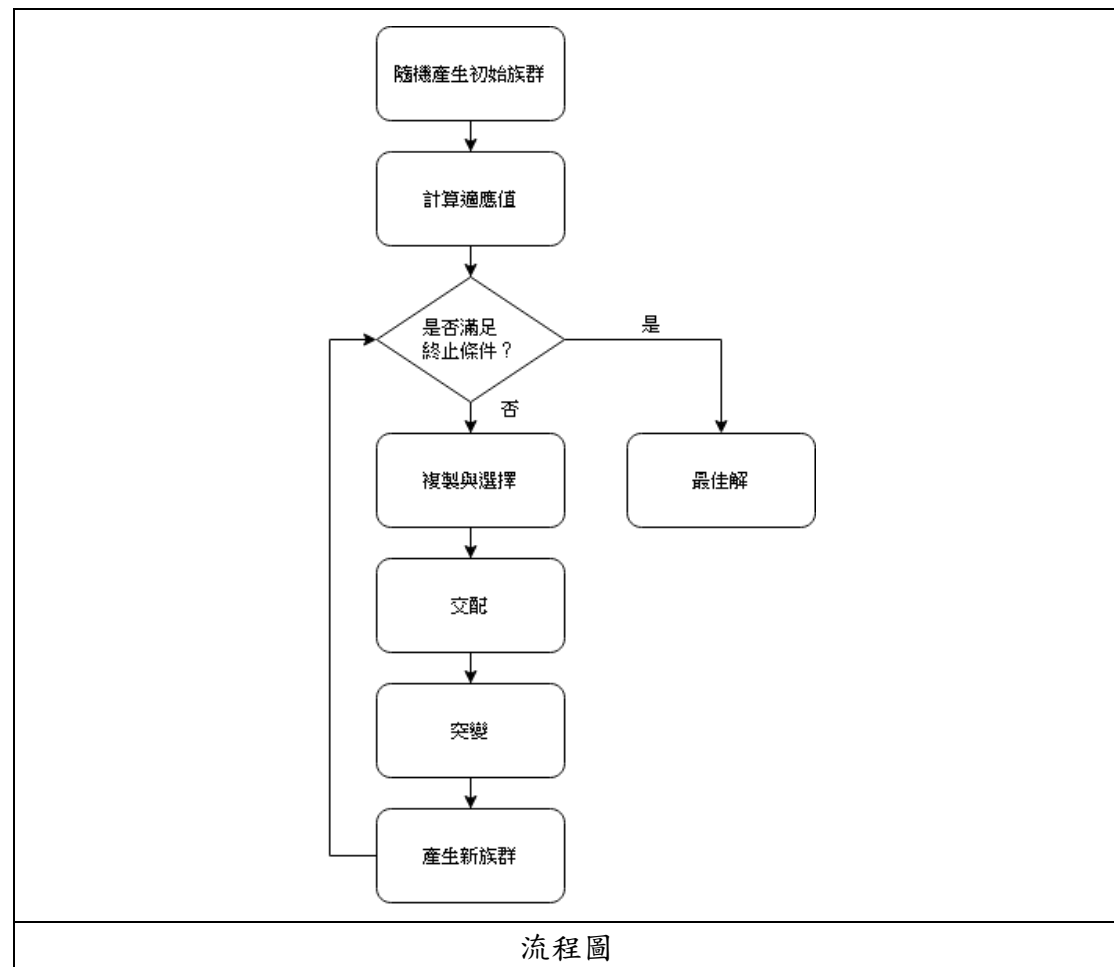
#### 4. 遇到的困難

上網查到了很多寫得很複雜的範例，花了不少時間研究後雖然大致知道理論及運行方式，但卻不太確定如何實作，最後是盡量的照著自己知道的意思嘗試去修改爬山演算法後寫了出來，雖然能夠成功執行以及計算答案，但依舊不確定是否真的算是模擬退火演算法。

#### 5. 參考資料

<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/502300/>  
<https://www.itread01.com/content/1544730876.html>  
<https://www.itread01.com/content/1547235128.html>

## 六、20 位數基因演算法



### 1. 部分程式碼註解

```
for(loop=0; loop<5; loop++)
{
    initialize();           // 初始化
    for(i=0; i<ITERA_CNT; i++){
        reproduction();     // 選擇(分配式)
        // reproduction_rnd(); // 選擇(隨機式), 收斂速度慢
        crossover();        // 交配
        mutation();         // 突變
    }
    a[loop]=best_gene.dec_value;
    times();
}
```

將答案拆為五部分猜測，每猜完一部分便初始化重新執行

```

void cal_xvalue(parent_t *x)
{
    int i, dec=0;
    for(i=0; i<GENETIC_LENGTH; i++){
        if(x->genes[i] ==1) dec = dec + (0x01 << i);
        if(dec>9999)
            dec=0;
    }
    x->dec_value = (double)dec;
}

```

將二進位染色體轉至為可用的十進位，因將答案拆解為 4 位數，故大於 4 位數字(即 9999 以上)皆視為 0

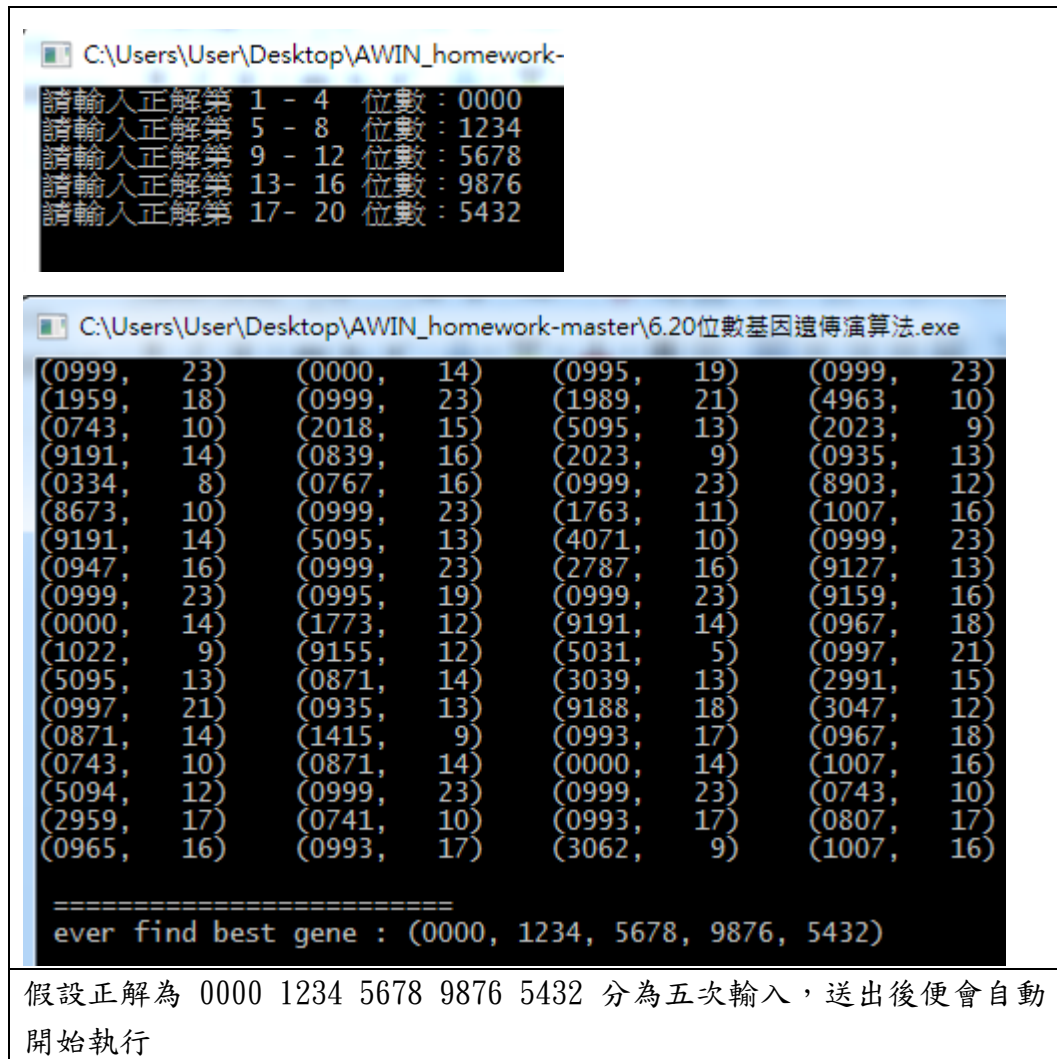
```

void cal_fitness(parent_t *x)
{
    int i = x->dec_value;
    int g[4]={0};
    g[0]=i/1000;
    g[1]=i/100%10;
    g[2]=i/10%10;
    g[3]=i%10;
    if(times==1)
        x->fitness =abs(g[3]-ans[3])+abs(g[2]-ans[2])+abs(g[1]-ans[1])+abs(g[0]-ans[0]);
    else if(times==2)
        x->fitness =abs(g[3]-ans1[3])+abs(g[2]-ans1[2])+abs(g[1]-ans1[1])+abs(g[0]-ans1[0]);
    else if(times==3)
        x->fitness =abs(g[3]-ans2[3])+abs(g[2]-ans2[2])+abs(g[1]-ans2[1])+abs(g[0]-ans2[0]);
    else if(times==4)
        x->fitness =abs(g[3]-ans3[3])+abs(g[2]-ans3[2])+abs(g[1]-ans3[1])+abs(g[0]-ans3[0]);
    else if(times==5)
        x->fitness =abs(g[3]-ans4[3])+abs(g[2]-ans4[2])+abs(g[1]-ans4[1])+abs(g[0]-ans4[0]);
}

```

適應函式，將染色體 x 拆解為 4 個數字並計算差距

## 2. 執行方法



## 3. 遇到的困難

一開始看了很多有關基因演算法的範例，幾乎不太了解到底該如何實作，但隨著每天下班後利用一點時間反覆的不斷從頭研究和嘗試後，最後才得知大致上應該如何修改來得到符合我所需要的結果。

## 4. 參考資料

[https://edisonshih.pixnet.net/blog/post/30373432?fbclid=IwAR2f-LfOLLz218PN0bykFgIF5xdUK05eIFjus6VGb0krm9DDnTfvWs\\_G77U](https://edisonshih.pixnet.net/blog/post/30373432?fbclid=IwAR2f-LfOLLz218PN0bykFgIF5xdUK05eIFjus6VGb0krm9DDnTfvWs_G77U)