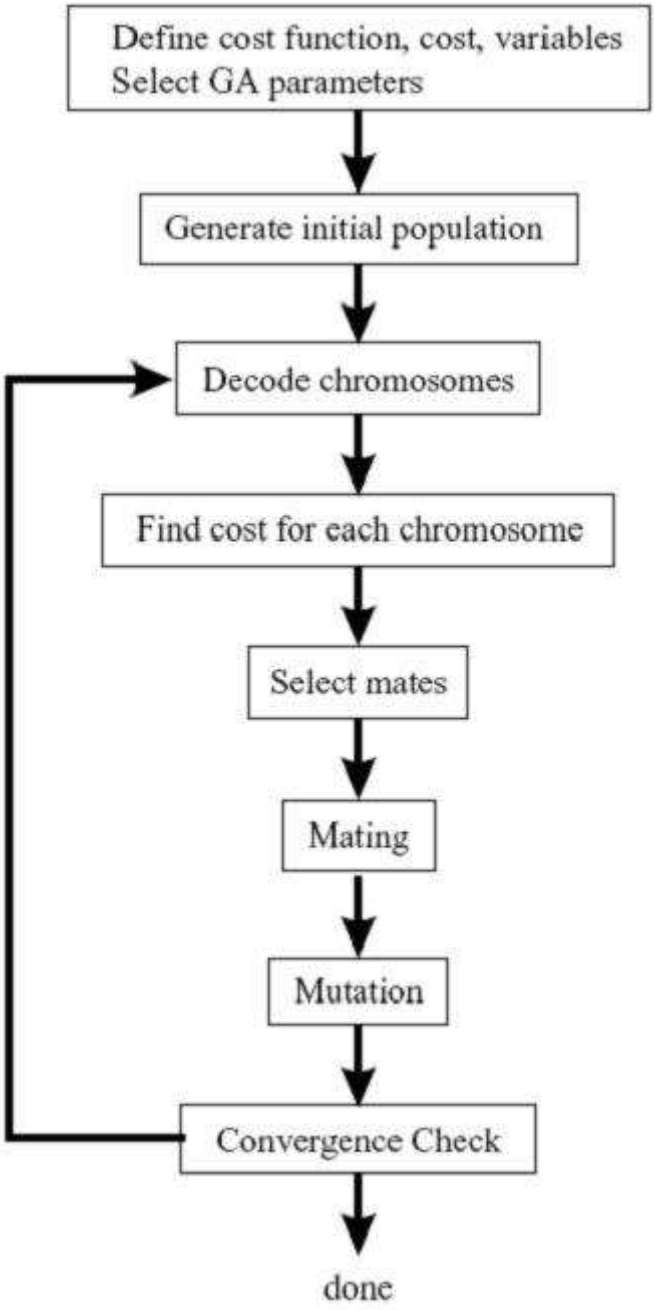
นาย ประวิทย์ หวังชูสันติ 5370532721

N-Queen Genetic Algorithm



Define :

#define NumberSolution	คือจำนวนคำตอบที่ต้องการหา
#define Q	คือ จำนวน Queen และขนาดของตาราง
#define Population	คือ จำนวนChromosome ที่จะสร้าง
#define MutationRate	คืออัตราการกลายพันธุ์
#define D (Q+Q-1)	คือ จำนวนเส้นเฉียง

Variable :

bool column[Q] , d1[D] , d2[D];	ไว้สำหรับหาค่า Fitness
int Chromosome[Population][Q];	ใช้เก็บChromosome ทั้งหมด
int Fitness[Population];	ใช้เก็บค่า Fitness ของแต่ละ Chromosome
int Sort[Population];	ใช้เรียงค่า Fitness ของ Chromosome

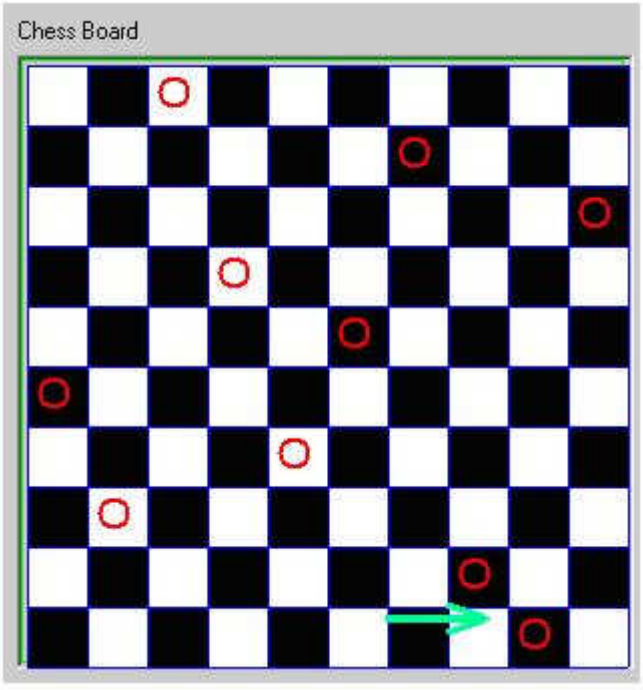
Main Program :

```
void solution_GA() {
    int i;
    if (InitRandom == 1) srand(time(0));
    for ( i = 0 ; i < NumberSolution ; i++ ) {      ทำจนได้จำนวนคำตอบเท่ากับ NumberSolution
       t1 = clock();
       CountCrossOver = 0;
       CountMutation = 0;
       CountGeneration = 0;
       InitPopulation();                            Generate Chromosome ขึ้นมาใหม่ โดยการ Random
       GetFitness(0,Population);                    หาค่า Fitness ของแต่ล่ะ Chromosome
       quickSort(0,Population-1);                   Quick Sort Algorithm เรียงค่า Fitness
       while (Fitness[Sort[0]] !=0 ){               ตรวจสอบChromosome บนสุด ถ้าค่า Fitness เป็น 0 แสดงว่าเจอคำตอบ
            Crossover_one(0);                       ทำการ Cross Over และ หาค่า Fitness ของChromosome ใหม่
            Mutation();                             ทำการ Mutation และ หาค่า Fitness ของChromosome ใหม่
            quickSort(0,Population-1);              เรียง Chromosome ทั้งหมดใหม่
            CountGeneration++;                      เก็บจำนวนGeneration ที่สร้างขึ้นมา
       }
       Printboard(0);
       srand(time(0));
    }
}
```

Initial Population :

```
void InitPopulation() {
    int i,a,b,chk,temp;
    for ( i = 0 ; i < Population ; i++ ) {
       Sort[i]=i;                                    ใส่ค่า Index
       for ( a = 0 ; a < Q ; a++ )
          column[a] = false;
       for ( a = 0 ; a < Q ; a++ ){
          temp = rand()%Q;
          while ( column[temp] ) temp = rand()%Q;    เมื่อมีค่าตัวเลขซ้ำจะทำการ Random ใหม่
          column[temp] = true;                       ทำการบอกว่าตัวเลขที่ได้ถูกเลือกแล้ว
          Chromosome[i][a]= temp;                    ใส่ค่าตัวเลขที่ได้
       }
    }
}
```

Function นี้จะทำการ Generate Chromosome จำนวนเท่ากับค่า Population ที่ตั้งไว้ ขนาดของ Chromosome เท่ากับค่า Q ตัวอย่างเช่น Chromosome[i] = { 2,6,9,3,5,0,4,1,7,8 } จะได้ตารางดังรูป



จากรูปตารางนี้มีค่า Fitness เท่ากับ 1 เนื่องจากมีการทับกับแนวทแยงของสองตัวล่างถ้าไม่มีการทับกันเลย ค่า Fitness จะเท่ากับ 0 ดังนั้นเราจะประเมินว่าถ้าค่า Fitness น้อยจะเป็นChromosome ที่ดี แต่ถ้าค่ามาก จะถือว่าเป็น Chromosome ที่ไม่ดี

Fitness Function :

```
void GetFitness(int pos, int e) {
    int i,a,index;
    for (index = pos ; index < e ; index++ ) {
       for (i = 0; i < D; i++) {                                    เคลียร์ค่าตรวจสอบแนวทแยง
          d1[i]=false;
          d2[i]=false;
       }
       Fitness[index] = 0;
       for (i = 0 ; i < Q ; i++) {
          if (d1[Chromosome[index][i] + i]) Fitness[index]++;        เมื่อมีการทับกันค่า Fitness จะเพิ่มขึ้น
          if (d2[Chromosome[index][i] + Q -1 - i]) Fitness[index]++;
          d1[Chromosome[index][i] + i] = true;                      กำหนดว่าแนวนี้มี Queen วางไว้แล้ว
          d2[Chromosome[index][i] + Q -1 - i] = true;
       }
    }
}
```

Function นี้ใช้หาค่า Fitness ของแต่ล่ะChromosome โดยถ้ามีการทับกันของ Queen เยอะ ค่าFitness จะมีค่ามาก แต่ถ้ามีการทับกันของตาเดินน้อยค่า Fitness จะมีค่าน้อยและถ้าไม่มีเลยค่า Fitness จะมีค่าเท่ากับ 0 ซึ่งเป็น Sulotion คำตอบของปัญหา N-Queen

Quick Sort :

```
int partition( int l, int r) {
  int pivot, i, j, t;
  pivot = Fitness[Sort[l]];
  i = l; j = r+1;
  while( 1) {
        do ++i; while( Fitness[Sort[i]] <= pivot && i <= r );
        do --j; while( Fitness[Sort[j]] > pivot );
        if( i >= j ) break;
        t = Sort[i]; Sort[i] = Sort[j]; Sort[j] = t;
  }
  t = Sort[l]; Sort[l] = Sort[j]; Sort[j] = t;
  return j;
}

void quickSort( int l, int r) {
  int j;
  if( l < r ) {
```

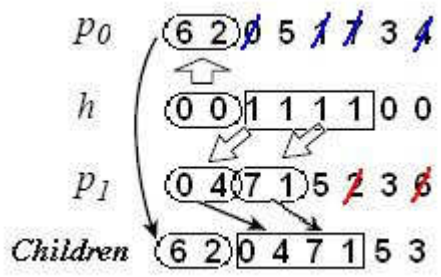
```
        // divide and conquer
      j = partition( l, r);
      quickSort( l, j-1);
      quickSort( j+1, r);
   }
}
```

เป็น Function ในการเรียงลำดับของ Chromosome ซึ่งจะเรียงลำดับ Chromosome จากค่า Fitness จากน้อยไปมาก โดย Function ใช้ Algorithm Quick Sort เรียง Array ของ Sort[Population] โดย Array นี้จะเก็บค่าตำแหน่งของ Chromosome

Cross Over :

```
void Crossover_one(int pos) {
    int i,a,temp,b,chk,p[2],temp2;
    for ( i = pos ; i < (Population-1)/2; i++) {
       CountCrossOver++;
       p[0]=0;
       p[1]=0;
       for ( a=0 ; a<Q ; a++) column[a]=false;
       for ( a=0;  a< D; a++) {
          d1[a]=false;
          d2[a]=false;
       }
       Fitness[Sort[Population-i-1]]=0;
       for ( a=0 ; a<Q ; a++){
          temp=rand()%2;
          while (column[Chromosome[Sort[i+temp]][p[temp]]])
               if (p[temp] < Q-1 ) p[temp]++; else temp = (temp+1)%2;

          Chromosome[Sort[Population-i-1]][a]= Chromosome[Sort[i+temp]][p[temp]];
          column[Chromosome[Sort[Population-i-1]][a]]=true;
          // Get Fitness
          if (d1[Chromosome[Sort[Population-i-1]][a]+a]) Fitness[Sort[Population-i-1]]++;
          if (d2[Chromosome[Sort[Population-i-1]][a]+Q-1-a]) Fitness[Sort[Population-i-1]]++;
          d1[Chromosome[Sort[Population-i-1]][a]+a]=true;
          d2[Chromosome[Sort[Population-i-1]][a]+Q-1-a]=true;
          if (p[temp] < Q-1 ) p[temp]++; else temp = (temp+1)%2;
       }
    }
}
```

เป็น Function ทำการ Cross Over เพื่อผลิต Chromosome รุ่นต่อไปโดยจะนำ Chromosome ลำดับที่ 0 ถึง ลำดับที่ (Population-1)/2-1 ซึ่งมีค่า Fitness น้อยๆ นำมาเป็น Parent และใส่กลับไปที่ Chromosome ลำดับท้ายๆ ซึ่งมีค่า Fitness มาก วิธีการ Cross Over คือ ทำการ Random H จำนวน Q ตัวแต่ละตัวมีค่า 0 หรือ 1 แล้วนำ h มาทำการ Cross Over ระหว่าง Parent P0 และ P1 โดยถ้า h เป็น 0 จะเลือกค่าจาก P0 มาใส่ ถ้าเป็น 1 จะเลือกค่าจาก P1 มาใส่ ดังรูป พร้อมทั้งคำนวณค่า Fitness ไปด้วย



Mutation :

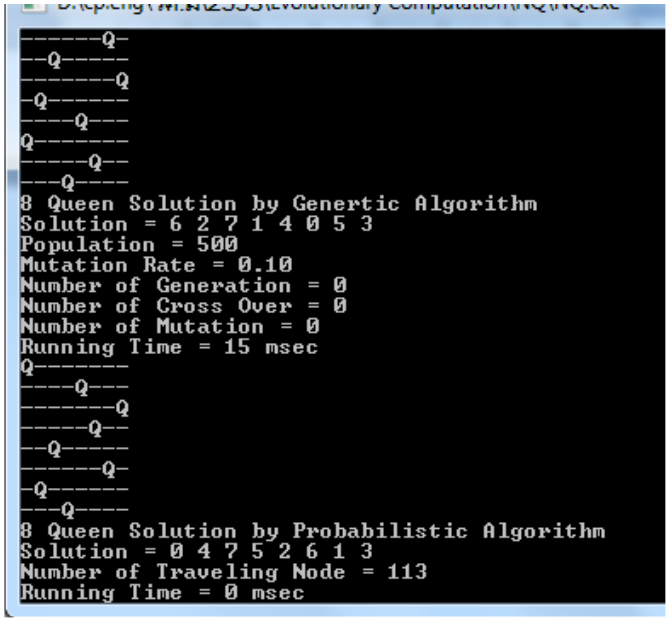
```
void Mutation(){
   int randomChromosome;
        int randomGen0,randomGen1;
        int Temp;
        int NumberOfMutation=int(MutationRate*Population);
        for(int k=0;k<NumberOfMutation;k++) {
                CountMutation++;
                randomChromosome=rand()%Population;
                while((randomChromosome=rand()%Population)==0);// random chromosome exept number 0

                randomGen0=rand()%Q;// random genes from chromosome
                while((randomGen1=rand()%Q)==randomGen0);
                // Apply Mutation
                Temp=Chromosome[Sort[randomChromosome]][randomGen0];
                Chromosome[Sort[randomChromosome]][randomGen0]=Chromosome[Sort[randomChromosome]]
[randomGen1];
                Chromosome[Sort[randomChromosome]][randomGen1]=Temp;
                GetFitness(Sort[randomChromosome],Sort[randomChromosome]+1);
        }
}
```

เป็น Function ในการกลายพันธุ์จำนวนการกลายพันธ์ เท่ากับ MutationRate*Population แล้วจะทำการ Random Chromosome ที่จะทำการ Mutation แต่จะไม่เลือก Chromosome ที่ 0 เนื่องจากมีค่า Fitness น้อยที่สุดและอาจเป็น Solution ได้ เมื่อได้ Chromosome ที่ทำการ Mutation แล้ว จะทำการRandom ข้อมูลในChromosome มาสองตัวที่ตำแน่งไม่ซ้ำกันแล้วทำการสลับที่กัน แล้วหาค่า Fitness ของChromosome ที่ได้ใหม่

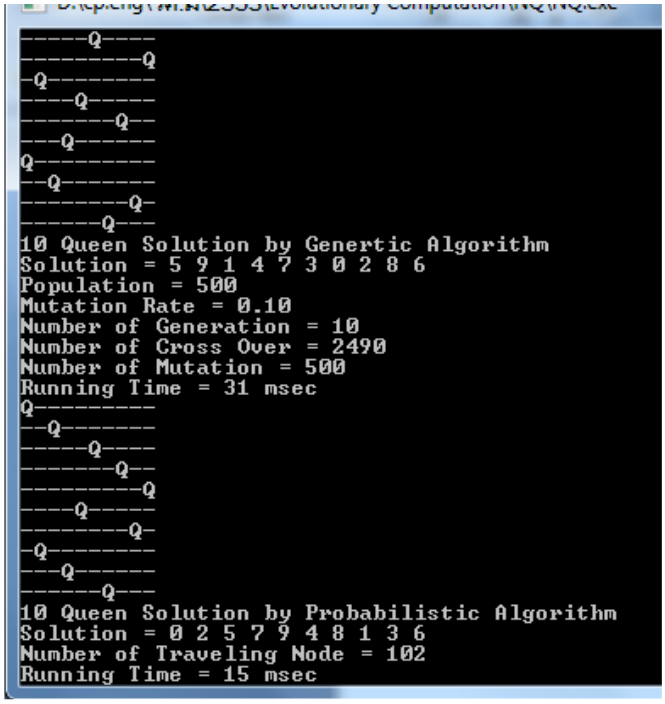
ทำการ Cross Over , Mutation , Quick Sort ไปจนกว่า ค่าFitness ของ Chromosome ลำดับ 0 ซึ่งมีค่า Fitness น้อยที่สุดมีค่าเป็น 0 ซึ่งเป็น Solution ของคำตอบ N-Queen

ผลการทดลอง

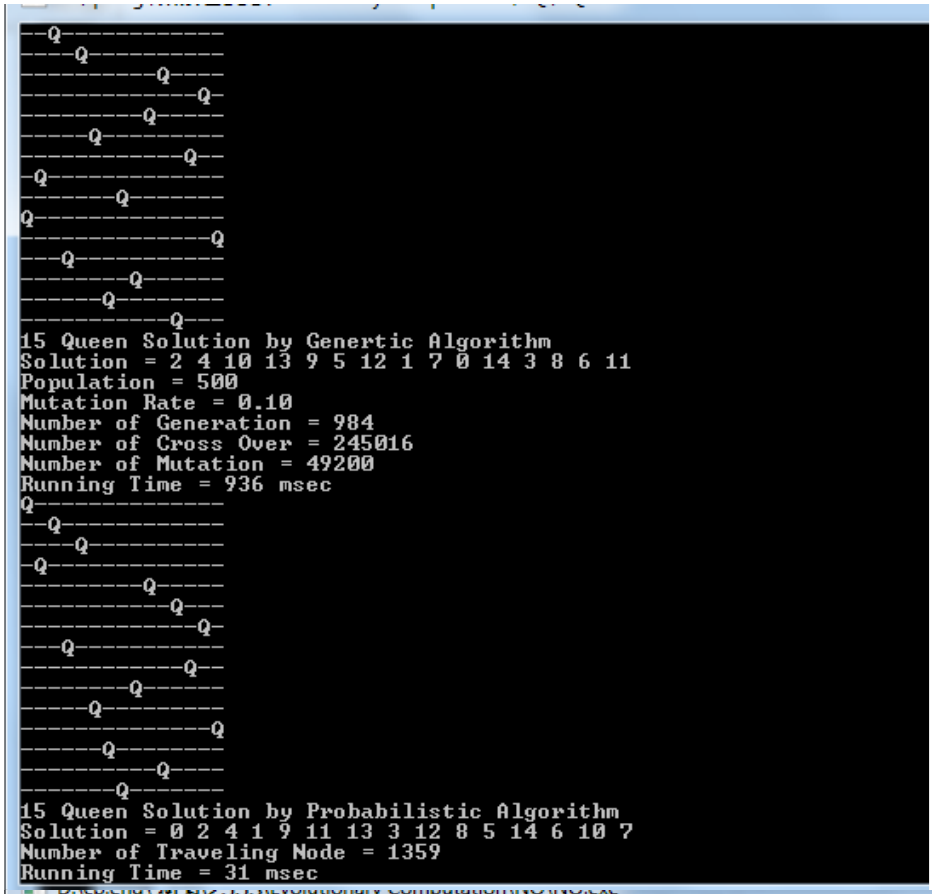
ทำการเปรียบเทียบกับวิธี Probabilistic Algorithm

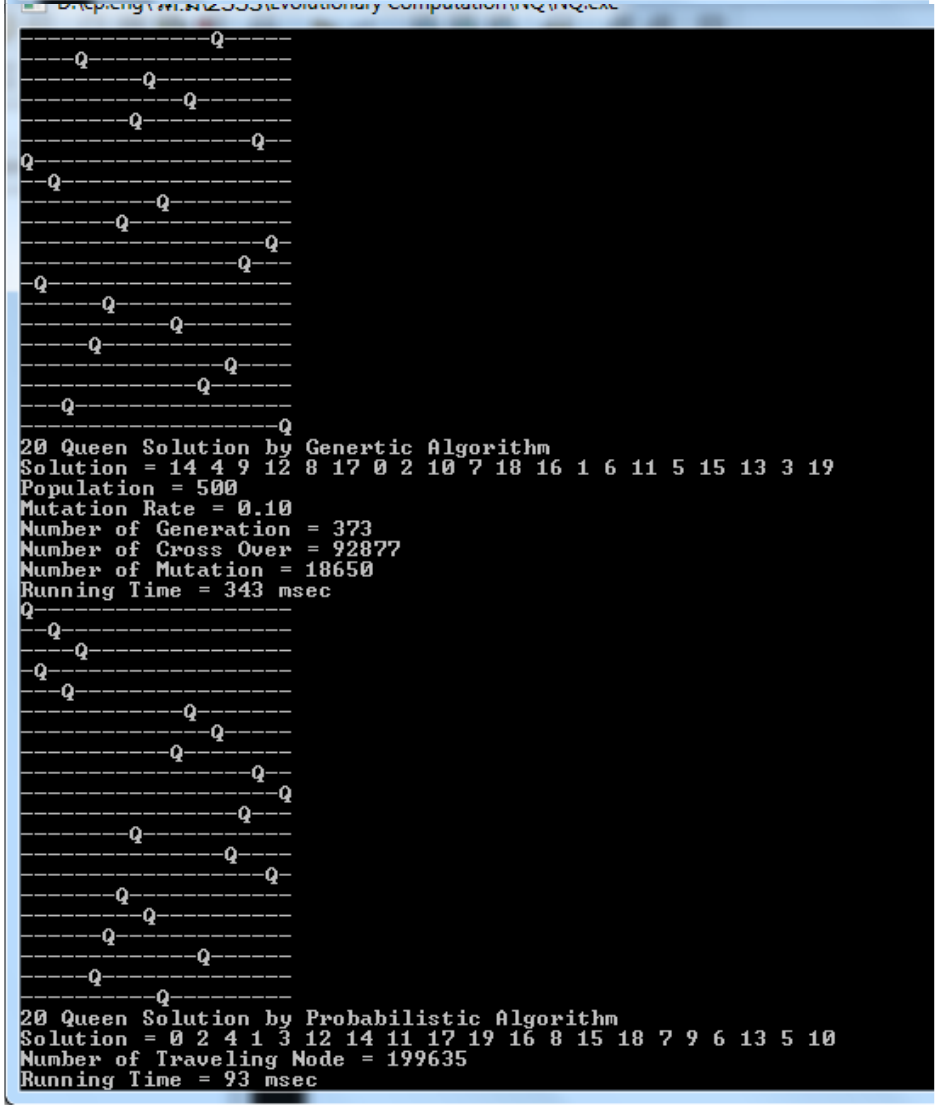


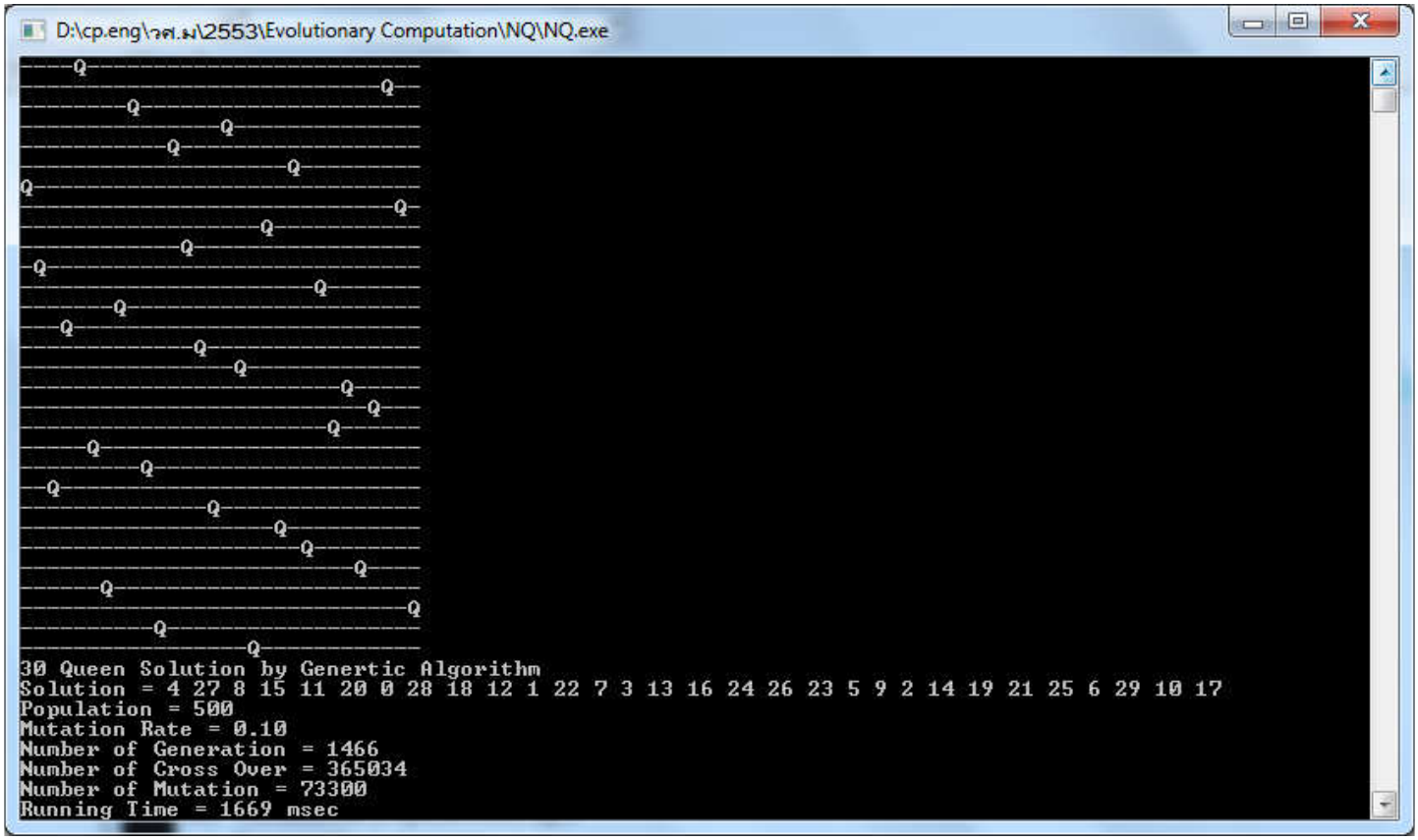
Q = 8

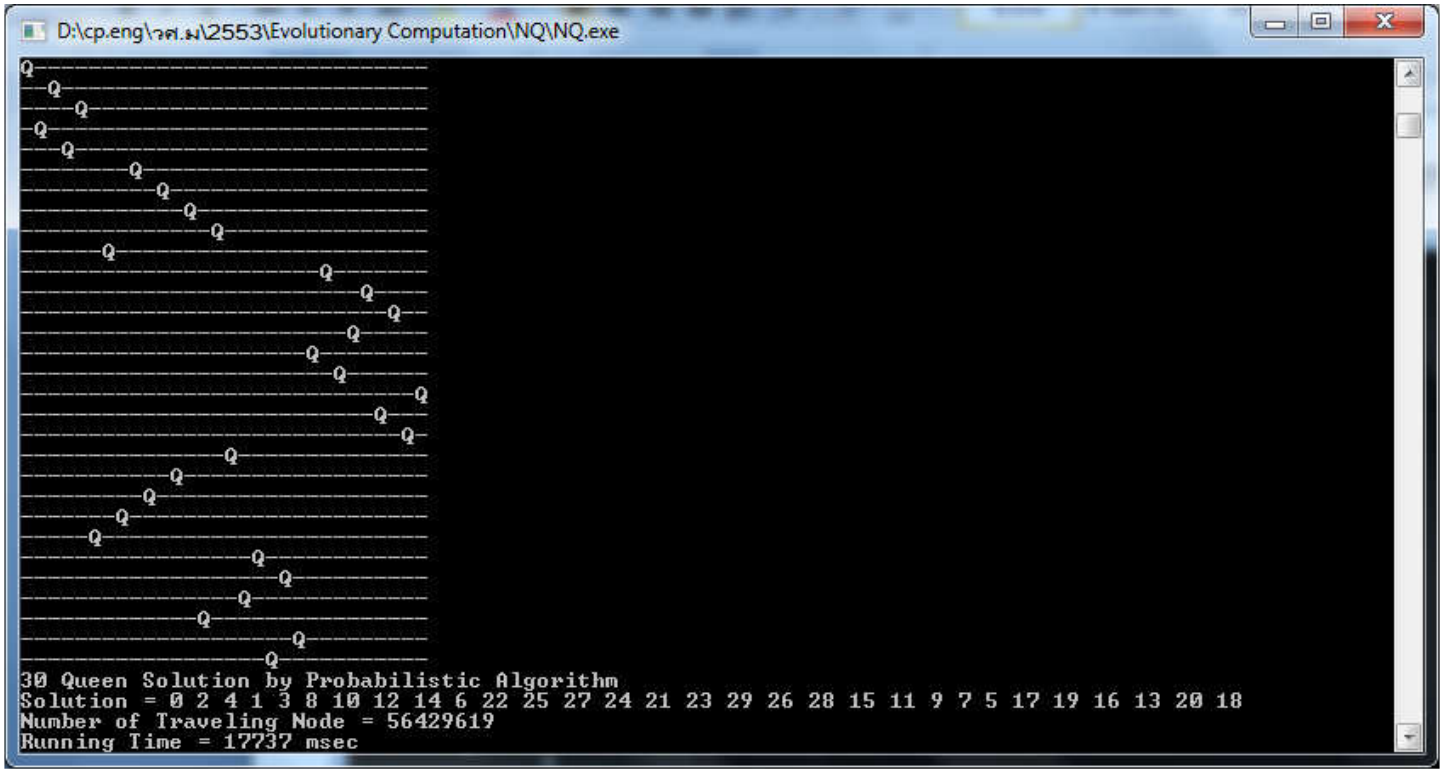


Q = 10

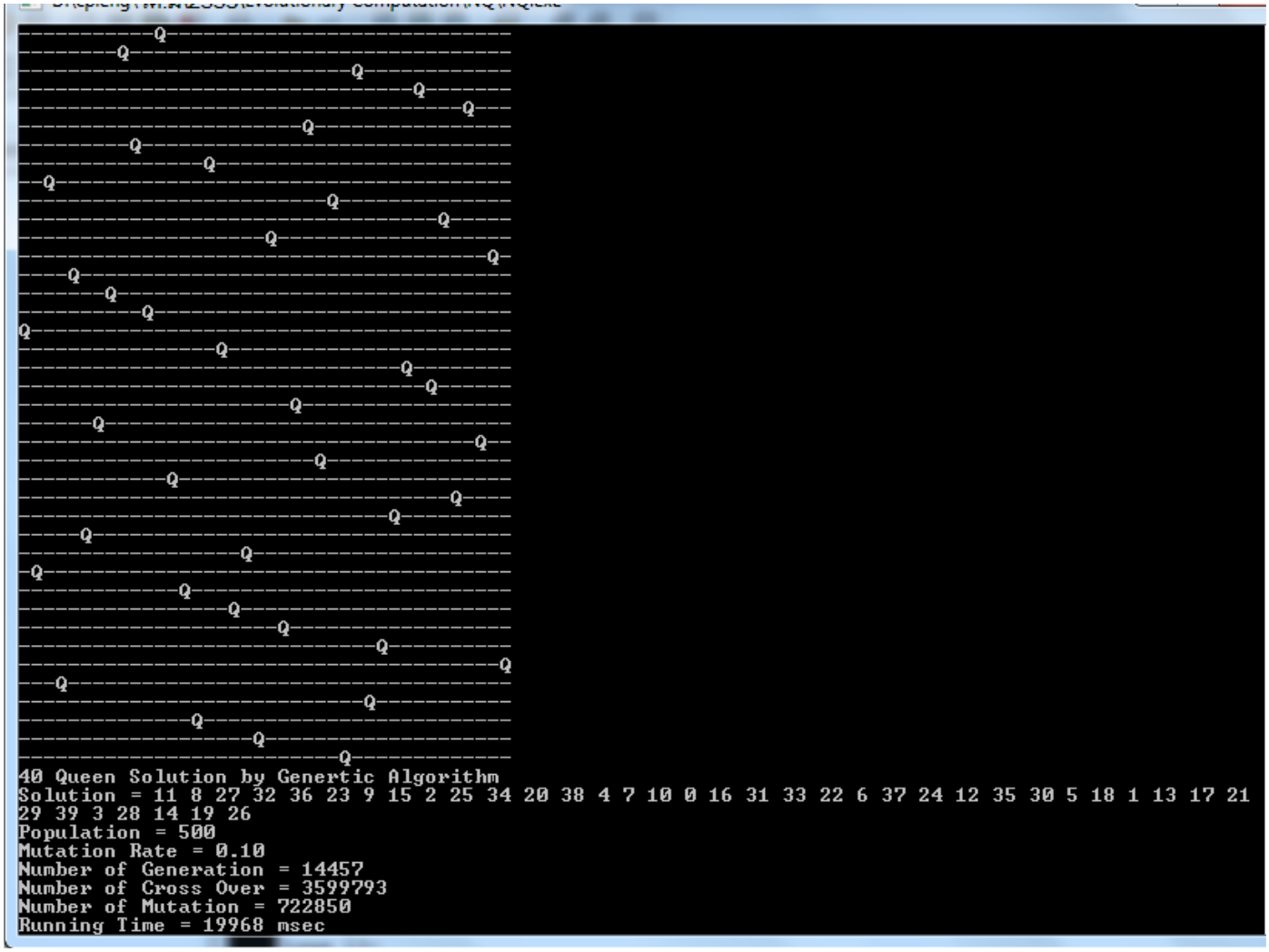

--Q-------------
----Q-----------
----------Q-----
-------------Q-
---------Q------
-----Q----------
------------Q---
-Q--------------
-------Q-------
Q---------------
--------------Q
---Q------------
--------Q------
------Q--------
-----------Q---
15 Queen Solution by Genertic Algorithm
Solution = 2 4 10 13 9 5 12 1 7 0 14 3 8 6 11
Population = 500
Mutation Rate = 0.10
Number of Generation = 984
Number of Cross Over = 245016
Number of Mutation = 49200
Running Time = 936 msec
Q---------------
--Q-------------
----Q-----------
-Q--------------
---------Q------
-----------Q---
-------------Q-
---Q------------
------------Q---
--------Q------
-----Q----------
--------------Q
------Q---------
----------Q-----
-------Q--------
15 Queen Solution by Probabilistic Algorithm
Solution = 0 2 4 1 9 11 13 3 12 8 5 14 6 10 7
Number of Traveling Node = 1359
Running Time = 31 msec


--------------Q-----
----Q-----------------
---------Q-----------
------------Q--------
--------Q------------
------------------Q--
Q-------------------
--Q-----------------
----------Q---------
-------Q------------
-------------------Q-
-----------------Q---
-Q------------------
------Q------------
----------Q--------
-----Q--------------
----------------Q---
-------------Q------
---Q-----------------
--------------------Q
20 Queen Solution by Genertic Algorithm
Solution = 14 4 9 12 8 17 0 2 10 7 18 16 1 6 11 5 15 13 3 19
Population = 500
Mutation Rate = 0.10
Number of Generation = 373
Number of Cross Over = 92877
Number of Mutation = 18650
Running Time = 343 msec
Q-------------------
--Q-----------------
----Q---------------
-Q------------------
---Q----------------
------------Q-------
--------------Q-----
----------Q---------
-----------------Q--
--------------------Q
----------------Q---
--------Q-----------
---------------Q----
-----------------Q-
-------Q------------
---------Q----------
------Q------------
------------Q------
-----Q--------------
----------Q--------
20 Queen Solution by Probabilistic Algorithm
Solution = 0 2 4 1 3 12 14 11 17 19 16 8 15 18 7 9 6 13 5 10
Number of Traveling Node = 199635
Running Time = 93 msec

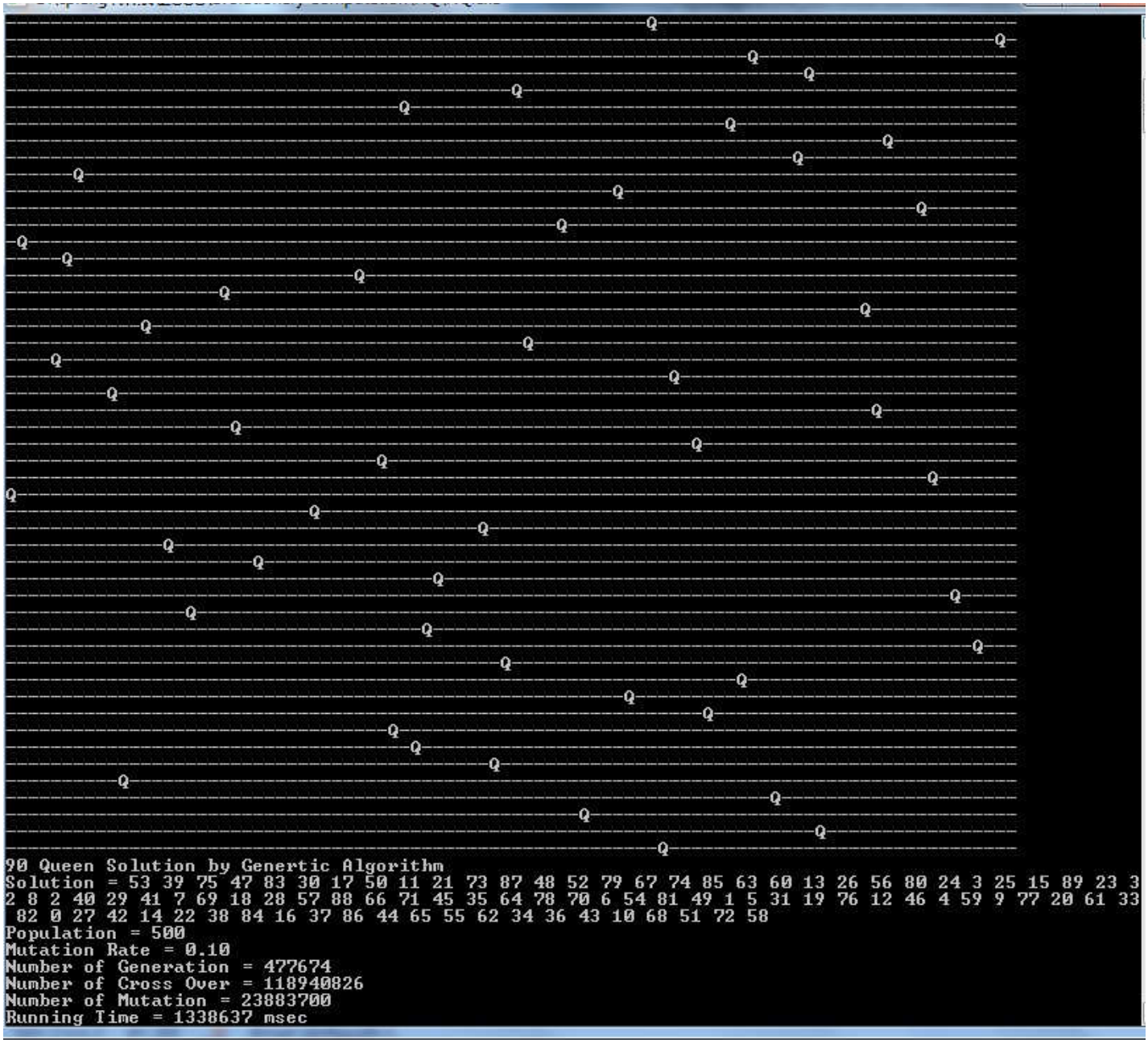




เมื่อ Q = 30 Genetic Algorithm มีความเร็วกว่า



และเมื่อ Q = 40 GA สามารถหาคำตอบได้ ในเวลา 20 วินาที แต่ Probabilistic Algorithm ใช้เวลานานมากก็ยังไม่มีคำตอบ



Q = 90 Genetic Algorithm สามารถหาคำตอบได้ !!!!!!!!

ได้ทำการทดลอง Set ค่า Population และ Mutation Rate ด้วยค่าต่างๆแล้วจากการทดลองพบว่า
ค่า Population = 500 และ Mutation Rate = 0.1 ทำให้มีความเร็วมากที่สุด