МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ
по СИИТ
по лабораторной работе №5

Выполнил:

студентка группы АС-32
Предко А.И.

Проверил:

Акира Имада

Брест 2014

Лабораторная работа №5
Генная инженерия

Задание 1
Создать 6 случайных городов, найти матрицу пути, построить карту, и просчитать все возможные пути.

Текст программы

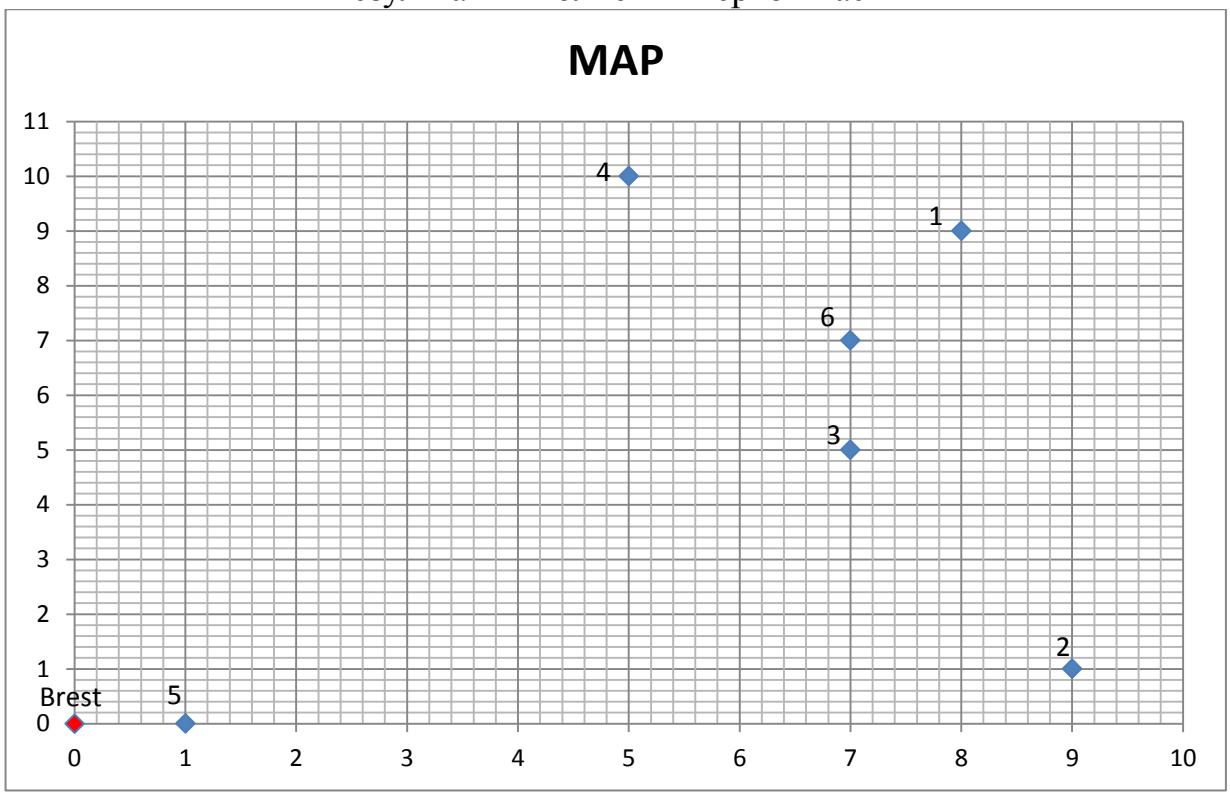
```
#include <QCoreApplication>
#include <stdio.h>
#define N 6

int main(int argc, char *argv[])
{
    struct Coord
        {
        int x;
        int y;
        };
    Coord Cities[7];
    float Distance[7][7];
    FILE *Way;
    FILE *dist;
    Way = fopen("C:\\way.txt", "wt");
    dist = fopen("C:\\dist.txt", "wt");
    FILE *All;
    All = fopen("C:\\all.txt", "wt");
    int temp1;
    int temp2;
    Cities[0].x = 0;
    Cities[0].y = 0;
    int flag = 0;
    int m, l;
    //Создание случайных городов
    for (m = 1; m<7; m++)
        {
         do{
            flag = 0;
            temp1 = rand()%11;
            temp2 = rand()%11;
            for( l = 0; l < m; l++)
                {
                if ( Cities[l].x == temp1 && Cities[l].y == temp2)
                    {
                    flag++;
                    }
                }
            }while(flag != 0);
        Cities[m].x = temp1;
        Cities[m].y = temp2;

        fprintf(Way, "%d\t%d\n", Cities[m].x, Cities[m].y);
        }
    fclose(Way);

    //Матрица расстояний
    for (m = 0; m < 7; m++ )
        {
        for( l = 0; l < 7; l++ )
            {
            if ( m == l )
                {
```

```
                Distance[m][l] = 0;
                }
            else
                {
                Distance[m][l] = sqrt( pow( (Cities[m].y - Cities[l].y), 2 )
+
                    pow( (Cities[m].x - Cities[l].x), 2) );
                }
            fprintf(dist, "%f\t", Distance[m][l]);
            }
        fprintf(dist, "\n");
        }
    fclose(dist);

    int Arr[720][N];
    FILE * Variant;
    Variant = fopen("C:\\variant.txt", "wt");
    QCoreApplication a(argc, argv);
    int s[N], t; int i, j, r, k;
    int c = 0;
     for(i=0; i<N; i++) s[i] = 1 + i;
     //s[N] = '\0';
     while(1) {
         for (i = 0; i<N; i++) {
             fprintf(Variant, "%d", s[i]);
             Arr[c][i]=s[i];}
         c++;
      // printf("%s\n", s);  // Вывод очередной перестановки
         fprintf(Variant, "\n");
           // Находим самое правое место, где s[i] < s[i+1]
       for(i=N-1; i>=0 && s[i] > s[i+1]; --i) ;
       if (i<0) break; // Уже получили "654321" - самую старшую перестановку
           // Находим s[j] - наименьший элемент справа от s[i] и больший его
       for(j=N-1; s[i] > s[j]; j--) ;
           // Меняем s[i] <-> s[j]
       t = s[j];
       s[j] = s[i];
       s[i] = t;
           // То, что за "i" - переворачиваем
       for(k=i+1, r=N-1; r > k; k++, r--) {
         t = s[r];
         s[r] = s[k];
         s[k] = t;
       }
     }
    fclose (Variant);



    //Поиск пути
        float fit[720];
    for(i=0; i<720; i++) fit[i] = 0;
    int idx, idx1;
    for(i =0; i<720; i++)
        {
        idx = Arr[i][0];
         fit[i]+=Distance[0][idx];
        for(l=0; l<5; l++)
         {
            idx = Arr[i][l];
            idx1 = Arr[i][l+1];
            fit[i]+=Distance[idx][idx1];
         }
        idx = Arr[i][5];
```

```
      fit[i]+=Distance[0][idx];
      }


 //Сортировка
float temp;
 for(i=0; i<719; i++)
      {
      for(j=0; j<719; j++)
          {
          if ( fit[j] > fit[j+1])
              {
              temp = fit[j+1];
              fit[j+1]= fit[j];
              fit[j]= temp;
              for( k = 0; k<6; k++)
                  {
                  temp = Arr[j+1][k];
                  Arr[j+1][k]= Arr[j][k];
                  Arr[j][k]= temp;
                  }
              }
          }
      }

 for (m = 0; m<720; m++)
 {
      if ( m%2 == 0)
      {
      for(l = 0; l<6; l++)
      {
          fprintf(All, "%d",  Arr[m][l]);
      }

     fprintf(All, "\t%f\n", fit[m]);
      }
 }
 fclose(All);


 Coord Cities2[31];
 Cities2[0].x = 0;
 Cities2[0].y = 0;

 FILE* way2;
 way2 = fopen("C:\\way2.txt", "wt");
 // Создание случайных городов
 for(m=1; m<31; m++)
 {
      do{
         flag = 0;
         temp1 = rand()%11;
         if(temp1 == 0 || temp1 == 10)
         {
         temp2 = rand()%11;
         }
         else
         {
             temp2 = rand()%2;
             if (temp2 == 1)
             {
                 temp2 = 10;
             }
         }
```

```
            for( l = 0; l < m; l++)
                {
                if ( Cities2[l].x == temp1 && Cities2[l].y == temp2)
                    {
                    flag++;
                    }
                }
            }while(flag != 0);
        Cities2[m].x = temp1;
        Cities2[m].y = temp2;

        fprintf(way2, "%d\t%d\n", Cities2[m].x, Cities2[m].y);
    }


    fclose(way2);
    return a.exec();
}
```

Результат выполнения первой части



Distance Matrix

0.000000	12.041595	9.055386	8.602325	11.180340	1.000000	9.899495
12.041595	0.000000	8.062258	4.123106	3.162278	11.401754	2.236068
9.055386	8.062258	0.000000	4.472136	9.848858	8.062258	6.324555
8.602325	4.123106	4.472136	0.000000	5.385165	7.810250	2.000000
11.180340	3.162278	9.848858	5.385165	0.000000	10.770329	3.605551
1.000000	11.401754	8.062258	7.810250	10.770329	0.000000	9.219544
9.899495	2.236068	6.324555	2.000000	3.605551	9.219544	0.000000

All ways

Way	Distance
416325	32.113079
236145	32.696198

523416	34.217400
523146	34.324821
523641	34.343815
364125	34.494667
234165	34.530575
231465	34.638000
461325	34.679459
214635	34.695721
236415	34.697105
526143	34.772648
314625	34.880074
361425	34.911789
261435	34.973701
531462	35.081123
241635	35.112839
231645	35.262577
413625	35.852539
416235	36.185627
263145	36.435654
326145	36.567692
164325	36.802773
346125	36.953625
216435	37.154678
234615	37.156059
316425	37.478165
521436	37.571453
246135	37.679218
214365	37.884628
436125	37.926086
526341	37.975849
532146	38.011967
463125	38.033512
524136	38.095993
321465	38.124092
431625	38.311493
263415	38.329136
532164	38.366600
241365	38.409172
532641	38.416363
532416	38.429085
216345	38.509205
561423	38.541206
514623	38.568596
213645	38.616631
321645	38.748669
261345	38.894608
412365	39.096554
541236	39.366497
413265	39.481956
412635	39.539680

541326	39.751900
362145	39.921745
521346	40.137833
213465	40.451008
134625	40.542229
613425	40.554947
163425	40.573944
136425	40.681366
243165	40.868126
264315	40.895515
243615	40.927231
164235	41.014458
246315	41.034653
324615	41.166695
432165	41.555511
143265	41.605270
534126	41.643997
514326	41.645382
463215	41.722038
142365	41.744411
341265	41.756123
514236	41.784519
543216	41.825451
123645	41.951866
432615	42.000015
423165	42.080051
531264	42.106056
462315	42.107445
421635	42.137772
516324	42.139153
531426	42.168541
142635	42.187534
162345	42.229847
314265	42.280666
132645	42.337273
362415	42.339767
542316	42.349991
163245	42.368984
312645	42.488125
361245	42.519836
426135	42.523174
542613	42.905239
253614	43.506580
123465	43.786247
512346	43.826359
254163	43.888645
126435	44.229370
534216	44.242092
132465	44.310787
531246	44.349518

513246	44.350899
342165	44.354218
124635	44.368511
346215	44.381607
312465	44.461643
364215	44.520744
162435	44.646492
516243	44.798725
431265	45.294968
436215	45.354073
421365	45.434105
543126	45.564907
126345	45.583900
253416	45.610897
542136	45.704044
253146	45.718323
256143	45.723026
146352	45.737316
256413	45.830448
426315	45.878609
251463	45.889553
461352	46.072956
136245	46.108440
254613	46.455021
256314	46.802910
325614	46.934948
254136	47.072849
235614	47.136002
325416	47.204887
352614	47.378075
235416	47.405941
124365	47.557419
512436	47.597530
134265	47.942822
261453	47.961189
513426	47.982933
164352	48.196270
346152	48.348507
143652	48.926228
251436	48.966335
146523	49.165688
352146	49.204414
325146	49.205799
256134	49.261868
251634	49.320972
146532	49.366741
235146	49.406849
251364	49.428394
214653	49.517590
613452	49.531803

163452	49.550800
352164	49.559052
461523	49.560432
146253	49.608810
614253	49.621529
136452	49.658222
235164	49.761486
164523	49.790260
241653	49.934708
264153	49.962097
164532	49.991314
413256	50.119156
354612	50.142162
412536	50.176876
415236	50.178261
365214	50.289001
145236	50.408092
236514	50.491436
413526	50.562279
231456	50.702271
145632	50.721268
362514	50.733509
452316	50.743729
214536	50.759995
163254	50.762726
214563	50.872120
265314	50.875458
361254	50.913578
263514	50.934559
625413	50.944344
145263	50.963341
231654	51.056908
453612	51.114628
236154	51.116013
145362	51.164391
265413	51.257523
316254	51.298981
134652	51.492603
453162	51.500031
643152	51.532711
164253	52.206902
143256	52.242466
341256	52.393322
246153	52.560192
214356	52.594372
234156	52.595753
165234	52.597107
435216	52.635830
432516	52.637215
143526	52.685593

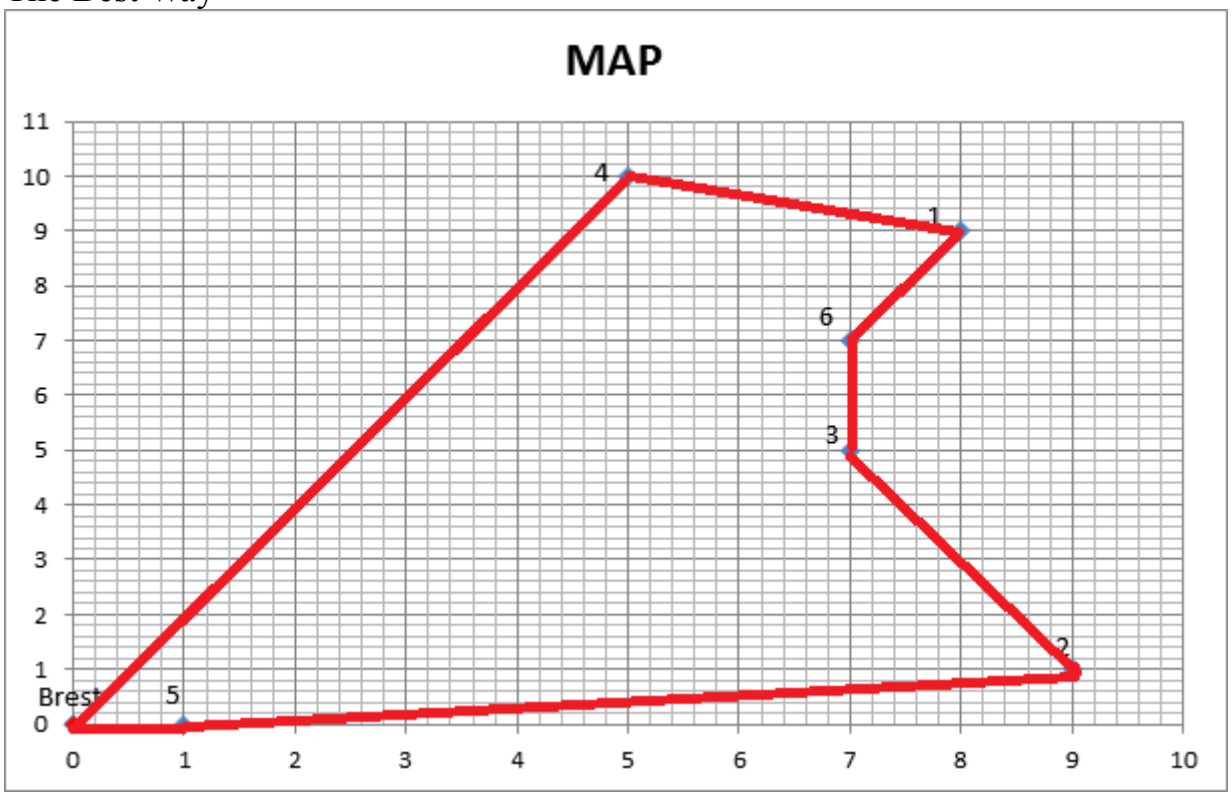
132564	52.704529
125364	52.762253
152364	52.763634
142536	52.824734
341526	52.837826
465213	52.855381
314256	52.917862
142563	52.936859
435612	52.949009
132546	52.974468
143562	52.998772
345216	53.017895
435261	53.040230
465312	53.056431
465132	53.057816
463512	53.115536
241356	53.118916
645213	53.125317
125463	53.144318
135264	53.147655
341562	53.151005
425316	53.160374
241536	53.178020
163524	53.179367
234561	53.180222
234516	53.220333
316524	53.272499
241563	53.290146
462513	53.299889
213546	53.326374
231546	53.327759
345612	53.331074
361524	53.331604
236451	53.346748
435162	53.393517
345261	53.422295
364512	53.497601
135462	53.730770
245316	53.743488
163542	53.762486
261543	53.775585
412356	53.806297
245613	53.855614
264513	53.883007
245163	53.914719
452136	54.097782
321456	54.188362
623514	54.250805
356214	54.361546
326514	54.362934

213654	54.410961
453216	54.430874
145326	54.480637
136254	54.502182
321654	54.542999
145623	54.592762
354126	54.631489
162354	54.835274
263154	54.855469
643521	54.866570
152346	54.867954
346521	54.978691
326154	54.987511
234651	55.181129
215346	55.219860
346512	55.331982
135246	55.391113
342561	55.395813
346251	55.423203
342516	55.435925
642513	55.543346
152463	55.562340
165342	55.596867
153462	55.624256
243516	55.636974
135642	55.704288
153642	55.763393
246513	55.856525
431256	55.932163
436521	55.951157
125436	56.221100
215634	56.304447
431526	56.376667
123564	56.391674
152634	56.395668
142356	56.454155
136524	56.475700
213456	56.515282
425136	56.515808
465123	56.543907
634512	56.574383
324156	56.606388
134526	56.606503
123546	56.661613
431562	56.689846
165324	56.808792
321546	56.813850
153264	56.836182
615324	56.848900
134562	56.919678

356124	56.959641
154362	56.978783
351264	56.987034
142653	57.009403
351426	57.049511
136542	57.058815
245136	57.098923
165423	57.190857
126453	57.216862
154623	57.218246
354216	57.229580
324516	57.230968
351624	57.404152
162453	57.633980
123654	57.746201
451236	57.786312
132654	58.131603
453126	58.170330
451326	58.171715
126354	58.189323
456213	58.282452
362154	58.341560
134256	58.580017
152436	58.639122
243156	58.933308
156342	58.952301
153246	59.079639
356421	59.190376
156423	59.191765
351246	59.230492
432156	59.620689
123456	59.850517
126534	60.023705
432651	60.025085
435126	60.063816
421356	60.143848
423156	60.145229
156324	60.164227
421536	60.202953
154326	60.295029
365124	60.315079
132456	60.375061
126543	60.405769
124536	60.432781
154236	60.434166
345126	60.445881
312456	60.525913
124563	60.544907
135624	60.548244
153624	60.607349

315624	60.700481
135426	60.818184
315426	60.970421
154263	60.989414
124356	62.267162
342156	62.419399
153426	62.711670

The Best Way



Задание часть 2

30 городов расположены на границе карты (10x10). Знаем оптимальный путь, найти его с помощью генетического алгоритма.

Program

```
#include <QCoreApplication>
#include <iostream>
#include <math.h>
#include <time.h>
#include <QVector>
using namespace std;
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    srand(time(NULL));
   struct Coord
       {
       int x;
       int y;
       };
    FILE *Way;
    FILE *dist;
    Way = fopen("C:\\way.txt", "wt");
   dist = fopen("C:\\dist.txt", "wt");
```

```
FILE *BEST;
         BEST = fopen("C:\\BEST.txt", "wt");
FILE *WORST;
          WORST = fopen("C:\\WORST.txt", "wt");
FILE *B;
          B = fopen("C:\\B.txt", "wt");
FILE *W;
          W = fopen("C:\\W.txt", "wt");

Coord Cities[31];
float Distance[31][31];

int temp1;
int temp2;
Cities[0].x = 0;
Cities[0].y = 0;
int flag = 0;
//Создание карты
for (int i = 1; i<31; i++)
    {

    do{
        flag = 0;
        temp1 = rand()%11;
        if(temp1 == 0 || temp1 == 10)
        {
        temp2 = rand()%11;
        }
        else
        {
            temp2 = rand()%2;
            if (temp2 == 1)
            {
                temp2 = 10;
            }
        }
        for( int l = 0; l < i; l++)
            {
            if ( Cities[l].x == temp1 && Cities[l].y == temp2)
                {
                flag++;
                }
            }
        }while(flag != 0);
    Cities[i].x = temp1;
    Cities[i].y = temp2;

    fprintf(Way, "%d\t%d\n", Cities[i].x, Cities[i].y);
    }
fclose(Way);
//Матрица расстояний
for (int i = 0; i < 31; i++ )
    {
    for( int j = 0; j < 31; j++ )
        {
        if ( i == j )
            {
            Distance[i][j] = 0;
            }
        else
            {
            Distance[i][j] = sqrt( pow( (Cities[i].y - Cities[j].y), 2 ) +
                pow( (Cities[i].x - Cities[j].x), 2) );
```

```
            }
        fprintf(dist, "%f\t", Distance[i][j]);
        }
    fprintf(dist, "\n");
    }
fclose(dist);

int firstArray[100][30];
float fitness[100];
int way[30];
int City[31];
for(int i = 0; i<31; i++)
    {
    City[i]=i+1;
    }

for(int i = 0; i<100; i++)
    {
    for(int j = 0; j<30; j++)
        {
        firstArray[i][j]=rand()%30+1;
        }
    }

for(int i = 0; i<100; i++)
    {
    fitness[i]=0;
    }

for(int i =0; i<100; i++)
    {
     int n = 30;
     for(int j = 0; j<30; j++)
         {
         int coun= firstArray[i][j];
         while(coun>=n)
             {
             coun-=n;
             }

         way[j]=City[coun];
         for(int k = coun; k < n; k++)
             {
             City[k]=City[k+1];
             }
         n--;
         }
     fitness[i]+=Distance[0][way[0]];

    for(int l=0; l<29; l++)
     {
        fitness[i]+=Distance[way[l]][way[l+1]];
     }
    fitness[i]+=Distance[0][way[29]];
    for(int i = 0; i<31; i++)
        {
        City[i]=i+1;
        }
    }

float temp;
//Сортировка
for(int i=0; i<99; i++)
    {
```

```
    for(int j=0; j<99; j++)
        {
        if ( fitness[j] > fitness[j+1])
            {
            temp = fitness[j+1];
            fitness[j+1]= fitness[j];
            fitness[j]= temp;
            for( int k = 0; k<30; k++)
                {
                temp = firstArray[j+1][k];
                firstArray[j+1][k]= firstArray[j][k];
                firstArray[j][k]= temp;
                }
            }
        }
    }

for(int i = 0; i<100; i++)
    {
    cout<<fitness[i]<<'\t';
    }
cout<<endl;

int n = 30;
for(int j = 0; j<30; j++)
    {
    int coun= firstArray[0][j];
    while(coun>=n)
        {
        coun-=n;
        }

    way[j]=City[coun];
    for(int k = coun; k < n; k++)
        {
        City[k]=City[k+1];
        }
    n--;
    }

for(int i = 0; i<31; i++)
    {
    City[i]=i+1;
    }

for(int i = 0; i<30; i++)
    {
    cout<<way[i]<<'\t';
    }
cout<<endl;

for( int cikle = 0; cikle<10000; cikle++)
    {
    int Child[100][30];

    for ( int i = 0; i < 100; i++ )
        {
        int first = rand()%50;
        int second;
        do
            {
            second = rand()%50;
            }
        while(first == second);
```

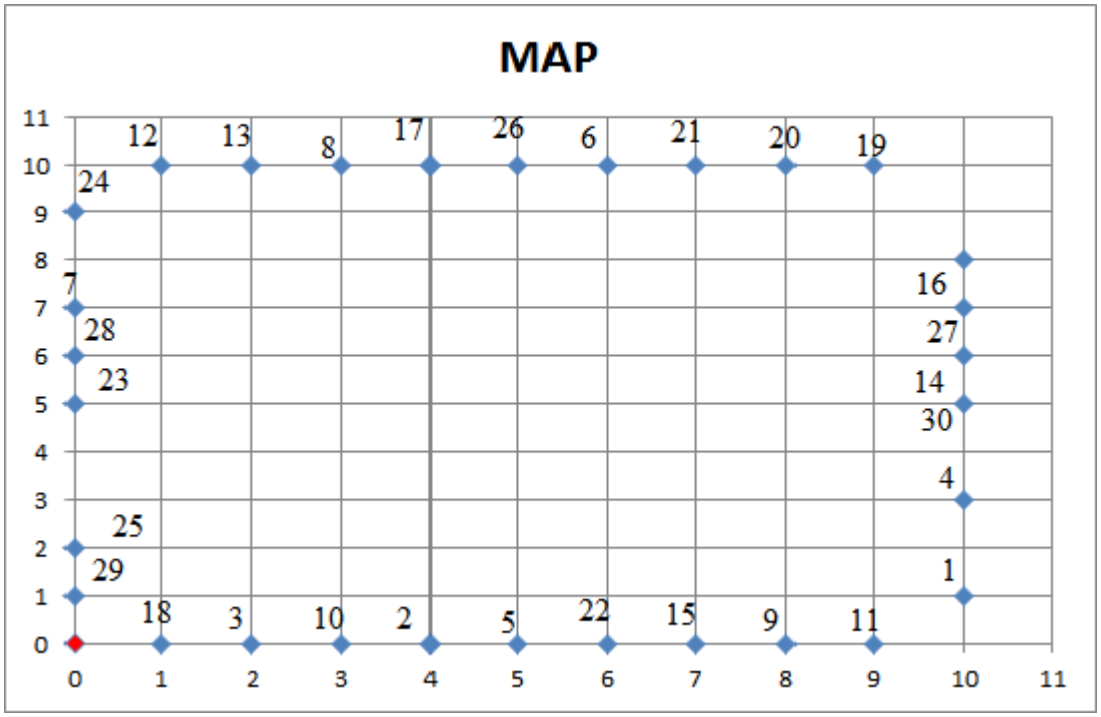
```
int ch1[30];
int ch2[30];
float summ1 = 0;
float summ2 = 0;
int way1[30];
int way2[30];

int t = rand()%30;
for( int k = 0; k<t; k++)
    {
    ch1[k] = firstArray[first][k];
    ch2[k] = firstArray[second][k];
    }
for(int k = t; k<30; k++)
    {
    ch1[k] = firstArray[second][k];
    ch2[k] = firstArray[first][k];
    }

int n = 30;
for(int j = 0; j<30; j++)
    {
    int coun= ch1[j];
    while(coun>=n)
        {
        coun-=n;
        }

    way1[j]=City[coun];
    for(int k = coun; k < n; k++)
        {
        City[k]=City[k+1];
        }
    n--;
    }
int ver_mut = rand()%100;
if(ver_mut<15)
{
    int gen1 = rand()%30;
    int memmory = way1[gen1];
    for (int l = gen1; l<29; l++)
    {
        way1[l]=way1[l+1];
    }
    int gen2 = rand()%30;
    for( int l = 29; l>gen2; l--)
    {
        way1[l] = way1[l-1];
    }
    way1[gen2]=memmory;
}

summ1+=Distance[0][way1[0]];

for(int l=0; l<29; l++)
    {
    summ1+=Distance[way1[l]][way1[l+1]];
    }
summ1+=Distance[0][way1[29]];
for(int l = 0; l<31; l++)
    {
    City[l]=l+1;
    }
```

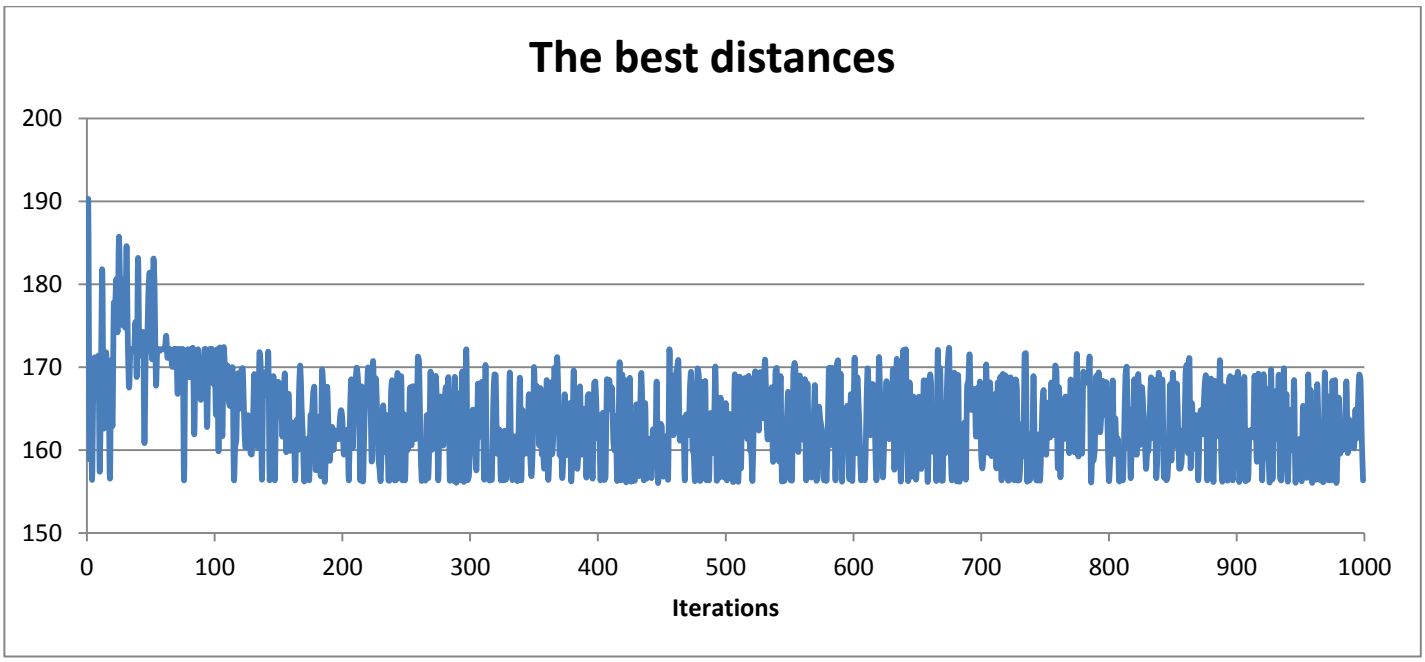
```
    n = 30;
    for(int j = 0; j<30; j++)
        {
        int coun= ch1[j];
        while(coun>=n)
            {
            coun-=n;
            }

        way2[j]=City[coun];
        for(int k = coun; k < n; k++)
            {
            City[k]=City[k+1];
            }
        n--;
        }
    if(ver_mut<15)
    {
        int gen1 = rand()%30;
        int memmory = way2[gen1];
        for (int l = gen1; l<29; l++)
        {
            way2[l]=way2[l+1];
        }
        int gen2 = rand()%30;
        for( int l = 29; l>gen2; l--)
        {
            way2[l] = way2[l-1];
        }
        way2[gen2]=memmory;
    }
    summ2+=Distance[0][way2[0]];

    for(int l=0; l<29; l++)
        {
        summ2+=Distance[way2[l]][way2[l+1]];
        }
    summ2+=Distance[0][way2[29]];
    for(int l = 0; l<31; l++)
        {
        City[l]=l+1;
        }

    if(summ2<=summ1)
        {
        for(int d = 0; d<30; d++)
            {
            Child[i][d]=ch2[d];
            }
        fitness[i]=summ2;
        }
    if(summ2>summ1)
        {
        for(int d = 0; d<30; d++)
            {
            Child[i][d]=ch1[d];
            }
        fitness[i]=summ1;
        }
}

for(int i = 0; i<100; i++)
    {
    for(int j= 0; j<30; j++)
```

```
            {
            firstArray[i][j]=Child[i][j];
            }
        }

    float tmp;
    for(int i=0; i<99; i++)
        {
        for(int j=0; j<99; j++)
            {
            if ( fitness[j] > fitness[j+1])
                {
                tmp = fitness[j+1];
                fitness[j+1]= fitness[j];
                fitness[j]= tmp;
                for( int k = 0; k<30; k++)
                    {
                    tmp = firstArray[j+1][k];
                    firstArray[j+1][k]= firstArray[j][k];
                    firstArray[j][k]= tmp;
                    }
                }
            }
        }
    n=30;
    for(int j = 0; j<30; j++)
        {
        int coun= firstArray[0][j];
        while(coun>=n)
            {
            coun-=n;
            }

        way[j]=City[coun];
        for(int k = coun; k < n; k++)
            {
            City[k]=City[k+1];
            }
        n--;
        }

    for(int i = 0; i<31; i++)
        {
        City[i]=i+1;
        }
   fprintf (BEST, "Nomer iter=%d\t", cikle);
   for(int d = 0; d<30; d++)
   { fprintf (BEST, "%f\t",way[d]);}
   fprintf (BEST, "\n");

    n=30;
    for(int j = 0; j<30; j++)
        {
        int coun= firstArray[99][j];
        while(coun>=n)
            {
            coun-=n;
            }

        way[j]=City[coun];
        for(int k = coun; k < n; k++)
            {
            City[k]=City[k+1];
            }
```

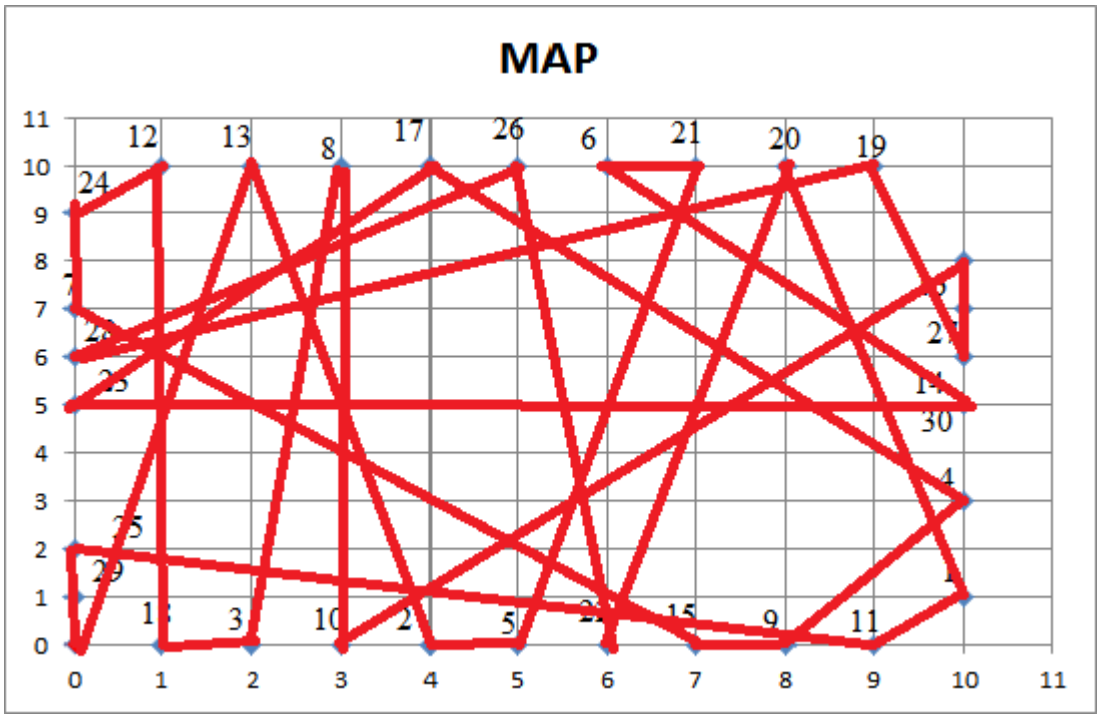
```
            n--;
            }

        for(int i = 0; i<31; i++)
            {
            City[i]=i+1;
            }
         fprintf (WORST, "Nomer iter=%d\t", cikle);
        for(int d = 0; d<30; d++)
        { fprintf (WORST, "%f\t",  way[d]);}
        fprintf (WORST, "\n");

        cout<<fitness[0]<<endl;
        fprintf (B, "Nomer iter=%d\tluch=%f\n", cikle, fitness[0]);
        fprintf (W, "Nomer iter=%d\thud=%f\n", cikle, fitness[99]);
    }

    fclose(B);
    fclose(W);
    fclose(BEST);
    fclose(WORST);
    system("pause");

     return a.exec();
}
```

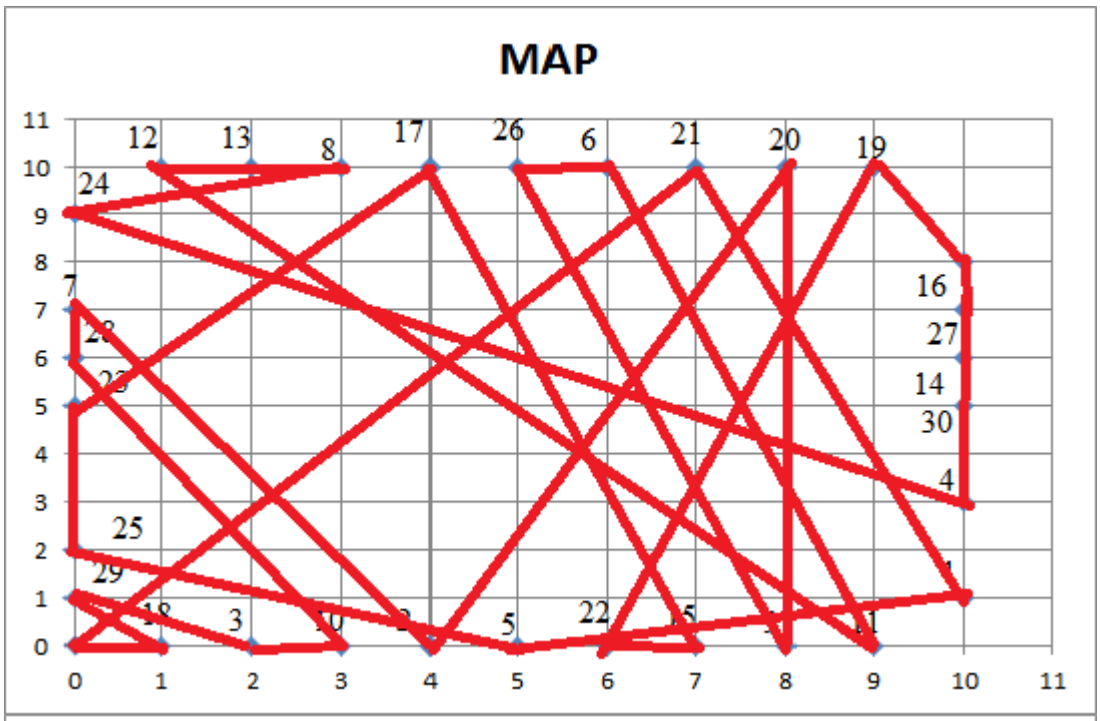
Result

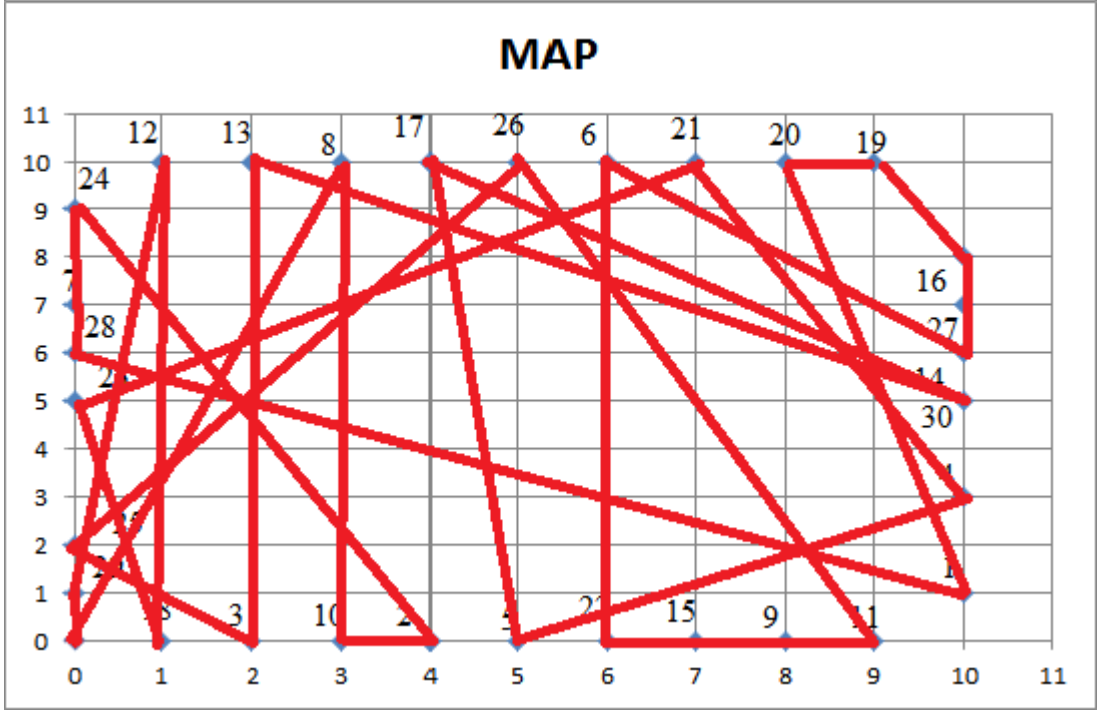


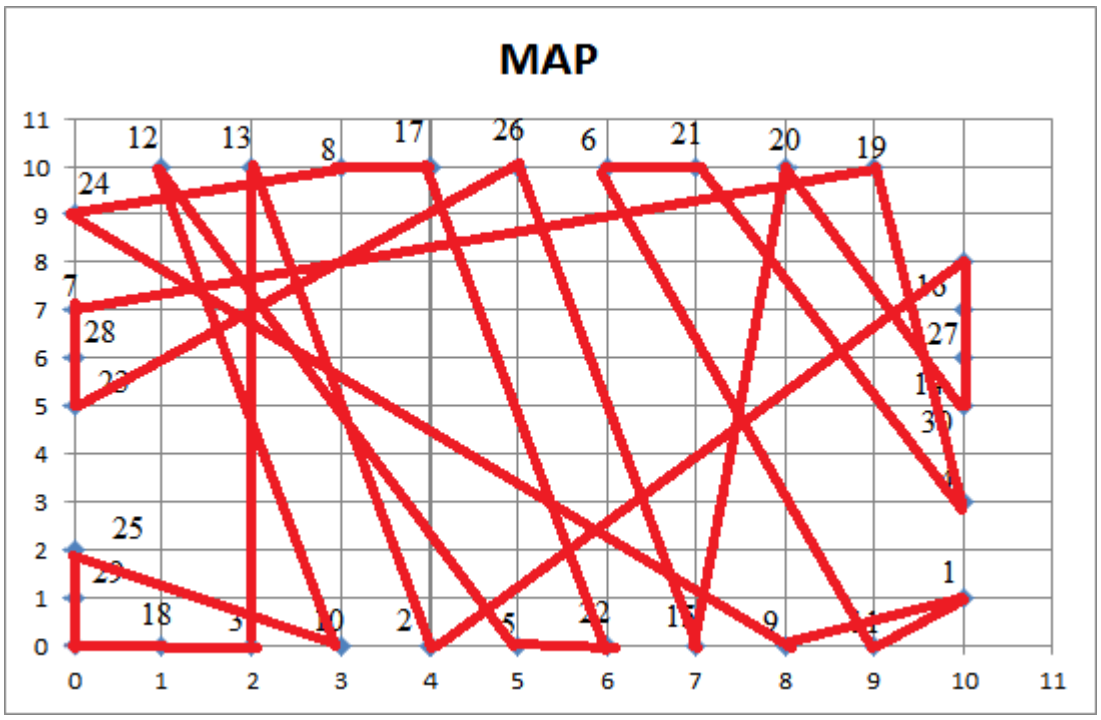


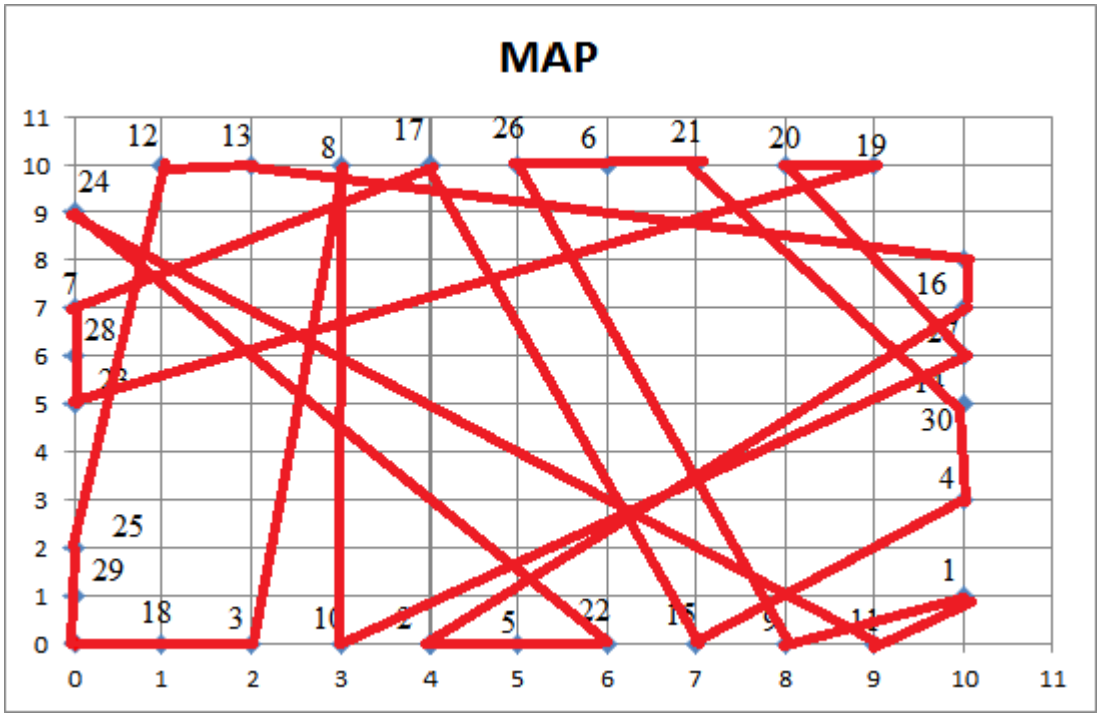
The best distance is 156.3369. Iteration = 76.

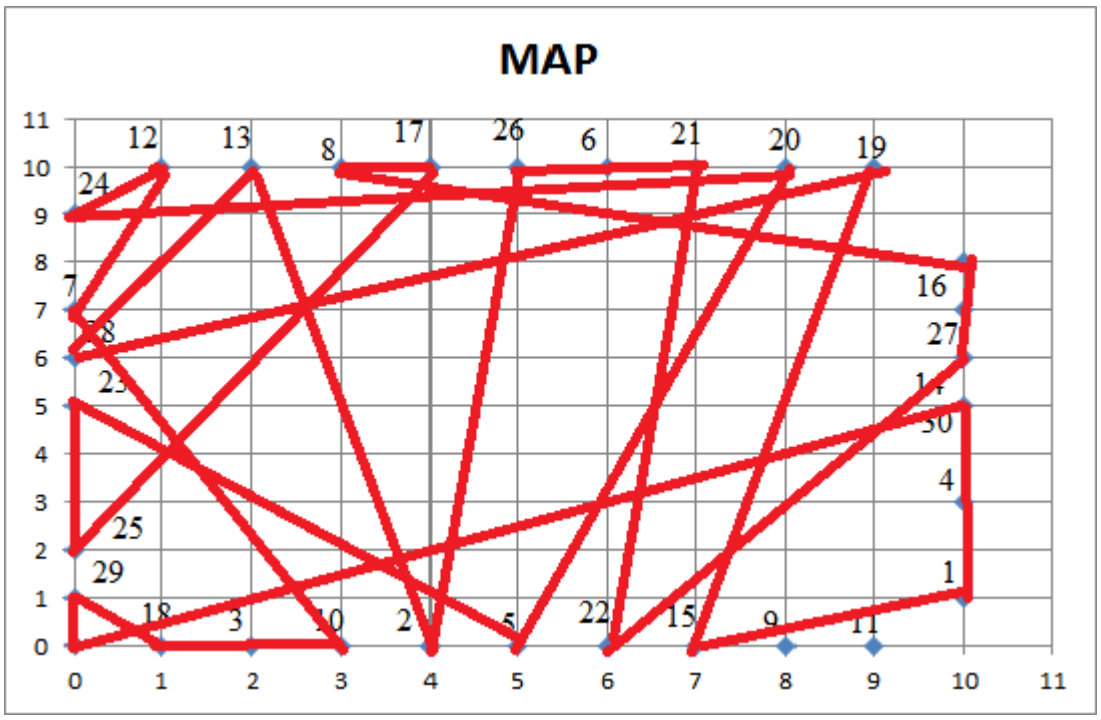


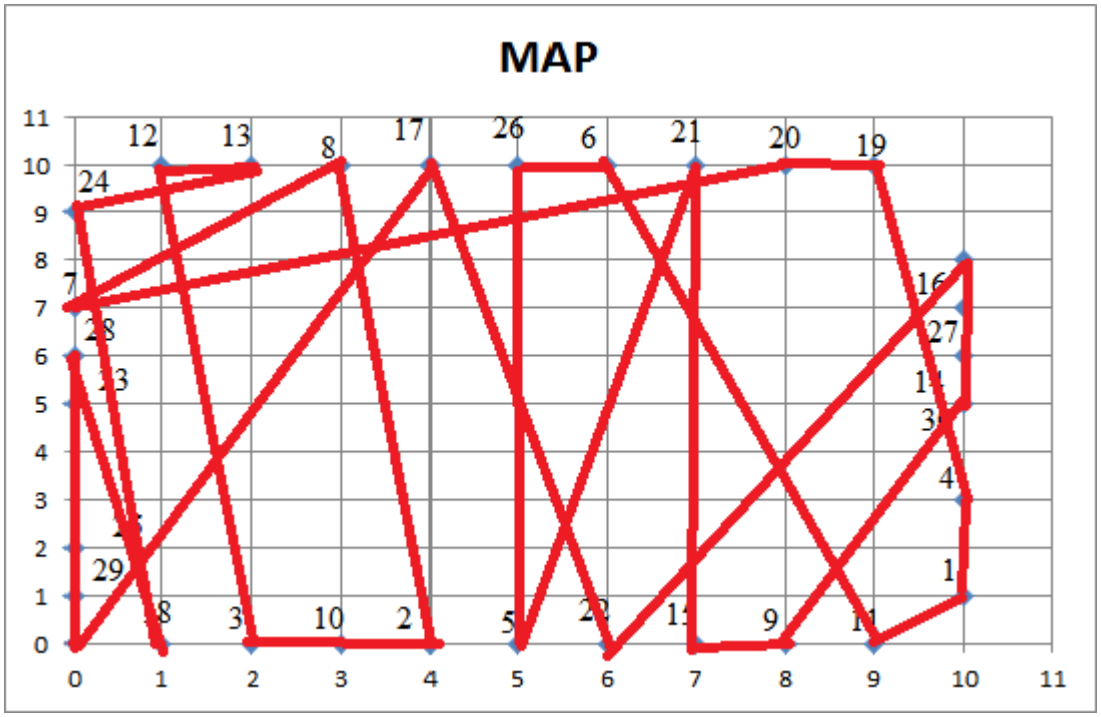

MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
7
16
27
14
30
4
25
29
18 3 10 2 5 22 15 9 11

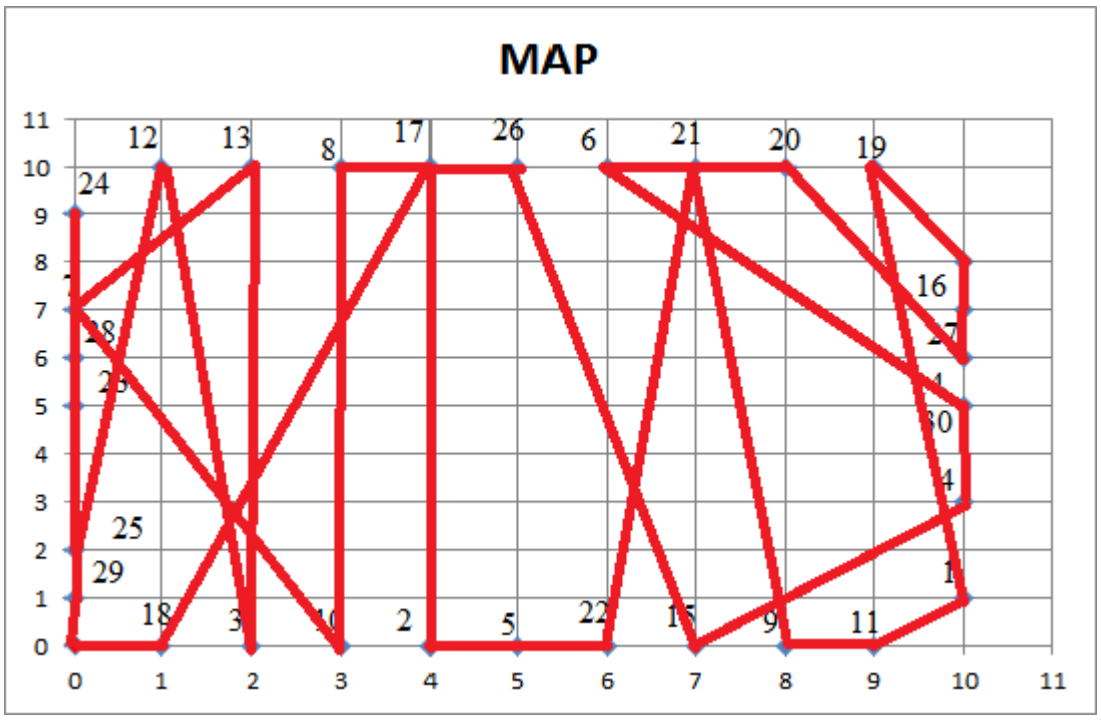

MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
16
27
28
14
30
3 10 15 9 11


MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
7
28
27
30
25
1
18
3
10
2
5
22
9


MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
7
28
16
30
4
25
29
1
18
3
10
2
5
22
9
11


MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
7
16
27
25
29
18 3 10 2 5 22 15 9 11
1
4


MAP
11
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10 11
12 13 8 17 26 6 21 20 19
24
7
16
27
3
4
29
1
3 10 2 5 9


MAP
11
10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
8
17
26
6
21
20
19
24
16
28
25
29
18
2
5
22
11