Modern intelligent IT

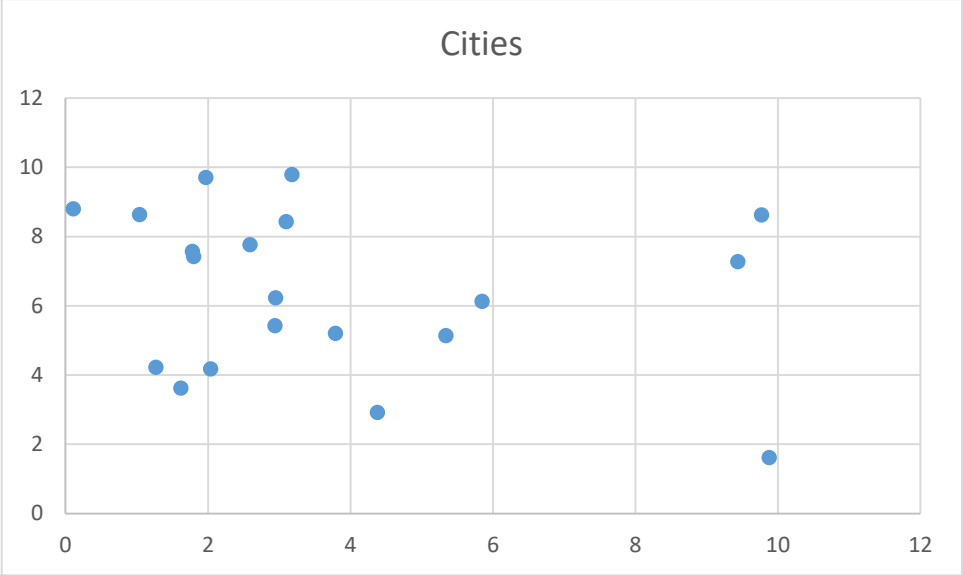
Lab 2 (05.10.2016)

Akira Imada

Student – Malcev Maxim (AI-10)

TSP with 20 cities.

1) I generate 20 cities:



points.txt

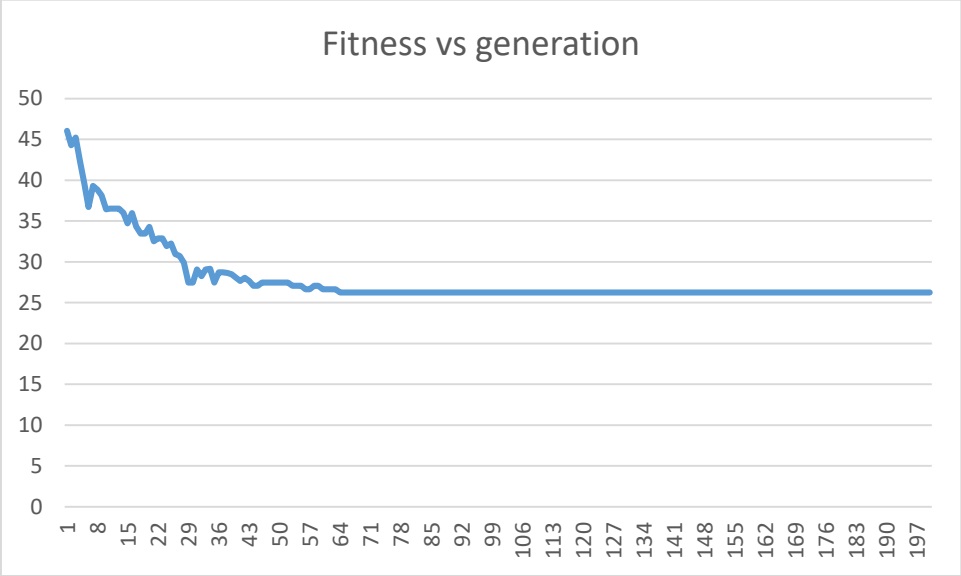
```
2.95 6.23
3.10 8.43
9.77 8.62
9.44 7.27
9.88 1.61
5.85 6.13
2.94 5.42
3.18 9.79
1.04 8.63
5.34 5.14
4.38 2.92
2.04 4.18
0.11 8.80
1.27 4.22
1.78 7.57
1.97 9.71
1.62 3.62
2.59 7.76
1.80 7.42
3.79 5.20
```

2) Calculate the distance matrix:

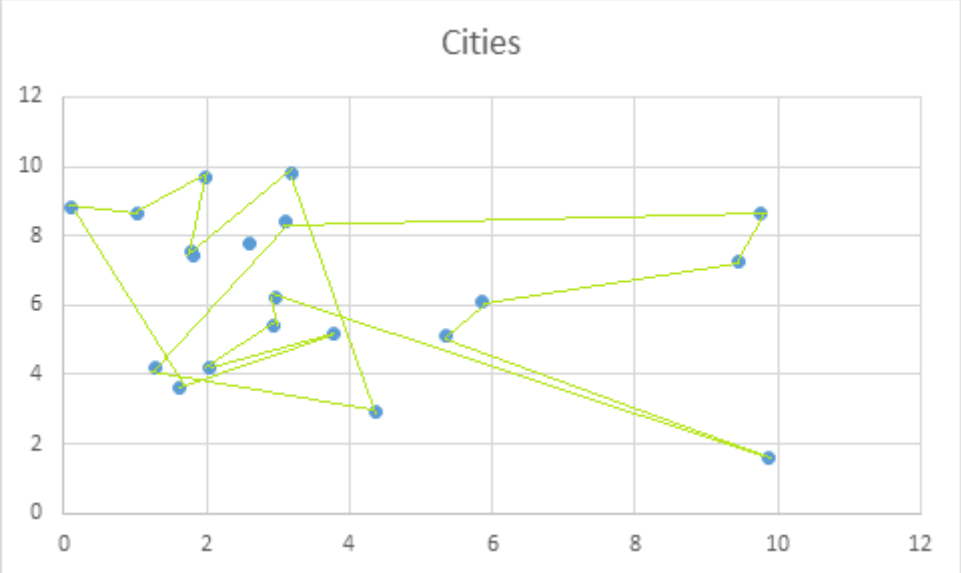
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	0	2.21	7.23	6.57	8.32	2.9	0.81	3.57	3.08	2.62	3.6	2.24	3.83	2.62	1.78	3.62	2.93	1.58	1.66	1.33
2		0	6.67	6.44	9.61	3.58	3.01	1.36	2.07	3.97	5.65	4.38	3.01	4.59	1.58	1.71	5.03	0.84	1.65	3.3
3			0	1.4	7.01	4.65	7.54	6.7	8.73	5.64	7.85	8.92	9.66	9.58	8.06	7.87	9.57	7.23	8.07	6.89
4				0	5.67	3.76	6.75	6.75	8.51	4.62	6.67	8.02	9.45	8.72	7.67	7.85	8.63	6.86	7.64	6.02
5					0	6.06	7.91	10.57	11.29	5.75	5.65	8.25	12.13	9	10.05	11.31	8.5	9.53	9.95	7.06
6						0	2.99	4.53	5.42	1.12	3.54	4.28	6.33	4.97	4.32	5.27	4.92	3.64	4.25	2.26
7							0	4.37	3.74	2.41	2.88	1.53	2.62	2.06	2.44	4.39	2.23	2.37	2.3	0.88
8								0	2.43	5.12	6.97	5.72	4.59	5.89	2.62	1.2	6.36	2.11	2.74	4.63
9									0	5.54	6.62	4.56	9.58	4.42	1.3	1.42	5.05	1.78	1.43	4.4
10										0	2.42	3.43	8.72	4.17	4.3	5.67	4.02	3.79	4.21	1.55
11											0	2.66	9	3.37	5.32	7.2	2.85	5.16	5.19	2.35
12												0	4.97	0.77	3.39	5.52	0.7	3.62	3.24	2.02
13													0	4.73	2.07	2.07	5.4	2.69	2.18	5.15
14														0	3.39	5.53	0.69	3.79	3.24	2.71
15															0	2.15	3.95	0.84	0.15	3.11
16																0	6.1	2.04	2.3	4.86
17																	0	4.26	3.8	2.68
18																		0	0.87	2.83
19																			0	2.98
20																				0

3) I use GA and evolve chromosomes to be the tours of minimum length.

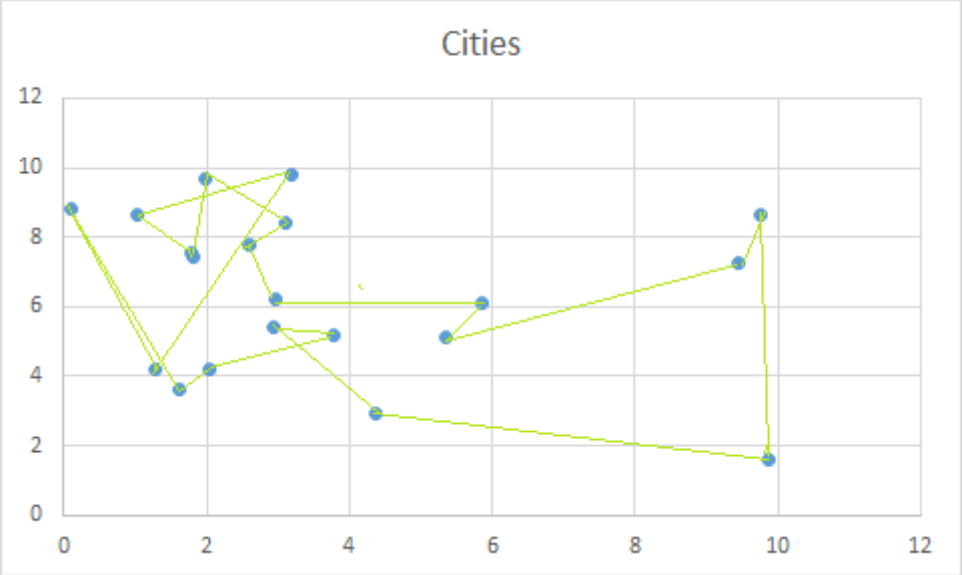
Show the graph of fitness vs generation:



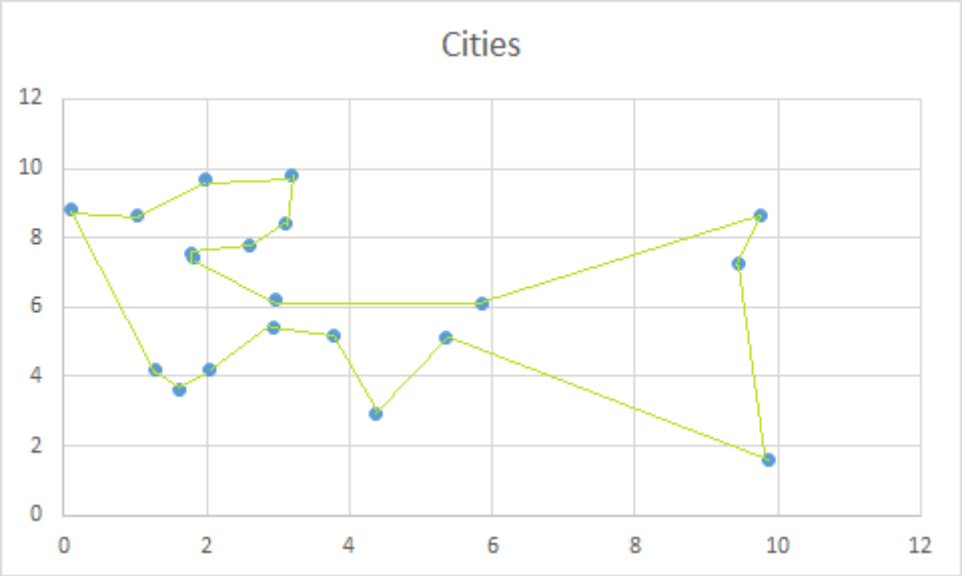
4) Minimum tour in the first generation:



5) Minimum tours in the intermediate generation:



6) Minimum tour in the final generation:



Source code:

```
package com.company;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;


public class Main {

    public static void main(String[] args) {

        Population population = new Population();

        Cities.init();
```

```
        File fFitness = new File("AverageLenghts.txt");
        File fTopFitness = new File("MaximumLenghts.txt");
        try {
            PrintWriter pwFitness = new PrintWriter(fFitness);
            PrintWriter pwTopFitness = new PrintWriter(fTopFitness);
            pwFitness.println(population.getAverageFitness());
            pwTopFitness.println(population.getTopFitness());
            for (int i = 0; i < 100; i++) {
                population.generateNew();
                pwFitness.println(population.getAverageFitness());
                pwTopFitness.println(population.getTopFitness());
            }
            pwFitness.close();
            pwTopFitness.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
public class City {
    private int x, y, id;
    City(int x, int y, int id){
        this.x = x;
        this.y = y;
        this.id = id;
    }

    public int getID(){
        return id;
    }
    double distance(City city){
        return sqrt(pow(city.x - x, 2) + pow(city.y - y, 2));
    }

}public class Population {
    public ArrayList<Chromosome> chromosomes = new ArrayList<>(POPULATION_SIZE);
```

```
    Population(){
        for(int i = 0; i < POPULATION_SIZE; i++){
            chromosomes.add(new Chromosome());
        }
    }

    void generateNew(){

        Collections.sort(chromosomes);

        ArrayList<Chromosome> newChromosomes = new ArrayList<>(POPULATION_SIZE);
        int id1 = 0, id2 = 0;
        Random random = new Random();
        ArrayList<Chromosome> parents = new ArrayList<>();

        for(int i = 0; i < POPULATION_SIZE; i++){
            id1 = random.nextInt((int)(POPULATION_SIZE*PARENTS_PART));
            id2 = random.nextInt((int)(POPULATION_SIZE*PARENTS_PART));
            while(id1 == id2){
                id2 = random.nextInt((int)(POPULATION_SIZE*PARENTS_PART));
            }

            newChromosomes.add(new Chromosome(chromosomes.get(id1),
chromosomes.get(id2)));
        }
        chromosomes = newChromosomes;

    }

    double getAverageFitness(){
        double fitness = 0;
        for(Chromosome chromosome : chromosomes){
            fitness += chromosome.getFitness();
        }
        return fitness/POPULATION_SIZE;
    }
```

```
    double getTopFitness(){
        double fitness = Integer.MAX_VALUE;
        for(Chromosome chromosome : chromosomes){
            if(fitness > chromosome.getFitness()){
                fitness = chromosome.getFitness();
            }
        }
        return fitness;
    }
}
```