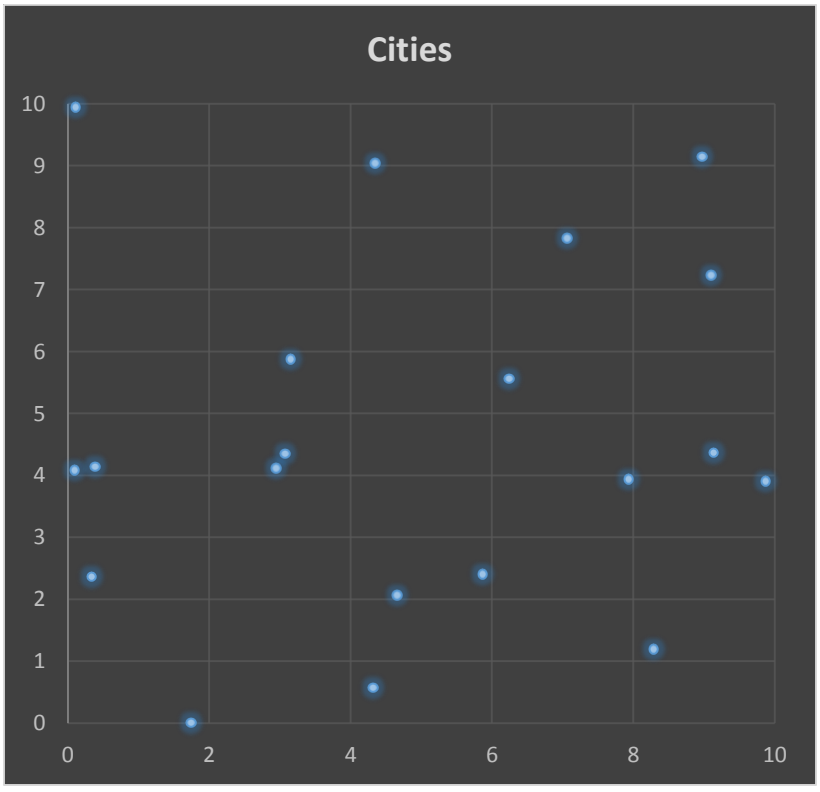
“Traveling Salesperson Problem (TSP)”

1. In first we generate 20 cities with coordinate X and Y.

X	Y
5,870277	2,410462
6,24687	5,561023
3,147018	5,877736
1,739831	0,012608
0,09536	4,086202
0,111207	9,943961
2,945724	4,111366
7,068257	7,833414
4,324227	0,575959
9,870223	3,913385
7,931424	3,942075
8,978373	9,145372
9,136727	4,367255
4,663458	2,067366
4,342821	9,041177
0,336114	2,368949
9,105318	7,233704
8,288854	1,192617
0,388608	4,14668
3,075011	4,351601

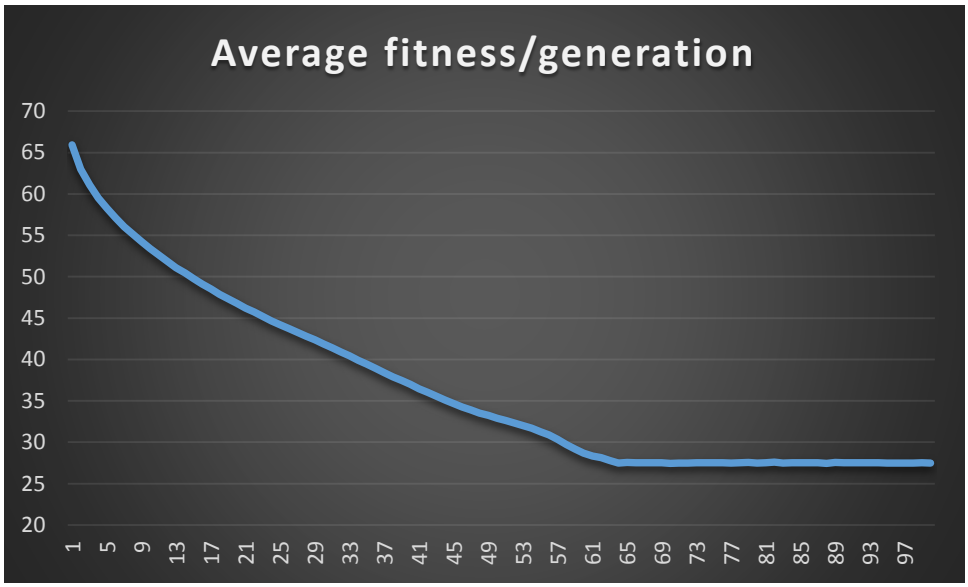
2. Map(0.0 – 10.10) with cities.



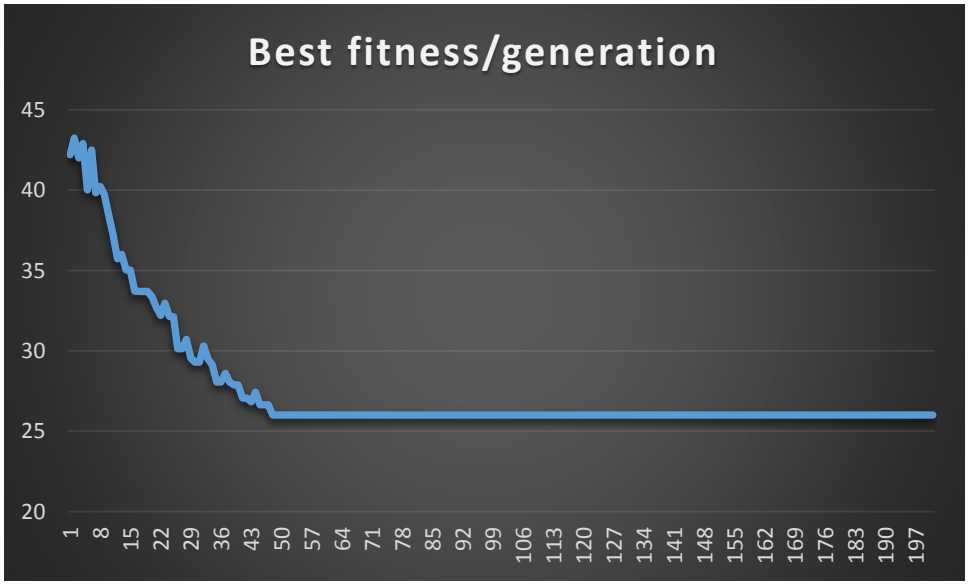
3. Distance Matrix:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	0,00	2,6819	8,8869	5,1515	5,1606	2,0855	6,0197	0,2554	9,9102	8,7459	3,6805	7,3788	8,4900	8,1136	9,2964	3,5548	0,4681	7,4154	5,9827	9,4062
2		0,00	8,8451	7,0992	6,0433	3,7730	9,2934	9,7393	2,3458	5,0769	9,8487	1,9951	4,2618	7,9334	9,9002	7,6726	8,0792	3,1779	3,0885	4,3540
3			0,00	3,6912	8,2145	4,1551	8,3486	3,1170	9,6558	9,6772	8,8580	1,4714	0,6721	0,2183	8,8504	3,2402	7,7407	4,3638	2,5974	6,1966
4				0,00	6,7906	4,4055	5,9693	0,4116	2,1284	8,7551	5,3678	1,1647	7,1613	4,2011	4,5992	5,5348	0,9332	6,7676	8,1082	9,4527
5					0,00	5,9723	2,6871	1,3573	4,0073	0,1572	5,4571	2,4617	1,5937	6,2778	4,4825	7,2922	8,0956	3,5685	2,9050	9,4967
6						0,00	5,2145	5,7038	8,8698	6,0220	5,1439	0,8517	6,7444	4,8553	4,4924	4,8487	01,07,789	0,1352	2,6515	7,7816
7							0,00	9,5870	1,4032	4,3202	2,0945	0,4491	7,6326	1,7478	7,4146	0,0029	9,3579	8,7653	5,5182	6,5507
8								0,00	1,5458	9,9431	4,7701	8,2509	2,8809	3,3885	9,2632	5,2365	0,4898	0,6453	3,3759	6,3721
9									0,00	6,6808	6,4846	6,7078	2,6916	2,5224	1,8482	9,4160	3,2837	4,9904	3,3727	4,5792
10										0,00	9,1508	8,8229	1,7920	4,1519	0,8155	1,3785	5,3456	0,7922	0,9675	1,6234
11											0,00	8,3612	1,5556	5,2327	1,5395	0,8343	9,0884	2,5736	5,6962	1,2351
12												0,00	9,1797	7,3106	2,2058	6,1226	1,3419	3,2069	0,6567	6,2203
13													0,00	4,2031	1,6274	7,3301	6,3388	3,9235	0,0686	7,1606
14														0,00	3,9281	8,5306	5,8337	2,7690	1,3078	0,4043
15															0,00	8,3979	2,9145	4,6182	2,9765	9,0157
16																0,00	1,4798	7,7008	9,6758	2,3416
17																	0,00	4,8042	9,4584	1,7062
18																		0,00	6,4320	7,7789
19																			0,00	6,6036
20																				0,00

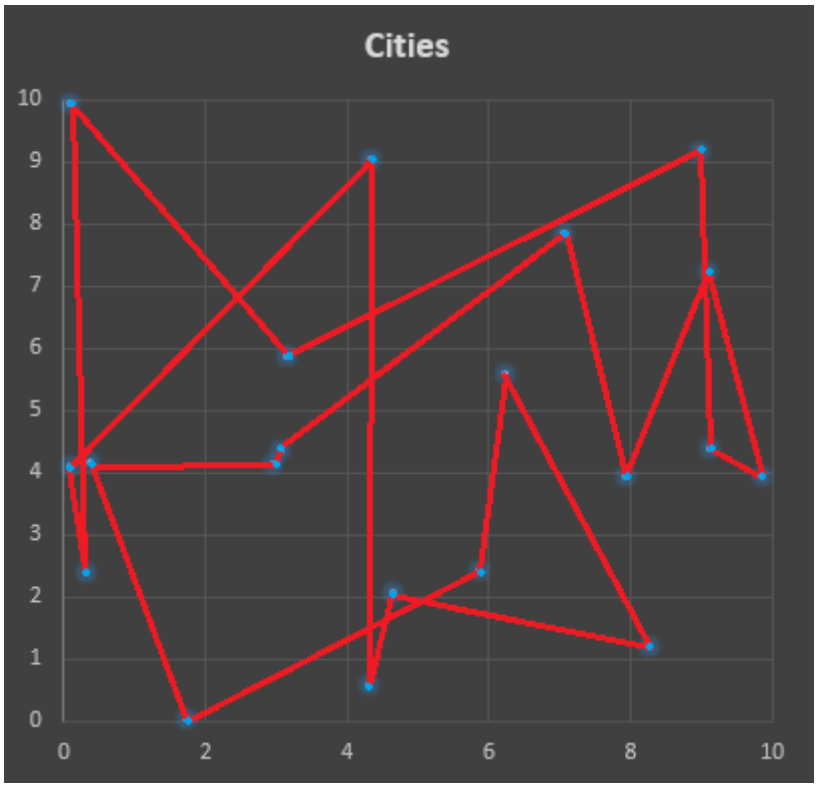
4. Graph average fitness/generation:



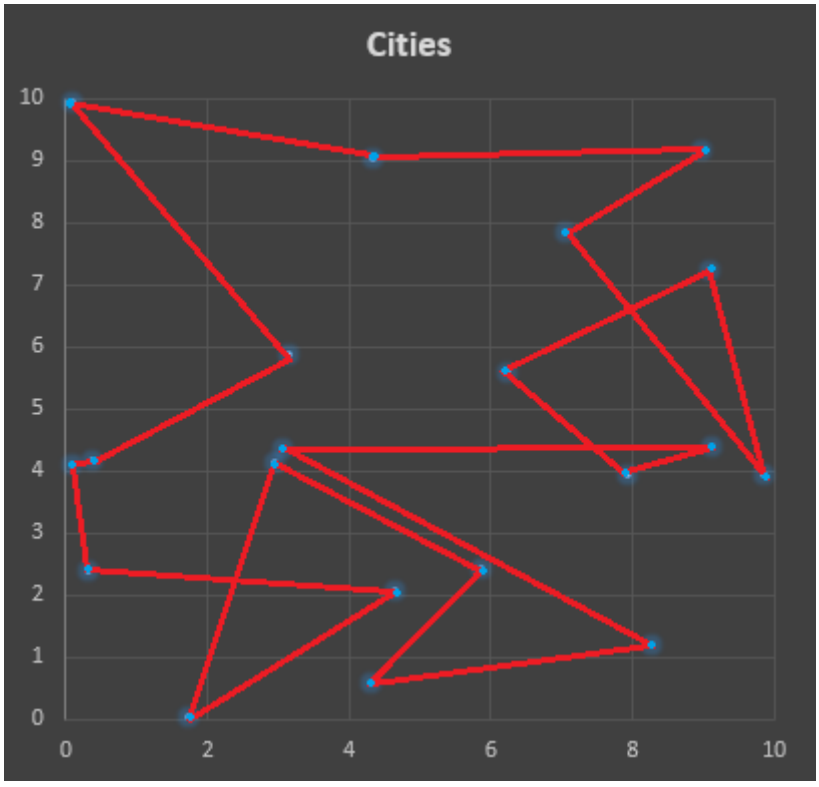
5. Graph best fitness/average:



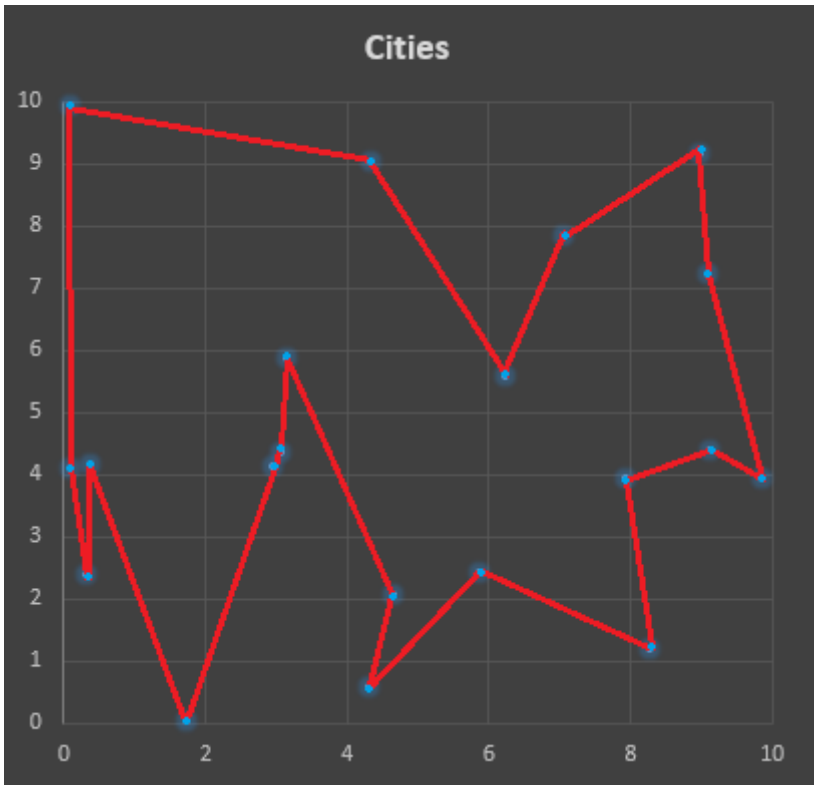
6. Minimum tour in the start generation:



7. Minimum tour in the intermediate generation:



6. Minimum tour in the final generation:



Source code:

Citi.java

```
package com.company;

import java.util.ArrayList;

import static com.company.Constants.*;

public class Cities {
    private static ArrayList<City> cities;


    public static void init(){
        cities = new ArrayList<>(CITIES_AMOUNT);
        int id = 0;
        for(int i = 0; i < HORIZONTAL_CITIES_AMOUNT; i++){
            for(int j = 0; j < VERTICAL_CITIES_AMOUNT; j++){
                cities.add(new City(i, j, id));
                id++;
            }
        }

    }
    public static City getCityByID(int id){
        return  cities.get(id);
    }
    public static ArrayList<City> getCities(){
        return new ArrayList<City>(cities);
    }
}
```

Chromosome.java

```
package com.company;


import java.util.ArrayList;
import java.util.Random;

import static com.company.Constants.*;
import static com.company.Cities.getCityByID;

public class Chromosome implements Comparable{
    public ArrayList<Integer> genes = new ArrayList<>(CITIES_AMOUNT);

    private static Random random = new Random();

    Chromosome(){
        int tmp;
        for(int i = 0; i < CITIES_AMOUNT; i++){

            genes.add(random.nextInt(CITIES_AMOUNT));
        }
    }

    Chromosome(Chromosome par1, Chromosome par2){
       int cutPoint = random.nextInt(CITIES_AMOUNT);


        for(int i = 0; i < cutPoint; i++) {
            genes.add(par1.genes.get(i));
        }

        for(int i = cutPoint; i < CITIES_AMOUNT; i++){
            genes.add(par2.genes.get(i));
        }


        if(random.nextDouble() > 0.85) {
            swapRandomGenes();
        }


/*
        if(random.nextDouble() >= 0){
            int id = random.nextInt(genes.size());
            genes.set(id, (int)(genes.get(id)*random.nextDouble()*2));
        }

        if(random.nextDouble() >= 0){
            int id = random.nextInt(genes.size());
            genes.set(id, random.nextInt(CITIES_AMOUNT));
        }
*/
    }

    private void swapRandomGenes(){
        int id1 = random.nextInt(genes.size());
        int id2 = random.nextInt(genes.size());
        while(id1 == id2){
            id2 = random.nextInt(genes.size());
        }
        int tmp = genes.get(id1);
        genes.set(id1, genes.get(id2));
```

```
    genes.set(id2, tmp);
}

Chromosome(ArrayList<Integer> genes){
    if(genes.size() == CITIES_AMOUNT){
        this.genes = new ArrayList<>(genes);
    } else {
        throw new RuntimeException("Got a wrong amount of genes!");
    }
}

public double getFitness(){
    ArrayList<Integer> path = getPath();
    City city1, city2;
    double fitness = 0;
    for(int i = 0; i < CITIES_AMOUNT - 1; i++){
        city1 = getCityByID(path.get(i));
        city2 = getCityByID(path.get(i + 1));
        fitness += city1.distance(city2);
    }
    city1 = getCityByID(path.get(0));
    city2 = getCityByID(path.get(path.size() - 1));
    fitness += city1.distance(city2);
    return fitness;
}


public ArrayList<Integer> getPath(){
    ArrayList<Integer> path = new ArrayList<>(CITIES_AMOUNT);
    ArrayList<City> citiesPool;
    citiesPool = Cities.getCities();
    path.add(12);
    citiesPool.remove(12);
    int tmp = 0, i = 0;
    while(citiesPool.size() > 0){
        tmp = genes.get(i);
        while(tmp >= citiesPool.size()){
            tmp -= citiesPool.size();
        }
        path.add(citiesPool.get(tmp).getID());
        citiesPool.remove(tmp);
        i++;
    }

    return path;
}

@Override
public int compareTo(Object chromosome) {
    double compareFitness=((Chromosome)chromosome).getFitness();
    if(getFitness() > compareFitness) {
        return 1;
    } else if(getFitness() == compareFitness){
        return 0;
    } else {
        return -1;
    }
}
```