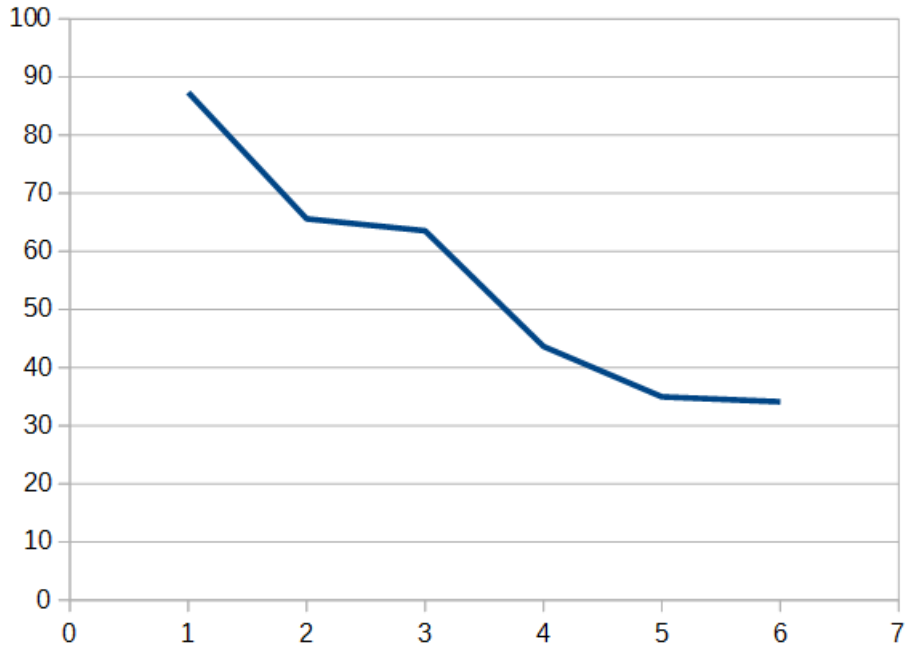
```cpp
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <ctime>
#include <utility>
#include <cmath>
#include <fstream>

using namespace std;

vector<vector<int>> evolution(vector<vector<int>> gen, unsigned number_of_chr, unsigned
number_of_gen, unsigned number_of_par);
void mutation(vector<vector<int>> &generation, unsigned procent);
void vector_output(vector<vector<int>> vec);
double fitness(vector<int> chromosome);
double distance(int point1, int point2);
vector<int> desh(vector<int> vec);


int main()
{
	srand(time(0));
	vector<vector<vector<int>>> library;
	ofstream fout, stat;
	fout.open("D:\\test\\test.txt");
	fout.close();
	fout.open("D:\\test\\test.txt");
	stat.open("D:\\test\\test2.txt");

	int size = 7; //размер хромосомы
	int num = 10; //количество хромосом в поколении
	int par = 4; //количество родителей
	int aspects = 5; //количество видов

	vector<vector<int>> generation;
	for (int i = 0; i < num; i++) {
		vector<int> temp;
		for (int j = 0; j < size; j++) {
			temp.push_back(rand() % 8);
		}
		generation.push_back(temp);
	}
	library.push_back(generation);
	cout << "generation 0:" << endl;
	vector_output(generation);
	for (int i = 0; i < generation.size(); i++) {
		for (int j = 0; j < generation[i].size(); j++) {
			stat << generation[i][j];
		}
		stat << "     ";
		stat << fitness(generation[i]) << "      ";
	}
	stat << '\n' << '\n';
	int score = 1;
	vector<double> lib;
	while (true) {
		generation = evolution(generation, num, size, par);
		mutation(generation, 5);
		score++;
		cout << "generation " << score << ":" << endl;
		vector_output(generation);
		//if (score == 1000) break;
		bool check = 0;
		double min = 1000;
		int P;
```

```
		for (int i = 0; i < generation.size(); i++) {
			if (fitness(generation[i]) < min) {
				min = fitness(generation[i]);
				P = i;
			}
		}
		fout << score - 1 << "	" << min << '\n';
		cout << min << endl;
		lib.push_back(min);
		int last = lib.size() - 1;
		if (score >= 6) {
			if (lib[last] == lib[last - 1] == lib[last - 2] == lib[last - 3] ==
lib[last - 4]) {
				check = 1;
			}
		}
		if (score == 400) {
			for (int i = 0; i < generation.size(); i++) {
				for (int j = 0; j < generation[i].size(); j++) {
					stat << generation[i][j];
				}
				stat << "	";
				stat << fitness(generation[i]) << "	";
			}
			stat << '\n' << '\n';
		}
		if (check == 1 || score == 10000) {
			for (int i = 0; i < generation.size(); i++) {
				for (int j = 0; j < generation[i].size(); j++) {
					stat << generation[i][j];
				}
				stat << "	";
				stat << fitness(generation[i]) << "	";
			}
			break;
		}
	}
	return 0;
}


vector<vector<int>> evolution(vector<vector<int>> gen, unsigned number_of_chr, unsigned
number_of_gen, unsigned number_of_par) {
	vector<vector<int>> par;
	vector<vector<int>> gen1 = gen;
	vector<double> mark;
	for (int i = 0; i < gen.size(); i++) {
		mark.push_back(fitness(gen[i]));
	}
	for (int i = 0; i < number_of_par; i++) {
		int min = 1000;
		int point = 0;
		for (int j = 0; j < gen1.size(); j++) {
			if (mark[j] < min) {
				min = mark[j];
				point = j;
			}
		}
		par.push_back(gen1[point]);
		gen1[point] = gen1[gen1.size() - 1];
		gen1.pop_back();
	}
	vector<vector<int>> result;
	for (int i = 0; i < number_of_chr; i++) {
		unsigned mother = rand() % par.size();
```

```
			unsigned father;
			while (true) {
				father = rand() % par.size();
				if (mother != father) break;
			}
			unsigned point = rand() % number_of_gen;
			vector<int> child;
			for (int j = 0; j < number_of_gen; j++) {
				if (j <= point) child.push_back(par[mother][j]);
				else child.push_back(par[father][j]);
			}
			result.push_back(child);
		}
		return result;
}

void mutation(vector<vector<int>> &generation, unsigned procent) {
		for (int i = 0; i < generation.size(); i++) {
			if (rand() % 100 + 1 >= procent) {
				int j = rand() % generation[i].size();
				generation[i][j] = rand() % 8;
			}
		}
}

void vector_output(vector<vector<int>> vec) {
		for (int i = 0; i < vec.size(); i++) {
			vector<int> vec2 = desh(vec[i]);
			for (int j = 0; j < vec2.size(); j++) {
				cout << vec2[j];
			}
			cout << "   " << fitness(vec[i]) << endl;;
		}
		cout << endl;
}

double fitness(vector<int> chromosome) {
		double result = 0;
		vector<int> temp;
		temp.push_back(1);
		temp.push_back(2);
		temp.push_back(3);
		temp.push_back(4);
		temp.push_back(5);
		temp.push_back(6);
		temp.push_back(7);
		int last_point = 0;
		for (int i = 0; i < chromosome.size(); i++) {
			int k = 0;
			for (int j = 0; j < chromosome[i]; j++) {
				k++;
				if (k >= temp.size()) k = 0;
			}
			result += distance(last_point, temp[k]);
			last_point = temp[k];
			for (int j = k; j < temp.size() - 1; j++) {
				temp[j] = temp[j + 1];
			}
			temp.pop_back();
			if (i == chromosome.size() - 1) result += distance(last_point, 0);
		}
		return result;
}

double distance(int point1, int point2) {
```

```
pair<double, double> coord1, coord2;
if (point1 == 1) {
	coord1.first = 0;
	coord1.second = 2.5;
}
if (point1 == 2) {
	coord1.first = 0;
	coord1.second = 7.5;
}
if (point1 == 3) {
	coord1.first = 2.5;
	coord1.second = 10;
}
if (point1 == 4) {
	coord1.first = 7.5;
	coord1.second = 10;
}
if (point1 == 5) {
	coord1.first = 7.5;
	coord1.second = 0;
}
if (point1 == 6) {
	coord1.first = 2.5;
	coord1.second = 0;
}
if (point1 == 7) {
	coord1.first = 10;
	coord1.second = 7.5;
}
if (point1 == 0) {
	coord1.first = 10;
	coord1.second = 2.5;
}
if (point2 == 1) {
	coord1.first = 0;
	coord1.second = 2.5;
}
if (point2 == 2) {
	coord1.first = 0;
	coord1.second = 7.5;
}
if (point2 == 3) {
	coord1.first = 2.5;
	coord1.second = 10;
}
if (point2 == 4) {
	coord1.first = 7.5;
	coord1.second = 10;
}
if (point2 == 5) {
	coord1.first = 7.5;
	coord1.second = 0;
}
if (point2 == 6) {
	coord1.first = 2.5;
	coord1.second = 0;
}
if (point2 == 7) {
	coord1.first = 10;
	coord1.second = 7.5;
}
if (point2 == 0) {
	coord1.first = 10;
	coord1.second = 2.5;
}
```

```
	double result;
	result = sqrt(pow(abs(coord1.first - coord2.first), 2) + pow(abs(coord1.second -
coord2.second), 2));
	return result;
}

vector<int> desh(vector<int> vec) {
	vector<int> temp;
	vector<int> res;
	temp.push_back(1);
	temp.push_back(2);
	temp.push_back(3);
	temp.push_back(4);
	temp.push_back(5);
	temp.push_back(6);
	temp.push_back(7);
	res.push_back(0);

	for (int i = 0; i < vec.size(); i++) {
		int k = 0;
		for (int j = 0; j < vec[i]; j++) {
			k++;
			if (k >= temp.size()) k = 0;
		}
		res.push_back(temp[k]);
		for (int j = k; j < temp.size() - 1; j++) {
			temp[j] = temp[j + 1];
		}
		temp.pop_back();
	}
	res.push_back(0);
	return res;
}
```



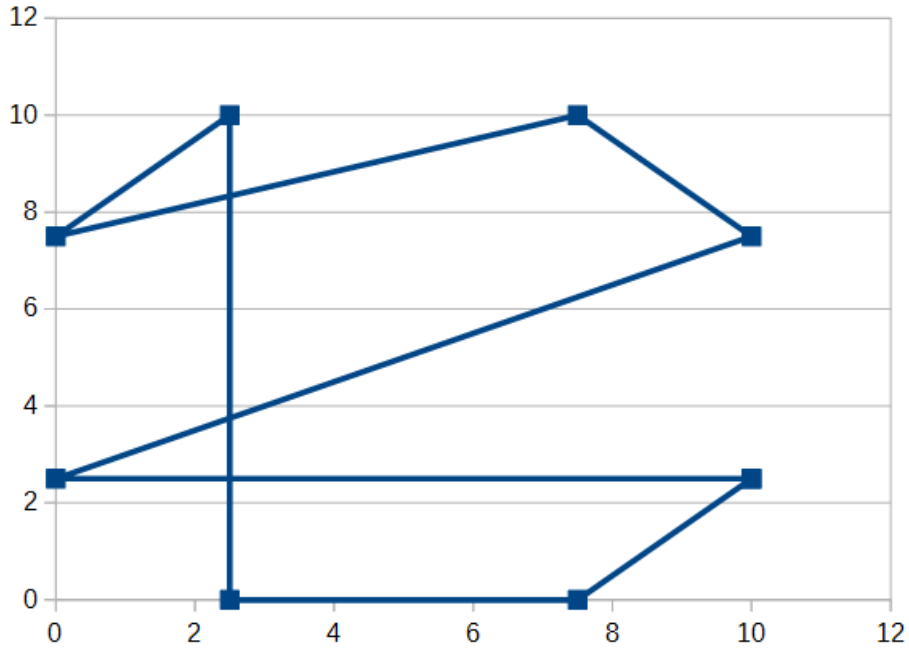
X is generation

Y is fitness (our distance between cities)

Statistic from 1st generation:

way	fitness
0300422	118
2477143	96
1356570	143
4512677	124
1344673	117
3326773	97
6077300	105
5662433	104
7713474	125
7725123	99

Example of way



Statisctic from 4th generation

4017335	65
4021335	58
7616730	46
7611730	41
1755330	49
6057770	52
7661730	71
1355030	47
1357275	59
4011355	65

Statistic from last generation

5511111	32
3201042	45
0307041	62
3306260	37
3206042	41
5306442	45
0306265	36
3206472	39
0106041	45
0406041	40

Example of way