# INFO 6200 Final Project

## Traveling Salesman Problem Based on Genetic algorithm

Team #232
Weikai Chen - 001499376
Shaodong Wang - 001441616
Hongbo Jiang  - 001440416

December 1, 2018

## Introduction

TSP problem
The Traveling Salesman Problem is one of the famous problems in the field of mathematics. Suppose there is a traveler who wants to visit *n* cities and the route is that each city can only visit once. The selection goal of the path is to get the path that will take least time to all cities.

Genetic Algorithm
Genetic algorithm is a widely applied and efficient random search and optimization method based on the principle of biological evolution theory. Its main feature is the group search strategy and the exchange of information between individuals in the group. This algorithm was not designed to solve the optimization problem. It together with the evolution strategy and evolutionary planning constitute the main framework of the evolutionary algorithm, so genetic algorithms are the most widely known algorithms in evolutionary algorithms. The result may not be the best solution, but this is a more easier way to get it.

## What we do

In our project, firstly, we randomly set the coordinate of *n* cities in X-Y coordinate system and calculate the distance between each 2 cities. Each city will be count from 0-*n*, which are *n* genes, in a specific code way. One route, for example, the sales man start from number 0 to number *n*, is a chromosome. The number of all routes are the scale of the species, which is set as 1,000. Put them in a two-dimensional array. In each time of the evolution which includes crossover and mutation, we select the best route and another route to the next generation. After generations we get the best route. Below are the details of the implement.

## Implement Details

1. The gene is the count number of a city. We set a number of 1,000 and use Random() method to return an integer *i* from 0-999. Then we get the value of $i \% n$ (*n* is the total number of all cities), which means we get a number from 0 to n-1.
2. The gene expression is the distance of each city. We want the total distance of the best route is least one among all routes.
3. We set the the reciprocal of the distance as the index of fitness.
4. We calculate the fitness of each chromosome, which represents the total distance of each route and of course we select the best fit chromosome, which means the route with the shortest distance to go through all cities. Besides, we use RouletteSelect method, which basically gives us a randomly selected array for the next generation.
5. For the evolution part, there are 2 methods: Crossover and Mutation. The chance for 2 chromosomes to take a crossover action or 1 chromosome mutates is 50%. As to the method of selecting which chromosome change in each generation, we use Random() method to decide which two take a crossover action and which one to mutates.
6. There are 2 two-dimensional arrays prepared for the whole population. We name them Parents[][] and Childs[][]. For each generation, Parents generate Childs, and Childs become Parents to generate new Childs in the next evolution.
7. For the unit test, we find data on the Internet which all cities are set and the best route has been calculated.

## Other Details

For the distance, we use Euclidean Distance to represent the distance between two cities. The distance between each two cities is put in a two-dimensional array: distance[][].
For the crossover action, we randomly get 2 chromosomes and randomly exchange the genes of a specific position in these 2 chromosomes.
In the whole process, there are several times to use the Random() method to get a number in a specific scale. Thus, we set 65,535 which is the biggest number of type Int as the scale.

## About Test

GA Solve Test:
Using GA algorithm to solve the data which is selected from the TSPlib http://www.math.uwaterloo.ca/tsp/world/countries.html#DJ whose optimal path and distance are given. Compare the optimal distance and the solved distance.

CrossoverTest:
Checking the possibility of crossover happening in GA responding to the factor Pc which can control the Crossover in GA method.

MutationTest:
Checking the possibility of mutation happening in GA responding to the factor Pm which can control the mutation in GA method.

CalDistanceTest:
The purpose is to test the gene expression in our method. Test the correction of CalDistance method output with the given input and optimal output.

InitialTest:
InitGroup method can generate the first parent population with the cityNum and scale. Test that each individual can fit the regulation.

PositionToDistanceTest:
Assert the result of PositionToDistance with the optimal distance matrix.

CountRateTest:
Assert the result of countRate method with the optimal possibility after calculation.

RouletteSelectTest:
Assert that after RouletteSelect method, all the new population are copy from the parent population by some regulations.

SelectBestTest:
Assert that the SelectBest method can select the individual gene with the best fitness and populate it to the new population.

## Conclusion

With all tests pass, our program is proved to solve such problem. Due to the cities are all randomly generated in the main class, this program can only provide a best route that guarantees the total distance of this route is the minus one.

The screenshots of test pass and program complete run is attached in the repo.