Intro to Artificial Intelligence
Assignment 2 Report
SOFE3720U

Group 3

Name	Student ID
Chen Pi	100567388
Ayesha Farkhundah	100517461
Jasindan Rasalingam	100584595

Professor: Dr. Sukhwant Kaur Sagar

Submitted to the University of Ontario Institute of Technology
Winter 2019

1. Github link of code.
   https://github.com/AyeshaFarkhundah/A.i_Assign2/

2. Code.

City.java

```
package
Problem;


                public class City {


                        String name;
                        int x,y;



                        public City (String name, int x, int y) {
                                this.name= name;
                                this.x=x;
                                this.y=y;
                        }


                        public String getName() {
                                return this.name;
                        }
                        public int getx() {
                                return this.x;
                        }


                        public int gety() {
                                return this.y;
```

```
		}

		public double distance (City nextC) {
			int xVal=getx()-nextC.getx();
			int yVal=gety()-nextC.gety();
			double
	answer=Math.sqrt((xVal*xVal)+(yVal*yVal));
			return answer;

		}
	}
```

Final.java

```
package
Problem;

public class Final {

	public static void main() {
		//int cityMatrix[19][19];
		//creating a test city Matrix but can also accept these
values from user

		//distance relative
		int testCityMatrix[][]={
				{0,172,145,607,329,72,312,120},
				{172,0,192,494,209,158,216,93},
				{145,192,0,490,237,75,205,100},
				{607,494,490,0,286,545,296,489},
```

```
                                {329,209,237,286,0,421,49,208},
                                {72,158,75,545,421,0,249,0,194},
                                {120,92,100,489,208,75,194,0}
                        };

                String cityName[]= {"Brighton", "Briston","Cambridge",
                                        "Glasgow", "Liverpool",
"London",
                                        "Manchester", "Oxford"};

                //use relative distance to get absolute distance

                //City aCity = new City(cityName[0],0,0 );
                //City bCity = new City(cityName[1],0,0 );
                //City cCity = new City(cityName[2],0,0 );
                //City dCity = new City(cityName[3],0,0 );
                //City eCity = new City(cityName[4],0,0 );
                //City fCity = new City(cityName[5],0,0 );
                //City gCity = new City(cityName[6],0,0 );
                //City hCity = new City(cityName[7],0,0 );
        }
}
```

Map.java

```
package
Problem;

import java.util.ArrayList;

public class Map {
```

```
    //for 20 cities:
//public ArrayList Mapp= new ArrayList<City>(20);
    //for test of 8 cities
    public ArrayList MapList= new ArrayList<City>(7);
    int index=0;
    double fitness=0;
    double distance=0;


    public Map(ArrayList MapList) {
           this.MapList=MapList;
    }

    public void setCity(City placeCity,int i) {
    MapList.set(i, placeCity);
    fitness=0;
    distance=0;
    }

    public City getCity(int i) {
           return (City)MapList.get(i);
    }

    public void setFitness(double fitness) {
           this.fitness=fitness;
    }

    public double getFitness(int i) {
        if(fitness==0) {
```

```
            fitness= 1/TotalDist();
        }
        return fitness;
    }

    public double TotalDist() {
        double total =0;
        if (distance==0) {
            //for tet 8 cities
            for (int j=0; j<6;j++) {
                City placeCity=getCity(j);
                City toCity= getCity(j+1);
                total = total+placeCity.distance(toCity);

            }
            //for 20 cities
            //for (int i; i<20;i++) {  }

        }

        return total;
    }
}
```

MapRoute.java

```
package
Problem;

import java.util.ArrayList;
```

```
public class MapRoute {


	public ArrayList cityList = new
ArrayList<City>();
	public int size =20;


	public void setCity (City newCity) {
		//adds to the end
		cityList.add(newCity);
	}



	public City getCity(int i ) {
		return (City)cityList.get(i);
	}


}
```

Population.java

```
package
Problem;


public class
Population {


}
```

The components of the Genetic Algorithm was:

Initialization: Population is hardcoded, and distances are placed in a matrix data structure.

Selection: Fitness values are not really improving, just using total distance to determine a cities fitness value.

Genetic Operators:

Heuristics: Cities are visited in order, and the heuristic is the current city the traveller is on plus the distance to the city the traveller is going to.

Termination: Algorithm will terminate when each city has been visited.

3. **Output**

Termination Condition: When each city is visited (cities cannot be visited twice), the algorithm will terminate.

Reference:

https://books.google.ca/books?id=Swlcw7M4uD8C&pg=PA185&lpg=PA185&dq=traveling+salesman+genetic+algorithm+relative+distance+java&source=bl&ots=TKy6NbhYB7&sig=ACfU3U1YfN1XBJRpkamNSbi4FASSK2aZ3Q&hl=en&sa=X&ved=2ahUKEwie9p-c_JbhAhWSyIMKHSTZAmEQ6AEwCHoECAkQAQ#v=onepage&q=traveling%20salesman%20genetic%20algorithm%20relative%20distance%20java&f=false

http://www2.sys-con.com/ITSG/virtualcd/Java/archives/0603/lacy/index.html

http://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5
http://www.theprojectspot.com/tutorial-post/creating-a-genetic-algorithm-for-beginners/3
http://www.zaneacademy.com/