Assignment No. 5

Aim: Implement Travelling Salesman problem using Genetic Algorithm.

Theory:

The Travelling Salesman Problem is an important real-life problem. In this, we have to find out the best route to be taken by a salesman, so that he can travel to all the destinations by the shortest possible distance (or cost). This finds a real-life application in delivery systems like that in Amazon, Flipkart, Zomato and Swiggy for example and also can be used in car-pooling applications such as Uber to find the shortest route.

The original algorithm used to find the best solution is very time-consuming. If we implement it, then the program may even take days or weeks to find the best possible answer to our problem. So, by using Artificial Intelligence, we can improve the speed of the program such that a program that would have taken days or weeks normally can be solved within a few minutes using the special technique of Genetic Algorithms. Although this does not guarantee the best solution, it always gives a solution which is very close to the best solution, but the speed efficiency is really great, so this is used as a substitute to the original algorithm to achieve faster performance.

For this program, we are given an input distance map which represents the cost of travelling from one city to the other.

Currently I have set the following as the input distance map. We can easily modify the code to let the user input the distance map, by taking input as we do in a matrix.

```
vector distance_map = {
{INF, 10, 3, 2, 5, 6, 7, 2, 5, 4},
{20, INF, 3, 5, 10, 2, 8, 1, 15, 6},
{10, 5, INF, 7, 8, 3, 11, 12, 3, 2},
{3, 4, 5, INF, 6, 4, 10, 6, 1, 8},
{1, 2, 3, 4, INF, 5, 10, 20, 11, 2},
{8, 5, 3, 10, 2, INF, 6, 9, 20, 1},
{3, 8, 5, 2, 20, 21, INF, 3, 5, 6},
{5, 2, 1, 25, 15, 10, 6, INF, 8, 1},
{10, 11, 6, 8, 3, 4, 2, 15, INF, 1},
{5, 10, 6, 4, 15, 1, 3, 5, 2, INF}
};
```

NOTE: INF represents a very large number. It means if we go to that position, it cannot lead to an optimal path.

Code:

```
#include <bits/stdc++.h>
using namespace std;
#define CITIES 10
#define GENES "ABCDEFGHIJ"
#define START 0
#define POPULATION_SIZE 15
#define INF 99999
struct Individual {
	string route;
	int distance;
};
int rand_num(int start, int end) {
	return start + rand() % (end - start);
}
bool repeat(string s, char ch) {
	for (char &c: s) {
		if (c == ch) {
			return true;
		}
	}
	return false;
}
string mutatedGene(string route) {
	while (true) {
		int r = rand_num(1, CITIES);
		int r1 = rand_num(1, CITIES);
		if (r1 != r) {
			char temp = route[r];
			route[r] = route[r1];
			route[r1] = temp;
			break;
		}
	}
	return route;
}
string createRoute() {
	string result = "0";
	while (true) {
		if (result.size() == CITIES) {
```

```
                result += result[0];
                break;
            }
            char temp = rand_num(1, CITIES) + '0';
            if (!repeat(result, temp))
                result += temp;
        }
        return result;
}
int calculateDistance(string route, vector<vector<int>> &map) {
        int distance = 0;
        for (int i = 0; i < route.size() - 1; i++) {
            if (map[route[i] - '0'][route[i + 1] - '0'] == INF)
                return INF;
            distance += map[route[i] - '0'][route[i + 1] - '0'];
        }
        return distance;
}
int coolDown(int temp) {
        return (95 * temp) / 100;
}

bool comp(Individual &t1, Individual &t2) {
        return t1.distance < t2.distance;
}
void TravellingSalesman(vector<vector<int>> &map) {
        int gen = 1;
        int genThreshold = 4;
        vector<Individual> population;
        Individual temp;
        for (int i = 0; i < POPULATION_SIZE; i++) {
            temp.route = createRoute();
            temp.distance = calculateDistance(temp.route, map);
            population.push_back(temp);
        }
        cout << "\n Initial Population: " << endl;
        cout << " ROUTE\t    DISTANCE" << endl;
        for (int i = 0; i < POPULATION_SIZE; i++) {
            cout << " " << population[i].route << " " << population[i].distance << endl;
        }
```

```
        cout << endl;
        bool found = false;
        int temperature = 10000;
        while (temperature > 1000 and gen <= genThreshold) {
                sort(population.begin(), population.end(), comp);
                cout << "\n Current Temp: " << temperature << endl;
                vector<Individual> new_population;
                for (int i = 0; i < POPULATION_SIZE; i++) {
                        Individual p1 = population[i];
                        while (true) {
                                string new_g = mutatedGene(p1.route);
                                Individual new_route;
                                new_route.route = new_g;
                                new_route.distance = calculateDistance(new_route.route, map);

                                if (new_route.distance < population[i].distance) {
                                        new_population.push_back(new_route);
                                        break;
                                }
                                else {
                                        float prob = pow(2.6, -1 * ((float)(new_route.distance -
population[i].distance) / temperature));
                                        if (prob > 0.5) {
                                                new_population.push_back(new_route);
                                                break;
                                        }
                                }
                        }
                }
                temperature = coolDown(temperature);
                population = new_population;
                cout << " Generation " << gen << " " << endl;
                cout << " ROUTE\t    DISTANCE" << endl;

                for (int i = 0; i < POPULATION_SIZE; i++)
                        cout << " " << population[i].route << " " << population[i].distance <<
endl;
                gen++;
        }
}
```

```
int main() {
	vector<vector<int>> distance_map = {
					{INF, 10, 3, 2, 5, 6, 7, 2, 5, 4},
					{20, INF, 3, 5, 10, 2, 8, 1, 15, 6},
					{10, 5, INF, 7, 8, 3, 11, 12, 3, 2},
					{3, 4, 5, INF, 6, 4, 10, 6, 1, 8},
					{1, 2, 3, 4, INF, 5, 10, 20, 11, 2},
					{8, 5, 3, 10, 2, INF, 6, 9, 20, 1},
					{3, 8, 5, 2, 20, 21, INF, 3, 5, 6},
					{5, 2, 1, 25, 15, 10, 6, INF, 8, 1},
					{10, 11, 6, 8, 3, 4, 2, 15, INF, 1},
					{5, 10, 6, 4, 15, 1, 3, 5, 2, INF}
				};
	TravellingSalesman(distance_map);
	return 0;
}
```

Output: The program outputs multiple routes which can lead to the shortest possible total distance. This repeats for some number of generations, (here I have chosen 4).

```
Initial Population:
ROUTE      DISTANCE
06985241370 48
08694175320 66
08437265190 67
02743915680 75
04826753910 94
03795642180 69
07261389540 33
03561274980 64
08962375140 53
07329814560 71
01782953460 57
01268957340 72
08524397610 71
02496318570 55
01934276580 98
```

```
Current Temp: 10000
Generation 1
ROUTE      DISTANCE
07261489530 58
06987241350 58
08964375120 67
07496318520 60
01782953640 69
03861274950 54
08697145320 60
08437215690 43
03795842160 52
07369814520 84
08574392610 90
01268457390 84
02743195680 66
04826751930 64
01934576280 64
```

```
Current Temp: 9500
Generation 2
ROUTE      DISTANCE
08467215390 52
03725849160 58
09861274350 62
03261489570 63
05987241360 53
07496358120 72
08647195320 81
04826791530 62
01937546280 66
02943175680 60
08264375190 78
01872953640 85
07369418520 92
01628457390 81
08574192630 63
```

```
Current Temp: 9025
Generation 3
ROUTE      DISTANCE
08467125390 52
05897241360 61
03721849560 44
02743195680 66
09461278350 80
04826971530 61
03961482570 65
05874192630 86
01437596280 62
07436958120 82
07264385190 59
08547196320 59
07628451390 47
01872653940 107
04369718520 64
```

```
Current Temp: 8573
Generation 4
ROUTE      DISTANCE
03712849560 31
07628154390 48
08467215390 52
07264385910 74
08547296310 63
05897246310 77
04896271530 54
01437596820 65
04369518720 72
03561482970 59
02743895610 71
02461978350 68
07435968120 58
05478192630 75
07812653940 90
```

After this, the program ends. We see that the smallest possible distance is 31 for the route 0->3->7->1->2->8->4->9->5->6->0.