CSL-412 ARTIFICAL INTELLIGENCE

VNIT

Assignment 2: Genetic Algorithm on TSP

SUBMITTED ON OCTOBER 20, 2020 BY

Mugdha S Kolhe

BT17CSE043

APPROACH

1. Initialize the population randomly.(Permutations)
2. Determine the fitness of the chromosome. (Cost of path of traversal)
3. Until done repeat: (IN iterations/temperature/saturation)
    1. Select parents (Roulette Wheel Selection)
    2. Perform ordered crossover and select the best child.
    3. Mutate the child (swapping) according the mutation rate(MR)
    3. Calculate the fitness of the new population and remember the fittest individual
    4. Append it to the gene pool
5. Return the global best individual.

STRUCTURE OF THE CODE

Functions:

BT17CSE043_AI_Assignment2.cpp | input1.txt | output2.txt | input2.txt | output1.txt | input3.txt | output3.txt | input4.txt | output4.

```
//function to print from a vector of vector
void print_pop(vector <vector < int > > pop)
{

//function to print from vector of integers
void print(vector <int> vect)
{

//function to print from vector of floats
void printf(vector <float> vect)
{

//function to calculate fitness of each member from the population
int fitness_calc(vector<int> sol, vector< vector < int > > graph)
{

//function to create initial randomize population
vector<int>permute(int n)
{

//function to create intial population
vector <vector < int> > create_pop(int n)
{

//select parent according to its fitnesss value
vector<int> random_selection (vector <vector <int > > population, vector <int> fitness)
{

//Stochastic Universal Sampling as no negative values of fitness function
vector<int> sus (vector <int> x, vector <int> y)
{

//create a child from two parents
vector<int> cross_over(vector<int>x, vector<int> y)
{

//function to swap a vector given two indexes
vector<int> swap(vector <int> s, int i, int j)
{

//.function to create mutation according to given mutation rate
```

```
214   //create a child from two parents
215   vector<int> cross_over(vector<int>x, vector<int> y)
216 ⊞ {
266
267   //function to swap a vector given two indexes
268   vector<int> swap(vector <int> s, int i, int j)
269 ⊞ {
287
288   //.function to create mutation according to given mutation rate
289   vector<int> mutation(vector<int> vect)
290 ⊞ {
311
312   //func in case you use temp in place of iterations
313   int cooldown(int temp)
314 ⊞ {
317
318   //Genetic Algorithm for TSP
319   vector<int> GeneticAlgoTSP(vector <vector < int > > graph)
320 ⊞ {
439
440   int main()
441 ⊞ {
```

PARAMETERS

```
#define SIZE 75
#define START 1
#define NI 1000
#define K 50
#define MR 0.5
#define CR 50
```

SIZE: when you want to create input graph of specified size

START: start node

NI: No of iterations

K: population size

MR: mutation rate

CR: No of children generated from 2 parents. (Best selected)

INPUT FORMAT

First line of the file contains no of nodes.

Second line contains name of cities

Adjacency Matrix

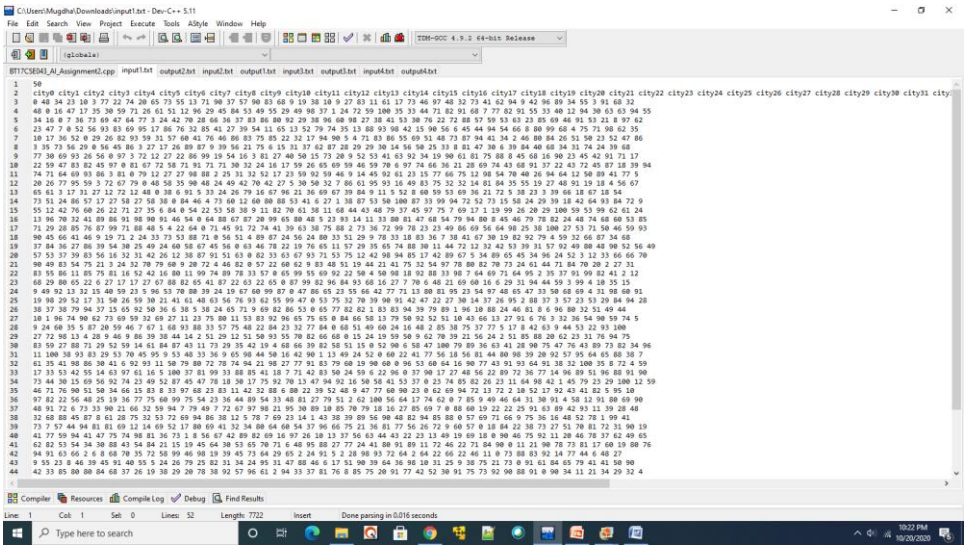
OUTPUT FORMAT

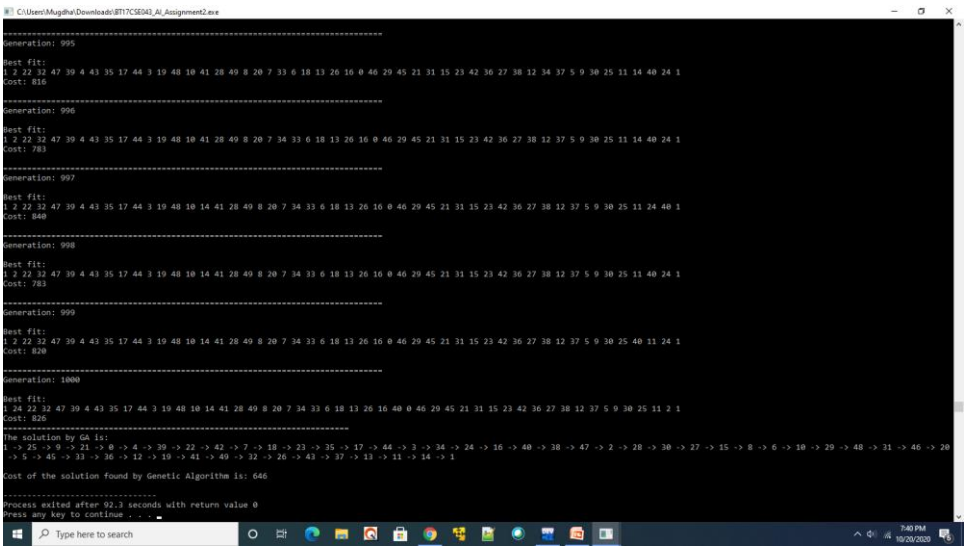
Best Path in each generation

Best sequence obtained(min cost)

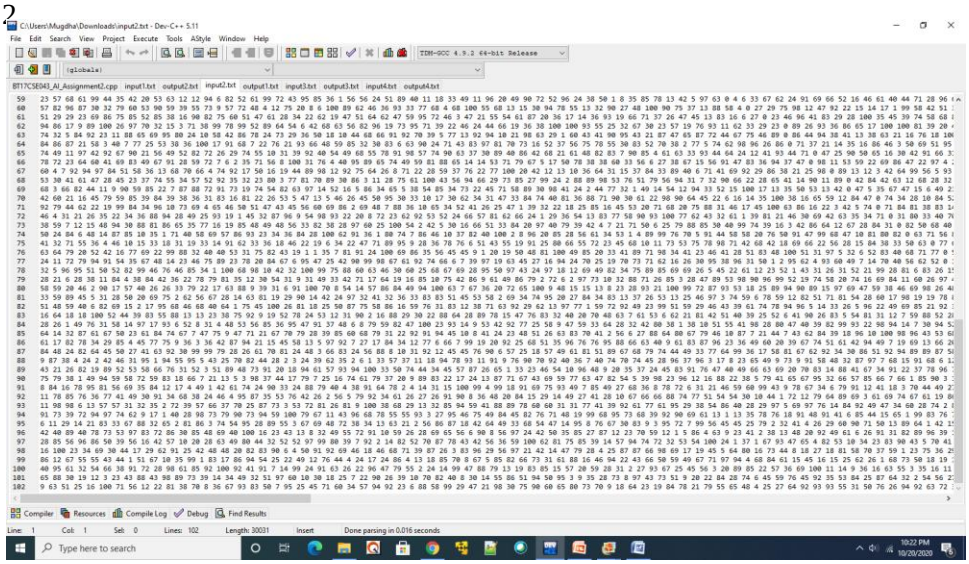
SCREENSHOTS OF THE RUNNING APPLICATION (AND THE EXPECTED INPUTS AND OUTPUTS ACHIEVED)

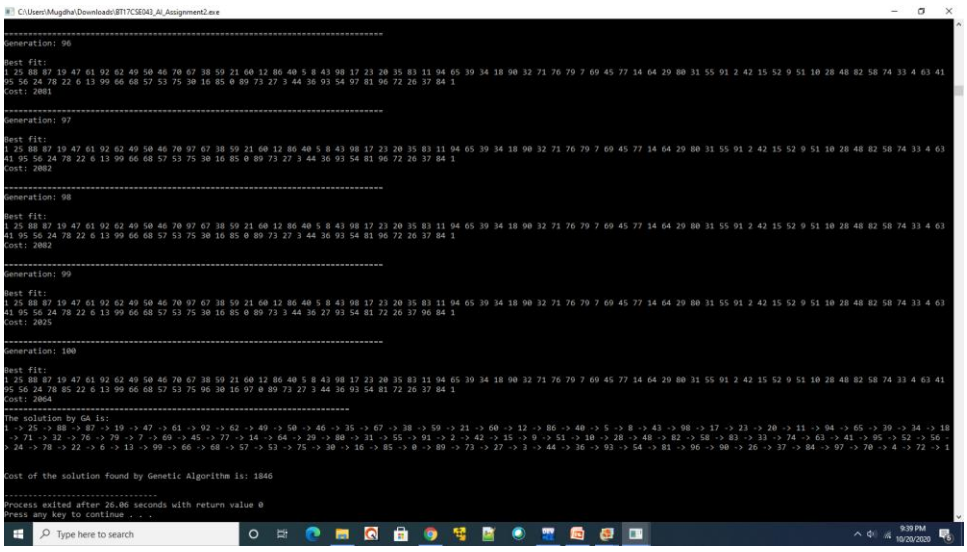
1.



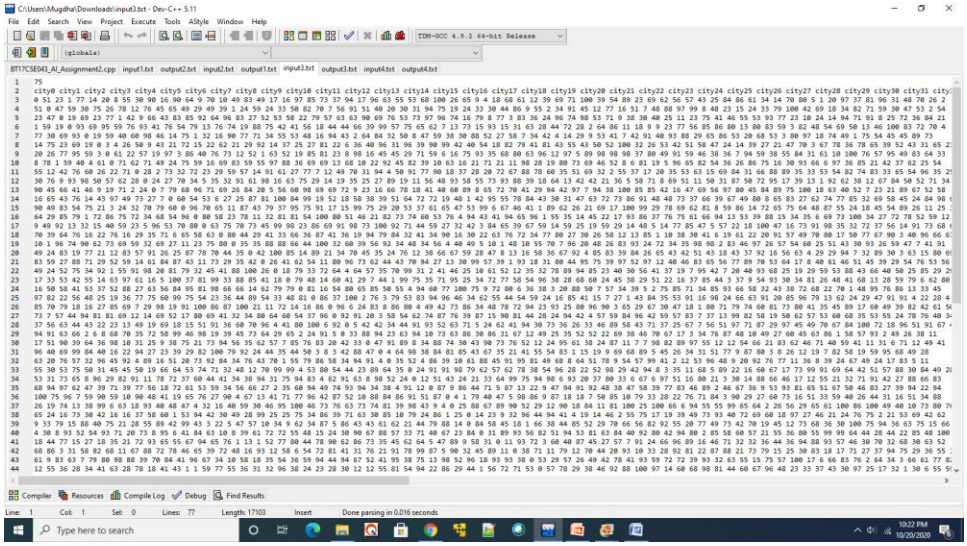


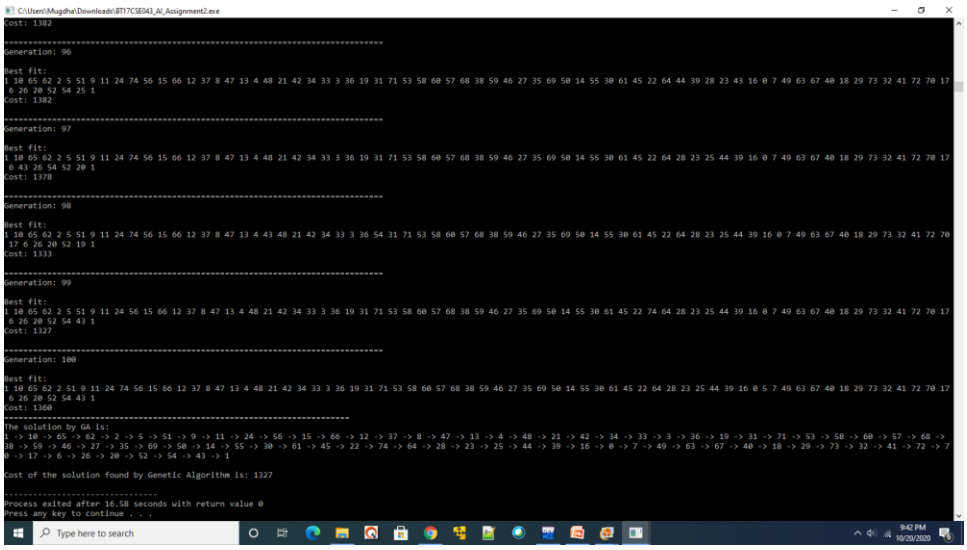
2



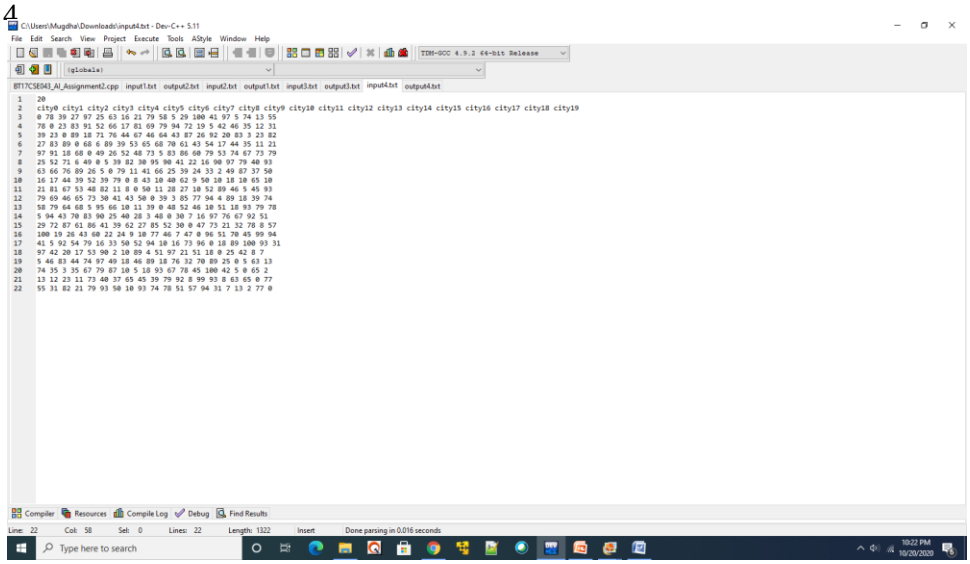

Generation: 96
Best fit:
Cost: 2081
Generation: 97
Best fit:
Cost: 2082
Generation: 98
Best fit:
Cost: 2082
Generation: 99
Best fit:
Cost: 2025
Generation: 100
Best fit:
Cost: 2064
The solution by GA is:
Cost of the solution found by Genetic Algorithm is: 1846
Process exited after 26.06 seconds with return value 0
Press any key to continue . . .

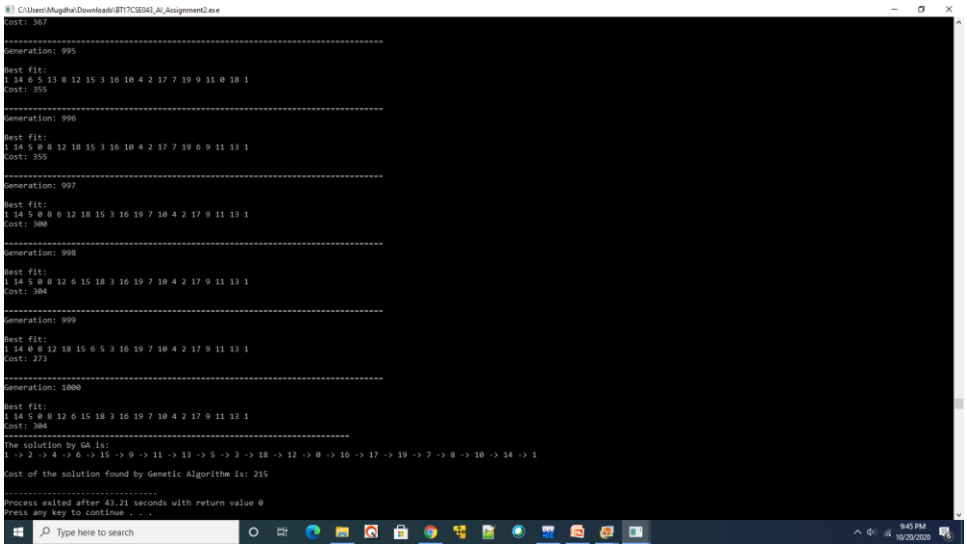
3.




Cost: 1382
Generation: 96
Best fit:
1 10 65 62 2 5 51 9 11 24 74 56 15 66 12 37 8 47 13 4 48 21 42 34 33 3 36 19 31 71 53 58 60 57 68 38 59 46 27 35 69 50 14 55 30 61 45 22 64 44 39 28 23 43 16 0 7 49 63 67 40 18 29 73 32 41 72 70 17
6 26 20 52 54 25 1
Cost: 1382
Generation: 97
Best fit:
1 10 65 62 2 5 51 9 11 24 74 56 15 66 12 37 8 47 13 4 48 21 42 34 33 3 36 19 31 71 53 58 60 57 68 38 59 46 27 35 69 50 14 55 30 61 45 22 64 28 23 25 44 39 16 0 7 49 63 67 40 18 29 73 32 41 72 70 17
6 43 26 54 52 20 1
Cost: 1378
Generation: 98
Best fit:
1 10 65 62 2 5 51 9 11 24 74 56 15 66 12 37 8 47 13 4 43 48 21 42 34 33 3 36 54 31 71 53 58 60 57 68 38 59 46 27 35 69 50 14 55 30 61 45 22 64 28 23 25 44 39 16 0 7 49 63 67 40 18 29 73 32 41 72 70
17 6 26 20 52 19 1
Cost: 1333
Generation: 99
Best fit:
1 10 65 62 2 5 51 9 11 24 56 15 66 12 37 8 47 13 4 48 21 42 34 33 3 36 19 31 71 53 58 60 57 68 38 59 46 27 35 69 50 14 55 30 61 45 22 74 64 28 23 25 44 39 16 0 7 49 63 67 40 18 29 73 32 41 72 70 17
6 26 20 52 54 43 1
Cost: 1327
Generation: 100
Best fit:
1 10 65 62 2 51 9 11 24 74 56 15 66 12 37 8 47 13 4 48 21 42 34 33 3 36 19 31 71 53 58 60 57 68 38 59 46 27 35 69 50 14 55 30 61 45 22 64 28 23 25 44 39 16 0 5 7 49 63 67 40 18 29 73 32 41 72 70 17
6 26 20 52 54 43 1
Cost: 1360
The solution by GA is:
1 -> 10 -> 65 -> 62 -> 2 -> 5 -> 51 -> 9 -> 11 -> 24 -> 56 -> 15 -> 66 -> 12 -> 37 -> 8 -> 47 -> 13 -> 4 -> 48 -> 21 -> 42 -> 34 -> 33 -> 3 -> 36 -> 19 -> 31 -> 71 -> 53 -> 58 -> 60 -> 57 -> 68 ->
38 -> 59 -> 46 -> 27 -> 35 -> 69 -> 50 -> 14 -> 55 -> 30 -> 61 -> 45 -> 22 -> 74 -> 64 -> 28 -> 23 -> 25 -> 44 -> 39 -> 16 -> 0 -> 7 -> 49 -> 63 -> 67 -> 40 -> 18 -> 29 -> 73 -> 32 -> 41 -> 72 -> 7
0 -> 17 -> 6 -> 26 -> 20 -> 52 -> 54 -> 43 -> 1
Cost of the solution found by Genetic Algorithm is: 1327
Process exited after 16.58 seconds with return value 0
Press any key to continue . . .

4


C:\Users\Mugdha\Downloads\input4.txt - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
BT17CSE043_AI_Assignment2.cpp input1.txt output2.txt input2.txt output1.txt input3.txt output3.txt input4.txt output4.txt
20
city0 city1 city2 city3 city4 city5 city6 city7 city8 city9 city10 city11 city12 city13 city14 city15 city16 city17 city18 city19
0 78 39 27 97 25 63 16 21 79 58 5 29 100 41 97 5 74 13 55
78 0 23 83 91 52 66 17 81 69 79 94 72 19 5 42 46 35 12 31
39 23 0 89 18 71 76 44 67 46 64 43 87 26 92 20 83 3 23 82
27 83 89 0 68 6 89 39 53 65 68 70 61 43 54 17 44 35 11 21
97 91 18 68 0 49 26 52 48 73 5 83 86 68 79 53 74 67 73 79
25 52 71 6 49 0 5 39 82 38 95 90 41 22 16 90 97 79 40 93
63 66 76 89 26 5 0 79 11 41 66 25 39 24 33 2 49 87 37 58
16 17 44 39 52 39 79 0 8 43 18 40 62 9 50 18 18 18 65 18
21 81 67 53 48 82 11 8 0 50 11 28 27 10 52 89 46 5 45 93
79 69 46 65 73 38 41 43 58 0 39 3 85 77 94 4 89 18 39 74
58 79 64 68 5 95 66 18 11 39 0 48 52 46 18 51 18 93 79 78
5 94 43 70 83 90 25 40 28 3 48 0 30 7 16 97 76 67 92 51
29 72 87 61 86 41 39 62 27 85 52 30 0 47 73 21 32 78 8 57
100 19 26 43 68 22 24 9 18 77 46 7 47 0 96 51 70 45 99 94
41 5 92 54 79 16 33 50 52 94 18 16 73 96 0 18 89 100 93 31
97 42 20 17 53 90 2 18 89 4 51 97 21 51 18 0 25 42 8 7
5 46 83 44 74 97 49 18 46 89 18 76 32 70 89 25 0 5 63 13
74 35 3 35 67 79 87 18 5 18 93 67 78 45 100 42 5 0 65 2
13 12 23 11 73 40 37 65 45 39 79 92 8 99 93 8 63 65 0 77
55 31 82 21 79 93 50 18 93 74 78 51 57 94 31 7 13 2 77 0
Compiler Resources Compile Log Debug Find Results
Line: 22 Col: 58 Sel: 0 Lines: 22 Length: 1322 Insert Done parsing in 0.016 seconds
Type here to search
10:22 PM
10/20/2020


C:\Users\Mugdha\Downloads\BT17CSE043_AI_Assignment2.exe
Cost: 367
Generation: 995
Best fit:
1 14 6 5 13 8 12 15 3 16 10 4 2 17 7 19 9 11 0 18 1
Cost: 355
Generation: 996
Best fit:
1 14 5 0 8 12 18 15 3 16 10 4 2 17 7 19 6 9 11 13 1
Cost: 355
Generation: 997
Best fit:
1 14 5 0 8 6 12 18 15 3 16 19 7 10 4 2 17 9 11 13 1
Cost: 300
Generation: 998
Best fit:
1 14 5 0 8 12 6 15 18 3 16 19 7 10 4 2 17 9 11 13 1
Cost: 304
Generation: 999
Best fit:
1 14 0 8 12 18 15 6 5 3 16 19 7 10 4 2 17 9 11 13 1
Cost: 273
Generation: 1000
Best fit:
1 14 5 0 8 12 6 15 18 3 16 19 7 10 4 2 17 9 11 13 1
Cost: 304
The solution by GA is:
1 -> 2 -> 4 -> 6 -> 15 -> 9 -> 11 -> 13 -> 5 -> 3 -> 18 -> 12 -> 0 -> 16 -> 17 -> 19 -> 7 -> 8 -> 10 -> 14 -> 1
Cost of the solution found by Genetic Algorithm is: 215
Process exited after 43.21 seconds with return value 0
Press any key to continue . . .
Type here to search
9:45 PM
10/20/2020