/\* A C++ Program developed to implement Ant Colony Algorithm for Traveling Salesman Problem(TSP) \*

\* DEVELOPED BY: ADEJUMO TAOFEEK TUNDE

\* MATRIC NO: 14/CSH/0582

\*/

#include <process.h>

#include <cstdlib>

#include <iostream>

#include <ctype.h>

#include <math.h>

#define MAX 10

#define INFINIT 9999

#define ZERO 0

using namespace std;

char City[MAX];

int CityIndex[MAX];

int NumberOfCities;

int Distance[MAX][MAX];

int bCost;

void getBestRoutes(int nIndex[][MAX], int N, int mCost, int K)

{

int i;

if(bCost > mCost)

{

bCost = mCost; //Set new value for the best Cost

for (i = 0; i < N; i++)

CityIndex[i] = nIndex[K][i];

}

}

void displayBestRoutes(int nIndex[], int N, int mCost, char strDisplay[])

{

int i;

cout<<strDisplay;

cout<<"\nCity: ";

for (i = 0; i < N; i++)

{

cout<<City[nIndex[i]]<<"-";

}

cout<<City[0]<<" = "<<mCost<<endl;

}

int getDistance(int nIndex[][MAX], int v, int k)

{

//Get the total distance represented by the ant movement in nIndex}

int t, I;

t = 0;

for (I = 1; I < v; I++)

t = t + Distance[nIndex[k][I-1]] [nIndex[k][I]];

t = t + Distance[nIndex[k][v]][0];

getBestRoutes(nIndex, v, t, k); //Check if and get the best routes

return t;

}

void DisplayRoutes(int nIndex[][MAX], int v, int k, char strDisplay[])

{

int i;

cout<<(strDisplay);

cout<<"Ant "<<k+1<<" City: ";

for (i = 0; i < v; i++)

{

cout<<City[nIndex[k][i]]<<"-";

}

cout<<City[0]<<" = "<< getDistance(nIndex, v, k);

}

void AntColony(int nIterations, int nAnts)

{

int Alpha = 1, Beta = 2;

double h[MAX][MAX], Gthoinit[MAX][MAX], thoinit[MAX][MAX];

int I, J, N, C;

double P[MAX], CumP[MAX];

double SumP, r, e;

int Count, K, V;

int RIndex[MAX][MAX];

int totalDistance[MAX];

N = NumberOfCities; Count = 1;

for (I = 0; I < N; I++)

{

for (J = 0; J < N; J++)

{

Gthoinit[I][J] = 1; //Initialize value of Global pheromones between cities}

}

}

label5: K = 0; //Initialize no of ants}

cout<<"\nIteration "<<Count<<":"<<endl;

label10:

for (I = 0; I < N; I++)

{

for (J = 0; J < N; J++)

{

h[I][J] = ZERO;

if(Distance[I][J] != ZERO)

h[I][J] = 1.0/Distance[I][J]; //{Initialize value of visibility between cities}

}

}

for (I = 0; I < N; I++)

{

for (J = 0; J < N; J++)

{

thoinit[I][J] = Gthoinit[I][J]; //{Initialize value of Local pheromones between cities}

}

}

C = 0; // {Initialize the starting city as city 1}

V = 0; RIndex[K][V] = C; //{Set the starting city as city 1 for ant K}

label20:

for (I = 0; I < N; I++)

h[I][C] = ZERO; //{Initialize value of visibility of the current City, C to NULL}

//{Calculate the probabilities to visit other cities from the current city (C)}

SumP = 0.0;

for (J = 0; J < N; J++)

{

P[J] = (double)pow(thoinit[C][J], Alpha) \* pow(h[C][J], Beta);

SumP = SumP + P[J]; //{Initialize value of visibility of the current City, C to NULL}

}

//{Compute the probabilities of ant K going to another city from city C}

for (J = 0; J < N; J++)

P[J] = P[J]/SumP;

//{Compute the Cummulative probabilities of ant K going to another city from city C}

CumP[0] = P[0];

for (J = 1; J < N; J++)

CumP[J] = CumP[J-1] + P[J];

//{\* Generate a random number r for the choice of cities to be visited by comparing it with

//\* the value of the cummulative probabilities}

//randomize;

r = (1.0 - 0.0) \* rand()/(RAND\_MAX + 1.0);

for (J = 1; J < N; J++)

{

if(r < CumP[J])

{

C = J; // {Get the next city index}

goto label30;

}

}

label30:

V = V + 1; RIndex[K][V] = C; //{Store the next city to visited}

if(V < N) goto label20;

DisplayRoutes(RIndex, V, K, "\nThe route starting from city 1 for the TSP:\n");

totalDistance[K] = getDistance(RIndex, V, K); //{Get the total distance represented by the ant movement in RIndex}

K = K + 1; //{Set the next ant K into operation}

if(K < nAnts) goto label10;

//{Using the total distance to update the global pheromones}

e = 0.5; //{initialize the evaporation coefficient}

for (I = 0; I < N; I++)

{

for (J = 0; J < N; J++)

Gthoinit[I][J] = (1 - e) \* Gthoinit[I][J];

}

for (K = 0; K < nAnts; K++)

{

for (V = 1; V < N; V++)

{

Gthoinit[RIndex[K][V-1]][RIndex[K][V]] = Gthoinit[RIndex[K][V-1]][RIndex[K][V]] + 1.0/totalDistance[K];

}

}

Count = Count + 1; //{Increase the number of Iteration}

if(Count < nIterations) goto label5; //{Repeat operations for the next iteration}

}

int main(int argc, char \*argv[])

{

//Main program body starts here

cout<<"Welcome to C++ Program developed to implement Ant Colony Algorithm for TSP:";

cout<<"\n===============================================================================";

cout<<"\nRead in Number(between 1 and 10) of Nodes(Cities) in Graph G(V,E) for the TSP: ";

cin>>NumberOfCities;

if ((NumberOfCities < 1) || (NumberOfCities > MAX))

cout<<"Wrong Input! Please, Read in correct digit from 1 to 10:";

else

{

//\*Construction of the Graph G(V,E) for the TSP\*)

ReadNodes(City, CityIndex, NumberOfCities); // Read in N nodes\*)

ReadDistance(Distance, NumberOfCities, City); //Read in weight/cost of each edges\*)

//\*performing Brute Force Algorithm on the Graph for TSP\*)

//InitIndices(RIndex, SIndex, NumberOfCities);

cout<<"\nAnt Colony Optimization Alogrithm Result I: ";

int numberOfIterations = 50;

int numberOfAnts = 5;

AntColony(numberOfIterations, numberOfAnts);// {Call to Ant Colony Optimization method to travel thru cities in the graph G }

cout<<"\nThe mimimum Cost = "<<bCost<<" after "<<numberOfIterations<<" Iteration with "<<numberOfAnts<<" Ants";

displayBestRoutes(CityIndex, NumberOfCities, bCost,"\nThe best route starting from city 1 for the TSP:");

}

system("PAUSE");

return EXIT\_SUCCESS;

}

(\* A Pascal Program developed to implement Ant Colony Algorithm for Traveling Salesman Problem(TSP)

\* DEVELOPED BY: ADEJUMO TAOFEEK TUNDE

\* MATRIC NO: 14/CSH/0582

\*)

Program AntColony\_TSP;

uses Crt;

Const

MAX = 10;

INFINITY = 9999;

NULL = 0;

TYPE

tbl2Dim = Array[1..MAX, 1..MAX] of Integer;

strTbl = Array[1..MAX] of String;

tbl1Dim = Array[1..MAX] of Integer;

Var

City :strTbl;

CityIndex :tbl1Dim;

NumberOfCities :Integer;

Distance :tbl2Dim;

bCost :Integer;

ch : char;

numberOfIterations :Integer;

numberOfAnts :Integer;

Procedure getBestRoutes(nIndex :tbl2Dim; N :Integer; mCost: Integer; K :Integer);

Var

i : Integer;

Begin

If(bCost > mCost) Then

Begin

bCost := mCost; {Set new value for the best Cost}

For i := 1 To N do

Begin

CityIndex[i] := nIndex[K,i];

End

End

End;

Procedure displayBestRoutes(nIndex :tbl1Dim; N :Integer; mCost: Integer; strDisplay :String);

Var

i : Integer;

Begin

Writeln(strDisplay);

Write('City: ');

For i := 1 To N do

Begin

Write(City[nIndex[i]],'-');

End;

Writeln(City[1],' = ', mCost);

End;

Function getDistance(nIndex :tbl2Dim; v :Integer; k :Integer):Integer;

{Get the total distance represented by the ant movement in RIndex}

Var

t, I :Integer;

Begin

t:= 0;

For I := 2 to v do

t := t + Distance[nIndex[k, I-1], nIndex[k, I]];

t := t + Distance[nIndex[k, v], 1];

getBestRoutes(nIndex, v, t, k); {Check if and get the best routes}

getDistance := t;

End;

Procedure DisplayRoutes(nIndex : tbl2Dim; v :Integer; k :Integer; strDisplay :String);

Var

i : Integer;

Begin

Writeln(strDisplay);

Write('Ant ', k, ' City: ');

For i := 1 To v do

Begin

Write(City[nIndex[k,i]],'-');

End;

Writeln(City[1],' = ', getDistance(nIndex, v, k));

End;

Procedure AntColony(nIterations :Integer; nAnts :Integer);

label 5, 10, 20, 30;

{Const

Alpha = 1;

Beta = 2; }

var

h, Gthoinit, thoinit :Array[1..MAX, 1..MAX] of Real;

I, J, N, C :Integer;

P, CumP :Array[1..Max] of real;

SumP, r, e : real;

Count, K, V :Integer;

RIndex : tbl2Dim;

totalDistance : tbl1Dim;

Begin

N := NumberOfCities; Count := 1;

For I := 1 to N do

Begin

For J := 1 to N do

Begin

Gthoinit[I,J] := 1 {Initialize value of Global pheromones between cities}

End

End;

5: K := 1; {Initialize no of ants}

Writeln('Iteration', Count,': ');

10:

For I := 1 to N do

Begin

For J := 1 to N do

Begin

h[I,J] := NULL;

if(Distance[I,J] <> NULL) Then

h[I,J] := 1.0/Distance[I,J] {Initialize value of visibility between cities}

End

End;

For I := 1 to N do

Begin

For J := 1 to N do

Begin

thoinit[I,J] := Gthoinit[I,J] {Initialize value of Local pheromones between cities}

End

End;

C := 1; {Initialize the starting city as city 1}

V := 1; RIndex[K, V] := C; {Set the starting city as city 1 for ant K}

20:

For I := 1 to N do

h[I,C] := NULL; {Initialize value of visibility of the current City, C to NULL}

{Calculate the probabilities to visit other cities from the current city (C)}

SumP := 0.0;

For J := 1 to N do

Begin

{P[J] := exp(Alpha \*ln(thoinit[C, J])) \* exp(Beta \*ln(h[C, J]));}

P[J] := thoinit[C, J] \* sqr(h[C, J]);

SumP := SumP + P[J] {Initialize value of visibility of the current City, C to NULL}

End;

{Compute the probabilities of ant K going to another city from city C}

For J := 1 to N do

P[J] := P[J]/SumP;

{Compute the Cummulative probabilities of ant K going to another city from city C}

CumP[1] := P[1];

For J := 2 to N do

CumP[J] := CumP[J-1] + P[J];

{\* Generate a random number r for the choice of cities to be visited by comparing it with

\* the value of the cummulative probabilities}

{randomize; }

{r := (1.0 - 0.0) \* random(1);}

r := random; {generate a pseudorandom number in the interval [0,1), .i.e. 0 <= result < 1}

For J := 2 to N do

Begin

If( r < CumP[J]) Then

Begin

C:= J; {Get the next city index}

goto 30

end

End;

30:

V := V + 1; RIndex[K,V] := C; {Store the next city to visited}

If(V < N) then goto 20;

DisplayRoutes(RIndex, V, K, 'The route starting from city 1 for the TSP:');

totalDistance[K] := getDistance(RIndex, V, K); {Get the total distance represented by the ant movement in RIndex}

K := K + 1; {Set the next ant K into operation}

If(K <= nAnts)Then goto 10;

{Using the total distance to update the global pheromones}

e := 0.5; {initialize the evaporation coefficient}

For I := 1 to N do

Begin

For J := 1 to N do

Gthoinit[I,J] := (1 - e) \* Gthoinit[I,J];

end;

For K := 1 to nAnts do

Begin

For V := 2 to N do

Begin

Gthoinit[RIndex[K,V-1],RIndex[K,V]] := Gthoinit[RIndex[K,V-1],RIndex[K,V]] + 1.0/totalDistance[K];

End;

End;

Count := Count + 1; {Increase the number of Iteration}

If(Count <= nIterations) Then goto 5; {Repeat operations for the next iteration}

End;

Begin

{Main program body starts here}

randomize;

Writeln('Welcome to Pascal Program developed to implement Ant Colony Algorithm for TSP:');

Writeln('===============================================================================');

Writeln('Read in Number(between 1 and 10) of Nodes(Cities) in Graph G(V,E) for the TSP: ');

Readln(NumberOfCities);

If ((NumberOfCities < 1) OR (NumberOfCities > MAX))Then

Writeln('Wrong Input! Please, Read in correct digit from 1 to 10:')

Else

Begin

(\*Construction of the Graph G(V,E) for the TSP\*)

ReadNodes(City, CityIndex, NumberOfCities); (\* Read in N nodes\*)

ReadDistance(Distance, NumberOfCities, City); (\*Read in weight/cost of each edges\*)

(\*performing Brute Force Algorithm on the Graph for TSP\*)

{InitIndices(RIndex, SIndex, NumberOfCities);}

Writeln('Ant Colony Optimization (ACO) Alogrithm Result: ');

numberOfIterations := 50;

numberOfAnts := 5;

AntColony(numberOfIterations, numberOfAnts); {Call to Ant Colony Optimization method to travel thru cities in the graph G }

Writeln; Writeln('The mimimum Cost =', bCost,' after ' , numberOfIterations,' Iteration with ', numberOfAnts,' Ants');

displayBestRoutes(CityIndex, NumberOfCities, bCost, 'The best route starting from city 1 for the TSP:');

End;

Writeln('Press any key to continue....');

ch := ReadKey;

End.

/\* A C++ Program developed to implement Brute Force Algorithm for Traveling Salesman Problem(TSP) \*

\* DEVELOPED BY: ADEJUMO TAOFEEK TUNDE

\* MATRIC NO: 14/CSH/0582

\*/

#include <cstdlib>

#include <iostream>

#include <ctype.h>

#define MAX 10

#define INFINITY 9999

#define ZERO 0

using namespace std;

char City[MAX];

int CityIndex[MAX], RIndex[MAX], SIndex[MAX];

int NumberOfCities;

int Distance[MAX][MAX];

int minimumCost, bCost;

void InitIndices(int R[], int S[], int N)

{

int I;

R[0] = CityIndex[0]; S[0] = ZERO;

for (I = 1; I < N; I++)

{

R[I] = ZERO; S[I] = CityIndex[I];

}

bCost = INFINITY; //Initialize best cost

}

void Swap(int& X, int& Y)

{

int temp;

temp = X;

X = Y;

Y = temp;

}

int minimumValue(int X, int Y)

{

int temp;

temp = X;

if(temp > Y)

temp = Y;

return temp;

}

void getBestRoutes(int nIndex[], int N, int mCost)

{

int i;

if(bCost > mCost)

{

bCost = mCost; //Set new value for the best Cost

for (i = 0; i < N; i++)

SIndex[i] = nIndex[i];

}

}

void displayRoutes(int nIndex[], int N, int mCost, char strDisplay[])

{

int i;

cout<<strDisplay;

cout<<"\nCity: ";

for (i = 0; i < N; i++)

{

cout<<City[nIndex[i]]<<"-";

}

cout<<City[0]<<" = "<<mCost<<endl;

}

int TSP\_BruteForce(int A[], int l, int lengthSoFar, int R[])

{

int minCost, I, newLength, N;

N = NumberOfCities;

if (l == N-1)

{

minCost = lengthSoFar + Distance[A[N-1]][A[0]]; //Finish by returning to city 1

displayRoutes(R, N, minCost,"\nThe route starting from city 1 for the TSP:");

getBestRoutes(R, N, minCost);

}

else

{

minCost = INFINITY;

for (I = l + 1; I < N; I++)

{

Swap(A[l+1], A[I]); //Select A[i] as the next city}

newLength = lengthSoFar + Distance[A[l]][A[l+1]];

R[l+1] = A[l+1];

minCost = minimumValue(minCost, TSP\_BruteForce(A, l+1, newLength, R));

Swap(A[l+1], A[I]); //Undo Selection}

}

}

return minCost;

}

int main(int argc, char \*argv[])

{

//Main program body starts here

cout<<"Welcome to C++ Program developed to implement Brute Foce Algorithm for TSP:";

cout<<"\n===============================================================================";

cout<<"\nRead in Number(between 1 and 10) of Nodes(Cities) in Graph G(V,E) for the TSP: ";

cin>>NumberOfCities;

if ((NumberOfCities < 1) || (NumberOfCities > MAX))

cout<<"Wrong Input! Please, Read in correct digit from 1 to 10:";

else

{

//\*Construction of the Graph G(V,E) for the TSP\*)

ReadNodes(City, CityIndex, NumberOfCities); // Read in N nodes\*)

ReadDistance(Distance, NumberOfCities, City); //Read in weight/cost of each edges\*)

//\*performing Brute Force Algorithm on the Graph for TSP\*)

InitIndices(RIndex, SIndex, NumberOfCities);

cout<<"\nBrute Force Alogrithm Result I: ";

minimumCost = TSP\_BruteForce(CityIndex, ZERO, ZERO, RIndex); //call to brute force method to travel thru cities in the graph G }

cout<<"\nThe mimimum Cost = "<< minimumCost;

displayRoutes(SIndex, NumberOfCities, bCost,"\nThe best route starting from city 1 for the TSP:");

}

system("PAUSE");

return EXIT\_SUCCESS;

}

(\* A Pascal Program developed to implement Brute-Force Algorithm for Traveling Salesman Problem (TSP)

\* DEVELOPED BY: ADEJUMO TAOFEEK TUNDE

\* MATRIC NO: 14/CSH/0582

\*)

PROGRAM BruteForceTSP(input, output);

Const

MAX = 10;

INFINITY = 9999;

NULL = 0;

TYPE

tbl2Dim = Array[1..MAX, 1..MAX] of Integer;

strTbl = Array[1..MAX] of String;

tbl1Dim = Array[1..MAX] of Integer;

Var

City :strTbl;

CityIndex, RIndex, SIndex :tbl1Dim;

NumberOfCities :Integer;

Distance :tbl2Dim;

minimumCost, bCost :Integer;

Procedure Swap(Var X :Integer; Var Y :Integer);

Var

temp :Integer;

Begin

temp := X;

X := Y;

Y := temp;

End;

Function minimumValue(X :Integer; Y :Integer) :Integer;

Var

temp :Integer;

Begin

temp := X;

if(temp > Y) then

temp := Y;

minimumValue := temp;

End;

Procedure getBestRoutes(nIndex :tbl1Dim; N :Integer; mCost: Integer);

Var

i : Integer;

Begin

If(bCost > mCost) Then

Begin

bCost := mCost; {Set new value for the best Cost}

For i := 1 To N do

Begin

SIndex[i] := nIndex[i];

End

End

End;

Procedure displayRoutes(nIndex :tbl1Dim; N :Integer; mCost: Integer; strDisplay :String);

Var

i : Integer;

Begin

Writeln(strDisplay);

Write('City: ');

For i := 1 To N do

Begin

Write(City[nIndex[i]],'-');

End;

Writeln(City[1],' = ', mCost);

End;

Function TSP\_BruteForce(A :tbl1Dim; l :Integer; lengthSoFar :Integer; R :tbl1Dim) :Integer;

Var

minCost :Integer;

I :Integer;

newLength :Integer;

N :Integer;

Begin

N := NumberOfCities;

If (l = N) then

Begin

minCost := lengthSoFar + distance[A[n], A[1]]; {Finish by returning to city 1}

displayRoutes(R, N, minCost, 'The route starting from city 1 for the TSP:');

getBestRoutes(R, N, minCost);

End

Else

Begin

minCost := INFINITY;

For I := l + 1 to n Do

Begin

Swap(A[l+1], A[i]); {Select A[i] as the next city}

newLength := lengthSoFar + distance[A[l], A[l+1]];

R[L+1] := A[l+1];

minCost := minimumValue(minCost, TSP\_BruteForce(A, l+1, newLength, R));

Swap(A[l+1], A[i]); {Undo Selection}

End

End;

TSP\_BruteForce := minCost;

End;

Procedure InitIndices(Var R :tbl1Dim; Var S :tbl1Dim; N :Integer);

Var

i : Integer;

Begin

R[1] := CityIndex[1]; S[1] := NULL;

For i := 2 To N do

Begin

R[I] := NULL; S[I] := CityIndex[I];

End;

bCost := INFINITY; {Initialise the best cost}

End;

Begin

{Main program body starts here}

Writeln('Welcome to Pascal Program developed to implement Brute Foce Algorithm for TSP:');

Writeln('===============================================================================');

Writeln('Read in Number(between 1 and 10) of Nodes(Cities) in Graph G(V,E) for the TSP: ');

Readln(NumberOfCities);

If ((NumberOfCities < 1) OR (NumberOfCities > MAX))Then

Writeln('Wrong Input! Please, Read in correct digit from 1 to 10:')

Else

Begin

(\*Construction of the Graph G(V,E) for the TSP\*)

ReadNodes(City, CityIndex, NumberOfCities); (\* Read in N nodes\*)

ReadDistance(Distance, NumberOfCities, City); (\*Read in weight/cost of each edges\*)

(\*performing Brute Force Algorithm on the Graph for TSP\*)

InitIndices(RIndex, SIndex, NumberOfCities);

Writeln('Brute Force Alogrithm Result: ');

minimumCost := TSP\_BruteForce(CityIndex, 1, 0, RIndex); {call to brute force method to travel thru cities in the graph G }

Writeln; Writeln('The mimimum Cost =', minimumCost);

displayRoutes(SIndex, NumberOfCities, minimumCost, 'The best route starting from city 1 for the TSP:');

End;

Writeln('Press any key to continue....');

Readln;

End.