In this assignment I used Kruskal’s algorithm. The kruscal’s algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in forest.

This is algorithm in a graph theory, which finds a minimum spinning tree for a connected weighted graph adding increasing cost arcs at each step.

So first we created a forest F, where each vertex in the graph is a separate tree. So we create a set S containing all the edges in the graph. While S is nonempty and F is not yet spanning: 1) remove an edge with minimum weight from S. 2) if the removed edge connects two different trees then add it to the forest F, combining two trees into a single tree.

At the termination of the algorithm, the forest forms a minimum spanning forest of the graph. If the graph is connected, the forest has a single component and forms a minimum spanning tree.

**Pseudocode:**

KRUSCAL(G):

A = Ø

foearch= v { G.V :

MAKE-SET(v)

foreach (u,v) ordered by weight (u,v), increasing:

if FIND-SET (u) != FIND-SET(v):

A= A U {(u,v)}

UNION(u,v)

Return A

Reference: <https://en.wikipedia.org/wiki/Kruskal%27s_algorithm#Pseudocode>

Here you can see the code and it’s explanation

classdef TSPExecution < handle

properties

m = []

sz = 20

end

methods

function out = calc (this)

cnt = 0;

A = [];

B = [];

C = [];

id = [];

for i = 1 : this.sz

id(i) = i; % assigning id to each cell

for j = i + 1 : this.sz

x1 = this.m(i,1);%getting first x1

y1 = this.m(i,2); %getting first y1

x2 = this.m(j,1);%getting cordinate x2

y2 = this.m(j,2);%cordinate y2

cnt = cnt + 1;%increasing size of array

A(cnt) = ((x1 - x2) ^ 2 + (y1 - y2) ^ 2)^0.5;% euclid distance btw 2 points

B(cnt) = i;%vertex i

C(cnt) = j;%vertex j

end

end

%sorting all edges by it distances from lower to higher

for i = 1 : cnt

for j = i + 1 : cnt

if A(i) > A(j)

curAi = A(i);

curAj = A(j);

curBi = B(i);

curBj = B(j);

curCi = C(i);

curCj = C(j);

A(i) = curAj;

A(j) = curAi;

B(i) = curBj;

B(j) = curBi;

C(i) = curCj;

C(j) = curCi;

end

end

end

res = 0;

%going through sorted list

for i = 1: cnt

l = A(i);

a = B(i);

b = C(i);

if (id(a) ~= id (b)) %if ids not equal

res = res + l; %adding cost to the cur length

ol = id (b);

ne = id (a);

for j = 1:this.sz

if (id (j) == ol) %colloring second component to new one

id (j) = ne;

end

end

end

end

out = res;

end

function this = printFunc(this)

disp ('Distance : ');

disp (this.calc); %printing answer

end

function this = TSPExecution (this)

this.m = xlsread('input.xlsx'); %getting data from 'input.xlsx'

this.printFunc;%this function prints our result

end

end

end

**The Output:**

