201533661 이승수’s algorithm homework#6 date: 2016.10.09

Saving Ink

**<code>**

#include <stdio.h>

#include <math.h>

struct coordinate{

float x;

float y;

}dot[30];

FILE \*inF, \*outF;

int dotNum = 0;

float distanceTable[30][30] = { 0 };

int linkTable[30][30] = { 0 };

int visitedNode[30] = { 1 };//start visiting at dot[0]

int visitedCount = 1;//already visited dot[0]

int searchMinimumDestination(int from);

float Distance(int i, int j);

void main()//search for Minimum Spanning Tree using Prim's Algorithm

{

inF = fopen("input#3.txt","r");

fscanf(inF,"%d",&dotNum);

for (int i = 0; i < dotNum; i++)

{

fscanf(inF,"%f %f",&dot[i].x,&dot[i].y);

}

fclose(inF);

float totalDistance = 0;

for (int i = 0; i < dotNum; i++)//put distance in table

{

for (int j = 0; j < dotNum; j++)

{

distanceTable[i][j] = Distance(i,j);

}

}//checked

while (visitedCount != dotNum)

{

//search(from i)-found(j)

//printf("\nWhileLooping!!");

int closeFrom[30] = {0};//list of minimum-cost destination from index

int minAt = 0;//closeFrom[minAt] have shortest path within visited node

for (int i = 0; i < dotNum; i++)//make minimumDistance

{

if (visitedNode[i] ==1)

{

closeFrom[i]=searchMinimumDestination(i);

}

}

for (int i = 0; i < dotNum; i++)//from minAt to closeFrom[minAt] is shortest path at visitedNodes

{

if (distanceTable[i][closeFrom[i]] < distanceTable[minAt][closeFrom[minAt]])

minAt = i;

}

//count

//printf("\nminAt:%d, closeFrom[minAt]:%d distance:%f",minAt,closeFrom[minAt],distanceTable[minAt][closeFrom[minAt]]);

totalDistance += distanceTable[minAt][closeFrom[minAt]];

linkTable[minAt][closeFrom[minAt]]++;//become 1

linkTable[closeFrom[minAt]][minAt]++;

visitedNode[closeFrom[minAt]]++;

visitedCount++;

printf("\n");//one looped

}//all nodes are visited,linkTable fitted

//result

outF = fopen("output#3.txt","w");

fprintf(outF,"%.2f",totalDistance);

fclose(outF);

}//end Main

int searchMinimumDestination(int from)//return closest destination's index from (int from), considering cycle

{

int sortedIndex[30] = {0};

for (int j = 0; j < dotNum; j++)

{

sortedIndex[j] = j;

}

for (int k = 0; k < dotNum; k++)//sort sortedIndex[] by distance

{

for (int j = (dotNum-1); j > k; j--)

{

if (distanceTable[from][sortedIndex[j - 1]]>distanceTable[from][sortedIndex[j]])

{

int temp = sortedIndex[j - 1];

sortedIndex[j - 1] = sortedIndex[j];

sortedIndex[j] = temp;

}

}

}

int minIndex=sortedIndex[1];//except from to itself

for (int j = 1; j<(dotNum-1);j++)

{

if (sortedIndex[j] == from)

continue;

else if (minIndex == sortedIndex[j] && visitedNode[sortedIndex[j]] != 0 && linkTable[from][sortedIndex[j]] != 0)//if minIndex is visited

{

minIndex = sortedIndex[j + 1];

}

}

return minIndex;

}

float Distance(int i,int j)//compute distance between dot[i] and dot[j]

{

float xLen = (dot[i].x - dot[j].x);

float yLen = (dot[i].y - dot[j].y);

return sqrt(xLen\*xLen + yLen\*yLen);

}