Closest Pair Algorithm Formulas uses

Distance between Two Points is used in this Algorithm

Code snippet:

float ans;

ans=pow((point2.x\_coord - point1.x\_coord), 2) + pow((point2.y\_coord - point1.y\_coord), 2);

return sqrt(ans);

ClosestPairAlgorithm.cpp

#include <bits/stdc++.h>

using namespace std;

class Graph

{

public:

float x\_coord, y\_coord;

};

bool compareTwoPointX(Graph point1, Graph point2)

{

return (point1.x\_coord < point2.x\_coord);

}

bool compareTwoPointY(Graph point1, Graph point2)

{

return (point1.y\_coord < point2.y\_coord);

}

void printElements(Graph points[])

{

for (int i = 0; i < 100; i++) {

cout << points[i].x\_coord << " " << points[i].y\_coord << "\n";

}

}

float min(float a, float b)

{

float ret;

if(a<b){

ret=a;

}

ret=b;

return ret;

}

float distance(Graph point1, Graph point2)

{

float ans;

ans=pow((point2.x\_coord - point1.x\_coord), 2) + pow((point2.y\_coord - point1.y\_coord), 2);

return sqrt(ans);

}

float closestPair(Graph points[], int Totalsize)

{

float minValue = FLT\_MAX;

if (Totalsize <= 3) {

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

for (int j = i + 1; j < Totalsize; j++)

if (distance(points[i], points[j]) < minValue)

minValue = distance(points[i], points[j]);

return minValue;

}

int mid = Totalsize / 2;

Graph middle = points[mid];

float left = closestPair(points, mid);

float right = closestPair(points + mid, Totalsize - mid);

float smallest\_distance = min(left, right);

return smallest\_distance;

}

float middleClosest(Graph middlePoints[], int size, float smallest\_distance)

{

float min = smallest\_distance;

sort(middlePoints, middlePoints + size, compareTwoPointY);

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

for (int j = i + 1; j < size && (middlePoints[j].y\_coord - middlePoints[i].y\_coord) < min; ++j)

if (distance(middlePoints[i], middlePoints[j]) < min)

min = distance(middlePoints[i], middlePoints[j]);

return min;

}

float middlePoints(Graph points[], int size, int smallest\_distance){

Graph mid = points[size / 2];

Graph middlePoints[size];

int j = 0;

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

if (abs(points[i].x\_coord - mid.x\_coord) < smallest\_distance){

middlePoints[j] = points[i];

j++;

}

return middleClosest(middlePoints, j, smallest\_distance);

}

int main()

{

srand((unsigned int)time(0));

Graph randomPoints[100];

for (int i = 0; i < 100; i++) {

randomPoints[i].x\_coord = rand() % 100;

randomPoints[i].y\_coord = rand() % 100;

}

sort(randomPoints, randomPoints + 100, compareTwoPointX);

float smallest\_number = closestPair(randomPoints, 100);

float middle\_smallest = middlePoints(randomPoints, 100, smallest\_number);

if (smallest\_number < middle\_smallest)

cout <<"\n"<<smallest\_number;

else

cout <<"\n"<<middle\_smallest;

return 0;

}

OUTPUT

The output is tested with GCC Compiler

Prints the smallest distance between the closest pair at the end