Md. Asad Chowdhury Dipu

Id: 2019-1-60-093

Section: 01

Ans#2(the naïve algorithm)

#include <float.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

struct Point

{

int x, y;

};

float Dist(struct Point p1, struct Point p2)

{

return sqrt( (p1.x - p2.x)\*(p1.x - p2.x) +

(p1.y - p2.y)\*(p1.y - p2.y)

);

}

struct Point \* BFClosestPoints(struct Point P[], int n)

{

if ((n < 2) || (P == NULL))

{

fprintf(stderr, "Error: not enough points to find closest pair");

exit(EXIT\_FAILURE);

}

struct Point \* closestPair = (struct Point \*)malloc(2 \* (sizeof(struct Point)));

if (closestPair == NULL)

{

fprintf(stderr, "Error: memory allocation failed");

exit(EXIT\_FAILURE);

}

float Min = FLT\_MAX;

float dist = 0.0;

int i, j;

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

{

for (j = i+1; j < n; ++j)

{

dist = Dist(P[i],P[j]);

if (dist < Min)

{

Min = dist;

closestPair[0] = P[i];

closestPair[1] = P[j];

}

}

}

return closestPair;

}

int main()

{

int i,n;

printf("Enter The Number Of Pairs: ");

scanf("%d",&n);

struct Point onPlain[n];

printf("Enter Pair coordinates X & Y : ");

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

{

scanf("%d%d",&onPlain[i].x,&onPlain[i].y);

}

struct Point \*closest = BFClosestPoints(onPlain,n) ;

printf( "Closest Pairs: \n"

"X1: %i Y1: %i\n"

"X2: %i Y2: %i\n"

"Distance: %f: ", closest[0].x, closest[0].y, closest[1].x, closest[1].y,

Dist(closest[0],closest[1]));

return 0;

}

Ans#1 (Divide and Conquer algorithm)

#include <float.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

struct Point

{

int x, y;

};

int CompXcoo(const void\* a, const void\* b)

{

struct Point \*p1 = (struct Point \*)a, \*p2 = (struct Point \*)b;

return (p1->x - p2->x);

}

int CompYcoo(const void\* a, const void\* b)

{

struct Point \*p1 = (struct Point \*)a, \*p2 = (struct Point \*)b;

return (p1->y - p2->y);

}

float min(float x, float y)

{

return (x < y)? x : y;

}

float Dist(struct Point p1, struct Point p2)

{

return sqrt( (p1.x - p2.x)\*(p1.x - p2.x) +

(p1.y - p2.y)\*(p1.y - p2.y)

);

}

float bruteForce(struct Point P[], int n, struct Point \*p1, struct Point \*p2)

{

float min = FLT\_MAX;

int i, j;

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

for (j = i+1; j < n; ++j)

if (Dist(P[i], P[j]) < min)

{

min = Dist(P[i], P[j]);

\*p1 = P[i];

\*p2 = P[j];

}

return min;

}

float stripClosest(struct Point strip[], int size, float d, struct Point \*p1, struct Point \*p2)

{

float min = d;

qsort(strip, size, sizeof(struct Point), CompYcoo);

int i, j;

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

for (j = i+1; j < size && (strip[j].y - strip[i].y) < min; ++j)

if (Dist(strip[i],strip[j]) < min)

{

min = Dist(strip[i], strip[j]);

\*p1 = strip[i];

\*p2 = strip[j];

}

return min;

}

float closestUtil(struct Point P[], int n, struct Point \*p1, struct Point \*p2)

{

static struct Point pTemp1, pTemp2, pTemp3, pTemp4;

if (n <= 3)

return bruteForce(P, n, &pTemp1, &pTemp2);

int mid = n/2;

struct Point midpoint = P[mid];

float dl = closestUtil(P, mid, &pTemp1, &pTemp2);

float dr = closestUtil(P + mid, n-mid, &pTemp3, &pTemp4);

if(dl < dr)

{

\*p1 = pTemp1;

\*p2 = pTemp2;

}

else

{

\*p1 = pTemp3;

\*p2 = pTemp4;

}

float dmin = min(dl, dr);

struct Point strip[n];

int j = 0, i;

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

{

if (abs(P[i].x - midpoint.x) < dmin)

{

strip[j] = P[i], j++;

}

}

double dmin\_strip = stripClosest(strip, j, dmin, &pTemp1, &pTemp2);

double final\_min = dmin;

if(dmin\_strip < dmin)

{

\*p1 = pTemp1;

\*p2 = pTemp2;

final\_min = dmin\_strip;

}

return final\_min;

}

struct Point \* closestPoints(struct Point P[], int n)

{

if ((n < 2) || (P == NULL))

{

fprintf(stderr, "Error: not enough points to find closest pair");

exit(EXIT\_FAILURE);

}

struct Point \* closestPair = (struct Point \*)malloc(2 \* (sizeof(struct Point)));

if (closestPair == NULL)

{

fprintf(stderr, "Error: memory allocation failed");

exit(EXIT\_FAILURE);

}

struct Point p1, p2;

qsort(P, n, sizeof(struct Point), CompXcoo);

closestUtil(P, n, &p1, &p2);

closestPair[0] = p1;

closestPair[1] = p2;

return closestPair;

}

int main()

{

int i,n;

printf("Enter The Number Of Pairs: ");

scanf("%d",&n);

struct Point onPlain[n];

printf("Enter Pair coordinates X & Y : ");

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

{

scanf("%d%d",&onPlain[i].x,&onPlain[i].y);

}

struct Point \*Closest = closestPoints(onPlain,n) ;

printf( "Closest Pairs: \n"

"X1: %i Y1: %i\n"

"X2: %i Y2: %i\n"

"Distance: %f ", Closest[0].x, Closest[0].y, Closest[1].x, Closest[1].y

,Dist(Closest[0],Closest[1]));

return 0;

}