江 蘇 大 學

JIANGSU UNIVERSITY

算法课后实现题

学院名称：计算机科学与通信工程学院

专业班级：信息安全 1501

学生姓名：沈鑫楠

学生学号：3150604028

教师姓名：朱小龙

完成日期：2018 年 5 月 13 日

一、最接近点对问题

1.问题描述

输入 n 个点的坐标，输出最接近的两个点之间的距离。

2.问题解决步骤

（1）输入 n 个点的坐标

（2）递归地求左半边和右半边的最小点间距 d

（3）筛选出横坐标为中点-d 到中点+d 的范围，并将这些点按纵坐标排序

（4）对于已排好序的筛选出的点，如果两点间距 dist 小于 d，则 d=dist，直到确定后面的点间距不可能小于 d 为止

（5）返回 d 的值

3.完整实现代码（c++）

```
#include <graphics.h>
#include <conio.h>
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <ctype.h>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <vector>
#define eps 1e-8 //精度
#define INF 0x7fffffff //无穷大
#define PI acos(-1.0) //精确值
#define seed 31 //种子
#define maxn 100005 //n 的最大值
#define min(X,Y) X<Y?X:Y //两个数的最小值
#define min3(X,Y,Z) X<(Y<Z?Y:Z)?X:(Y<Z?Y:Z) //三个数的最小值
typedef long LL;
```

```
typedef unsigned long ULL;
using namespace std;
//分治算法求二维最近点对
struct Point
{
   double x,y;
}p[maxn];//定义点的结构体
int a[maxn];
int minpix1=-1,minpix2=-1;
inline int cmpx(Point a,Point b)//比较 x 坐标
{
   return a.x<b.x;
}
inline int cmpy(int a,int b)//比较 y 坐标
{
   return p[a].y<p[b].y;
}
inline double dis(Point a,Point b)//计算两点间距
{
     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
double closest(int low,int high)//计算一系列点间距的最小值
{
     int i,j,k;
     int mid3;
     if(low+1==high) //只有两个点
     {
       minpix1=low;
       minpix2=high;
       return dis(p[low],p[high]);
     }
     if(low+2==high)//只有三个点
     {
        mid3=low+1;
        double dis1=dis(p[low],p[high]);
```

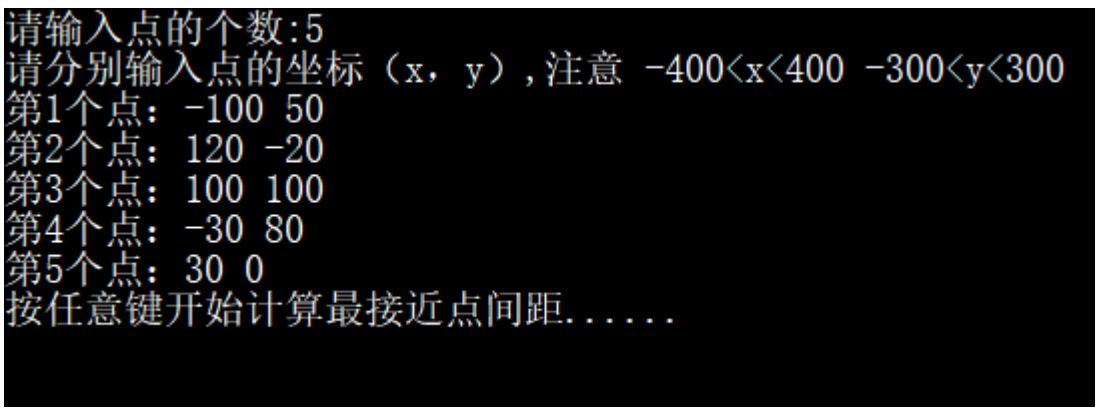
```
    double dis2=dis(p[low],p[mid3]);
    double dis3=dis(p[mid3],p[high]);
    if(dis1<=dis2&&dis1<=dis3)
    {
        minpix1=low;
        minpix2=high;
    }
    else if(dis2<=dis1&&dis2<=dis3)
    {
        minpix1=low;
        minpix2=mid3;
    }
    else
    {
        minpix1=mid3;
        minpix2=high;
    }
    return min3(dis(p[low],p[high]),dis(p[low],p[mid3]),dis(p[mid3],p[high]));
}
int mid=(low+high)/2; //求中点即左右子集的分界线
double d=min(closest(low,mid),closest(mid+1,high));//左边和右边的最小点间距
for(i=low,k=0;i<=high;i++)//把 x 坐标在 p[mid].x-d   ~   p[mid].x+d 范围内的点筛选出来
{
    if(p[i].x>=p[mid].x-d&&p[i].x<=p[mid].x+d){
        a[k++]=i; //保存这些点的下标索引
    }
}
sort(a,a+k,cmpy); //按 y 坐标进行升序排序
for(i=0;i<k;i++)
{
  for(j=i+1;j<k;j++)
  {
      if(p[a[j]].y-p[a[i]].y>=d) //注意下标索引
            break;
      if(dis(p[a[i]],p[a[j]])<d)
```

```
            {
                minpix1=a[i];
                minpix2=a[j];
            }
            d=min(d,dis(p[a[i]],p[a[j]]));
        }
    }
    return d;
}
int main(int argc, char* argv[])
{
    int i,n;
    cout<<"请输入点的个数:";
    cin>>n;
    cout<<"请分别输入点的坐标（x，y）,注意  -400<x<400 -300<y<300 "<<endl;
    for(i=0;i<n;i++)
    {
        do
        {
            cout<<"第"<<i+1<<"个点：";
            cin>>p[i].x>>p[i].y;
        }
        while(p[i].x<-400||p[i].x>400||p[i].y<-300||p[i].y>300);
    }
    sort(p , p + n , cmpx);//按 x 坐标进行升序排序
    cout<<"按任意键开始计算最接近点间距.....";
    getch();
    double mindist=closest(0,n-1);
    char buffer[20];
    sprintf(buffer,"%lf",mindist);
    char noti[50]="最短距离：";
    // 初始化图形模式
    initgraph(800, 600);
    setbkcolor(0x005478);//设置背景色
    cleardevice();//清屏
```

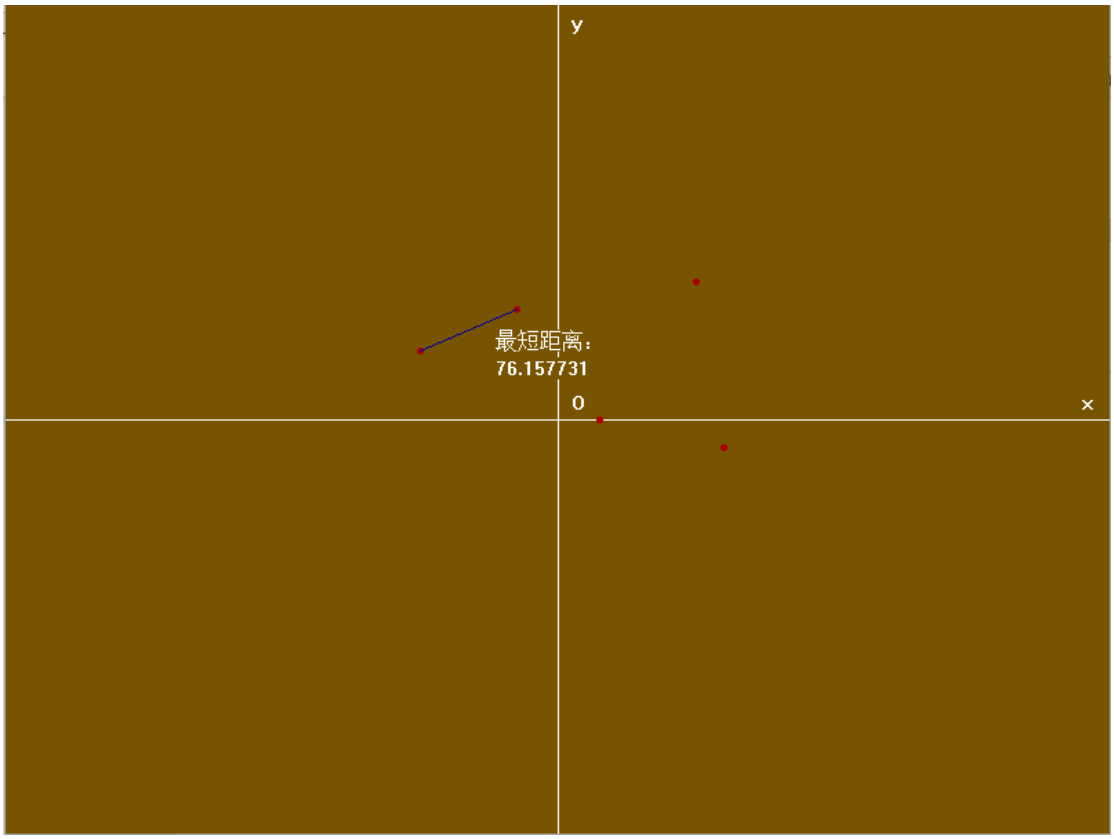
```
        setcolor(WHITE);
        line(0,300,800,300);
        line(400,0,400,600);
        outtextxy(410,280,'O');
        outtextxy(780,280,'x');
        outtextxy(410,5,'y');
        BeginBatchDraw();
        setfillcolor(RED);
        setlinecolor(RED);
        for(i=0;i<n;i++)
              fillcircle(p[i].x+400,-p[i].y+300,2);
        setcolor(BLUE);
        line(p[minpix1].x+400,-p[minpix1].y+300,p[minpix2].x+400,-p[minpix2].y+300);
        setcolor(WHITE);
        outtextxy((p[minpix1].x+400+p[minpix2].x+400)/2+20,(-p[minpix1].y+300-
p[minpix2].y+300)/2,noti);
        outtextxy((p[minpix1].x+400+p[minpix2].x+400)/2+20,(-p[minpix1].y+300-
p[minpix2].y+300)/2+20,buffer);
        FlushBatchDraw();
        EndBatchDraw();
        while(!kbhit());//按任意键退出
        closegraph();
        return 0;
}
```

4.示例运行结果

（1）示例输入：



（2）示例输出：



二、Prim 算法求最小生成树

1.问题描述

输入图的邻接矩阵，打印出最小生成树的权值和。

2.问题解决步骤

（1） 设初始结点为 v1，将一个图分为两部分，一部分归为点集 U，一部分归为点集 V，U 的初始集合为{V1}，V 的初始集合为{除 v1 外的点}。

（2） 针对 U 开始找 U 中各节点的所有关联的边的权值最小的那个，然后将关联的节点 Vi 加入到 U 中，并且从 V 中删除，如果形成了环就跳过这个结点。

（3） 递归执行步骤(2)，直到 V 中的集合为空。

（4） U 中所有节点构成的树就是最小生成树。

（5）返回最小生成树中的权值之和

3.完整实现代码（c++）

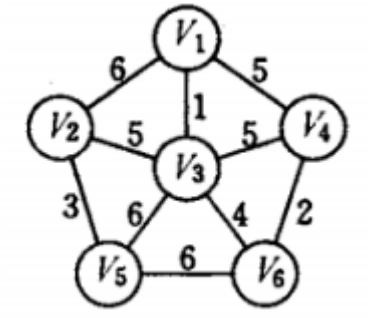
```
#include <iostream>
#include <vector>
using namespace std;
#define max_int 0x7FFFFFFF
//Prim 算法实现
void prim_alg()
{
    int n;
    cout<<"请输入结点数目：";
    cin>>n;
    vector<vector<int> > a(n, vector<int>(n));
    cout<<"请输入邻接矩阵（"<<n<<"*"<<n<<"），注：无路径请输入-1"<<endl;
    for(int i = 0; i < n ; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cin>>a[i][j];
            if(a[i][j]<0) a[i][j]=max_int;
        }
    }
    cout<<"最小生成树生成中....."<<endl;
    cout<<"(";
    int flag=1;
```

```
    int pos=0, minimum;
    int min_tree = 0;
    //lowcost 数组记录每 2 个点间最小权值，visited 数组标记某点是否已访问
    vector<int> visited, lowcost;
    for ( i = 0; i < n; i++)
        visited.push_back(0);       //初始化为 0，表示都没加入
    visited[0] = 1;     //最小生成树从第一个顶点开始
    for ( i = 0; i < n; i++)
        lowcost.push_back(a[0][i]);       //权值初始化为 0
    for ( i = 0; i < n; i++) //枚举 n 个顶点
    {
        minimum = max_int;
        for (int j = 0; j < n; j++) //找到最小权边对应顶点
        {
            if(!visited[j] && minimum > lowcost[j])
            {
                minimum = lowcost[j];
                pos = j;
            }
        }
        if (minimum == max_int)
//如果 min = max_int 表示已经不再有点可以加入最小生成树中
            break;
        min_tree += minimum;
        if(flag) cout<<minimum;
        else cout<<","<<minimum;
        flag=0;
        visited[pos] = 1;         //加入最小生成树中
        for ( j = 0; j < n; j++)
       if(!visited[j] && lowcost[j] > a[pos][j]) lowcost[j] = a[pos][j];     //更新可更新边的权值
    }
    cout<<")"<<endl;
    cout<<"最小生成树生成完毕，权值为：";
    cout<<min_tree<<endl;
}
```

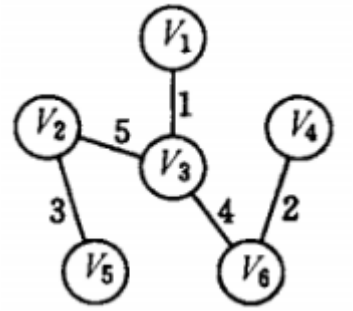
```
int main(int argc, char* argv[])
{
    prim_alg();
    return 0;
}
```

4.示例运行结果

（1）运行示例采用的图为：



（2）生成的最小生成树为：



(3) 运行示例：

```
请输入结点数目：6
请输入邻接矩阵（6*6），注：无路径请输入-1
0 6 1 5 -1 -1
6 0 5 -1 3 -1
1 5 0 5 6 4
5 -1 5 0 -1 2
-1 3 6 -1 0 6
-1 -1 4 2 6 0
最小生成树生成中.....
(1,4,2,5,3)
最小生成树生成完毕，权值为：15
```