Công thức sử dụng để tính diện tích đa giác lồi tạo bởi N điểm: Shoelace formula

Thuật toán sử dụng để tìm 2 điểm gần nhất trong N điểm: Sweep line

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  #include <math.h>  #include <set>  #include <iomanip>  #define ll long long  #define ld long double  const double eps = (double)1e-6;  const double inf = (double)1e9;  bool equals(double a,double b){      return fabs(a-b) < eps;  }  using namespace std;  struct point{      ld x,y;      ll id;      point(){}      point(ld x,ld y) : x(x),y(y){}      point(ld x,ld y,int id) : x(x),y(y),id(id){}      bool operator < (const point& other) const{          if (equals(x,other.x)) return y<other.y;          return x < other.x;      }      bool operator == (const point& other) const{          return equals(x,other.x) && equals(y,other.y);      }  };  ostream& operator << (ostream& o, const point& a){      o << '(' << a.x << ',' << a.y << ')';      return o;  }  struct cmp{      bool operator () (const point& a, const point& b) const {          if (!equals(a.y,b.y)) return a.y < b.y;          return a.x < b.x;      }  };  ld dist(const point& a,const point& b){      ld u = (a.x-b.x)\*(a.x-b.x);      ld v = (a.y-b.y)\*(a.y-b.y);      return sqrt(u+v);  }  ld squared\_dist(const point& a,const point& b){      ld u = (a.x-b.x)\*(a.x-b.x);      ld v = (a.y-b.y)\*(a.y-b.y);      return u+v;  }  // Shoelace formula - bai 1  ld area(const vector<point>& list){      int n = list.size();      ld ans=0;      for (int i = 0;i < n;++i){          int j = (i+1)%n;          ans += (list[i].x\*list[j].y - list[j].x\*list[i].y);      }      return fabs(ans/2.0);  }  // sweep line - bai 2  ld shortest\_dist(vector<point> list){      set<point,cmp>st;      sort(list.begin(),list.end());      ld squared\_d = squared\_dist(list[0],list[1]);      for (point p : list){          ld x = p.x;          ld y = p.y;            ld d = sqrt(squared\_d);          point cur(-1000001,y-d,p.id);          while(true){              auto it = st.upper\_bound(cur);              if (it == st.end()) break;              cur = (\*it);              if (cur.y > y + d) break;              if (cur.x < x-d){                  st.erase(it);                  continue;              }              if (squared\_dist(p,cur) < squared\_d){                  squared\_d = squared\_dist(p,cur);              }          }          st.insert(p);      }      return sqrt(squared\_d);  }  int main(){      int n;      cout << "Nhap n: ";      cin >> n;      vector<point>list;      ld a,b;      for (int i=1;i<=n;++i){          cin >> a >> b;          list.push\_back({a,b,i-1});      }      cout << "N diem da nhap la: " << '\n';      for (const point& p : list){          cout << p.x << ' ' << p.y << '\n';      }      cout <<"Dien tich tao nen tu n diem tren la: "<<fixed<<setprecision(6) << area(list) << '\n';      cout <<"Khoang cach 2 diem gan nhat trong n diem tren la: " <<fixed<<setprecision(6)<<shortest\_dist(list);      return 0;  } |

