黄宇凡补充模板

1. ac自动机

struct Ac{

int ch[maxn \* 50][26];

int fail[maxn \* 50];

int val[maxn \* 50];

int q[maxn \* 50];

int tot;

Ac(){}

int id(char c){

return c - 'a';

}

void init(){

tot = 0;

fail[0] = -1;memset(ch,0,sizeof(ch));val[0] = 0;

}

void insert(char \*s,int len){

int now = 0;

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

int tr = id(s[i]);

if(ch[now][tr] == 0){

ch[now][tr] = ++tot;

memset(ch[tot],0,sizeof(ch[tot]));

val[tot] = 0;

fail[tot] = 0;

}

now = ch[now][tr];

}

val[now]++;

}

void make(){

int h = 1,t = 0;

for(int i = 0;i < 26;i++) if(ch[0][i] != 0) q[++t] = ch[0][i];

while(h <= t){

int now = q[h++];

for(int i = 0;i < 26;i++){

if(ch[now][i] == 0){

ch[now][i] = ch[fail[now]][i];

}else{

fail[ch[now][i]] = ch[fail[now]][i];

q[++t] = ch[now][i];

}

}

}

}

int search(char \*s,int len){

int now = 0;int ret = 0;

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

int tr = id(s[i]);

now = ch[now][tr];

int tmp = now;

while(tmp && val[tmp] != -1){

ret += val[tmp];

val[tmp] = -1;

tmp = fail[tmp];

}

}

return ret;

}

}ac;

1. sa

struct SuffixArray{

public:

const static int MAXN = 100000 + 10;

int cnt[MAXN],tr[2][MAXN],ts[MAXN];

int sa[MAXN],rk[MAXN],ht[MAXN],len;

void construct(const char \* s,int n,int m = 256){

int i,j,k,\*x = tr[0],\*y = tr[1];

this->len = n;

memset(cnt,0,sizeof(cnt[0]) \* m);

for(i = 0;i < n;i++) cnt[s[i]]++;

partial\_sum(cnt,cnt + m,cnt);

for(i = 0;i < n;i++) rk[i] = cnt[s[i]] - 1;

for(k = 1;k <= n;k <<= 1){

for(i = 0;i < n;i++) x[i] = rk[i],y[i] = i + k < n ? rk[i + k] + 1:0;

fill(cnt,cnt + n + 1,0);

for(i = 0;i < n;i++) cnt[y[i]]++;

partial\_sum(cnt,cnt + n + 1,cnt);

for(i = n - 1;i >= 0;i--) ts[--cnt[y[i]]] = i;

fill(cnt,cnt + n + 1,0);

for(i = 0;i < n;i++) cnt[x[i]]++;

partial\_sum(cnt,cnt + n + 1,cnt);

for(i = n - 1;i >= 0;i--) sa[--cnt[x[ts[i]]]] = ts[i];

for(i = rk[sa[0]] = 0;i + 1 < n;++i){

rk[sa[i + 1]] = rk[sa[i]] + (x[sa[i]] != x[sa[i + 1]] || (y[sa[i]] != y[sa[i + 1]]));

}

}

for(i = 0,k = 0;i < n;++i){

if(!rk[i]) continue;

for(j = sa[rk[i] - 1];i + k < n && j + k < n && s[i + k] == s[j + k];)k++;

ht[rk[i]] = k;

if(k) k--;

}

rmq\_init(n);

}

inline int lcp(int a,int b){

a = rk[a],b = rk[b];

if(a == b) return len - a + 1;

if(a > b) swap(a,b);

return rmq(a + 1,b);

}

private:

int mx[MAXN][20],LOG[MAXN];

void rmq\_init(int n){

for(int i = -(LOG[0]=-1);i < n;i++) LOG[i] = LOG[i >> 1] + 1;

for(int i = 0;i < n;i++) mx[i][0] = ht[i];

for(int i,j = 1;(1 << j) < n;++j){

for(i = 0;i + (1 << j) <= n;++i){

mx[i][j] = min(mx[i][j - 1],mx[i + (1 << (j - 1))][j - 1]);

}

}

}

inline int rmq(int a,int b){

int k = LOG[b - a + 1];

return min(mx[a][k],mx[b - (1 << k) + 1][k]);

}

};

1. sam

struct Sam{

int fail[maxn \* 2];

int ch[maxn \* 2][256];int tot,last;

int mx[maxn \* 2];

int ans;

void clear(int i){

memset(ch[i],0,sizeof(ch[i]));

}

void clear(){

tot = last = 1;

ans = 0;

clear(0);clear(1);

}

Sam(){

tot = last = 1;

ans = 0;

clear(0);clear(1);

}

int cal(int x){

return mx[x] - mx[fail[x]];

}

void add(int c){

int p = last,np = last = ++tot;clear(tot);

mx[np] = mx[p] + 1;

while(p && ch[p][c] == 0) ch[p][c] = np,p = fail[p];

if(!p) fail[np] = 1,ans += cal(np);

else{

int q = ch[p][c];

if(mx[q] == mx[p] + 1) fail[np] = q,ans += cal(np);

else{

int nq = ++tot;clear(tot);mx[nq] = mx[p] + 1;

memcpy(ch[nq],ch[q],sizeof(ch[nq]));

fail[nq] = fail[q];ans+=cal(nq)-cal(q);

fail[q] = fail[np] = nq;ans+=cal(np)+cal(q);

while(p && ch[p][c] == q) ch[p][c] = nq,p = fail[p];

}

}

}

}sam;

几何部分

1. 整体板子

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <cmath>

#include <vector>

#include <deque>

#include <queue>

using namespace std;

#define rep(i,a,n) for(int i=(a);i<(n);i++)

#define per(i,a,n) for(int i=(n)-1;i>=(a);i--)

#define mp make\_pair

#define pb push\_back

typedef double db;

const db EPS = 1e-8;

inline int sign(db a) {

return a < -EPS ? -1 : a > EPS;

}

inline int cmp(db a, db b){//精度比较函数

return sign(a-b);

}

struct P {//点

db x, y;

P() {}

P(db \_x, db \_y) : x(\_x), y(\_y) {}

P operator+(P p) { return P(x + p.x, y + p.y); }

P operator-(P p) { return P(x - p.x, y - p.y); }

P operator\*(db d) { return P(x \* d, y \* d); }

P operator/(db d) { return P(x / d, y / d); }

bool operator<(P p) const {

int c = cmp(x, p.x);

if (c) return c == -1;

return cmp(y, p.y) == -1;

}

db dot(P p) { return x \* p.x + y \* p.y; }//点积

db det(P p) { return x \* p.y - y \* p.x; }//叉积

db distTo(P p) { return (\*this-p).abs(); }//距离

db alpha() { return atan2(y, x); }//求极角

void read() { cin>>x>>y; }

db abs() { return sqrt(abs2());}//长度

db abs2() { return x \* x + y \* y; }//长度的平方

P rot90() { return P(-y,x);}//逆时针旋转90度

P unit() { return \*this/abs(); }//单位化

int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }//极角是否在[0,180)之间

};

struct L{ //ps[0] -> ps[1]

P ps[2];

P& operator[](int i) { return ps[i]; }

P dir() { return ps[1] - ps[0]; }//直线的方向向量

bool include(P p) { return sign((ps[1] - ps[0]).det(p - ps[0])) > 0; }//是否在左边（半平面交的时候有用）

L push(){ // push eps outward

const double eps = 1e-6;

P delta = (ps[1] - ps[0]).rot90().unit() \* eps;

return {ps[0] - delta, ps[1] - delta};

}

};

#define cross(p1,p2,p3) ((p2.x-p1.x)\*(p3.y-p1.y)-(p3.x-p1.x)\*(p2.y-p1.y))

#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

P isLL(P p1, P p2, P q1, P q2) {//求两直线交点

db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);

return (p1 \* a2 + p2 \* a1) / (a1 + a2);

}

P isLL(L l1,L l2){ return isLL(l1[0],l1[1],l2[0],l2[1]); }//求两直线交点

bool intersect(db l1,db r1,db l2,db r2){

if(l1>r1) swap(l1,r1); if(l2>r2) swap(l2,r2);

return !( cmp(r1,l2) == -1 || cmp(r2,l1) == -1 );

}

bool isSS(P p1, P p2, P q1, P q2){

return intersect(p1.x,p2.x,q1.x,q2.x) && intersect(p1.y,p2.y,q1.y,q2.y) &&

crossOp(p1,p2,q1) \* crossOp(p1,p2,q2) <= 0 && crossOp(q1,q2,p1)

\* crossOp(q1,q2,p2) <= 0;

}

bool isMiddle(db a, db m, db b) {//判断m是否在a,b之间

return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);

}

bool isMiddle(P a, P m, P b) {//判断点m的坐标是否在a,b之间

return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);

}

bool onSeg(P p1, P p2, P q){//判断点是否在直线上

return crossOp(p1,p2,q) == 0 && isMiddle(p1, q, p2);

}

P proj(P p1, P p2, P q) {//关于直线的投影

P dir = p2 - p1;

return p1 + dir \* (dir.dot(q - p1) / dir.abs2());

}

P reflect(P p1, P p2, P q){//关于直线的对称

return proj(p1,p2,q) \* 2 - q;

}

db nearest(P p1,P p2,P q){ //线段p1p2到q的最近距离

P h = proj(p1,p2,q);

if(isMiddle(p1,h,p2))

return q.distTo(h);

return min(p1.distTo(q),p2.distTo(q));

}

db disSS(P p1, P p2, P q1, P q2){ //线段和线段的距离

if(isSS(p1,p2,q1,q2)) return 0;

return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)), min(nearest(q1,q2,p1),nearest(q1,q2,p2)) );

}

db disSS(L l1,L l2){ //线段和线段距离

return disSS(l1[0],l1[1],l2[0],l2[1]);

}

db rad(P p1,P p2){ // 返回p1p2关于原点的弧度

return atan2l(p1.det(p2),p1.dot(p2));

}

db incircle(P p1, P p2, P p3){ //

db A = p1.distTo(p2);

db B = p2.distTo(p3);

db C = p3.distTo(p1);

return sqrtl(A\*B\*C/(A+B+C));

}

//polygon

db area(vector<P> ps){

db ret = 0; rep(i,0,ps.size()) ret += ps[i].det(ps[(i+1)%ps.size()]);

return abs(ret/2);

}

//引向右的虚线判断是否在多边形内

int contain(vector<P> ps, P p){ //2:inside,1:on\_seg,0:outside

int n = ps.size(), ret = 0;

rep(i,0,n){

P u=ps[i],v=ps[(i+1)%n];

if(onSeg(u,v,p)) return 1;

if(cmp(u.y,v.y)<=0) swap(u,v);

if(cmp(p.y,u.y) >0 || cmp(p.y,v.y) <= 0) continue;

ret ^= crossOp(p,u,v) > 0;

}

return ret\*2;

}

vector<P> convexHull(vector<P> ps) { // 凸包

int n = ps.size(); if(n <= 1) return ps;

sort(ps.begin(), ps.end());

vector<P> qs(n \* 2); int k = 0;

for (int i = 0; i < n; qs[k++] = ps[i++])

while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;

for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])

while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;

qs.resize(k - 1);

return qs;

}

vector<P> convexHullNonStrict(vector<P> ps) { //非严格凸多边形

//caution: need to unique the Ps first

int n = ps.size(); if(n <= 1) return ps;

sort(ps.begin(), ps.end());

vector<P> qs(n \* 2); int k = 0;

for (int i = 0; i < n; qs[k++] = ps[i++])

while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;

for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])

while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;

qs.resize(k - 1);

return qs;

}

db convexDiameter(vector<P> ps){ //旋转卡壳求凸多边形直径

int n = ps.size(); if(n <= 1) return 0;

int is = 0, js = 0; rep(k,1,n) is = ps[k]<ps[is]?k:is, js = ps[js] < ps[k]?k:js;

int i = is, j = js;

db ret = ps[i].distTo(ps[j]);

do{

if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j]) >= 0)

(++j)%=n;

else

(++i)%=n;

ret = max(ret,ps[i].distTo(ps[j]));

}while(i!=is || j!=js);

return ret;

}

vector<P> convexCut(const vector<P>&ps, P q1, P q2) {

vector<P> qs;

int n = ps.size();

rep(i,0,n){

P p1 = ps[i], p2 = ps[(i+1)%n];

int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);

if(d1 >= 0) qs.pb(p1);

if(d1 \* d2 < 0) qs.pb(isLL(p1,p2,q1,q2));

}

return qs;

}

//min\_dist

db min\_dist(vector<P>&ps,int l,int r){

if(r-l<=5){

db ret = 1e100;

rep(i,l,r) rep(j,l,i) ret = min(ret,ps[i].distTo(ps[j]));

return ret;

}

int m = (l+r)>>1;

db ret = min(min\_dist(ps,l,m),min\_dist(ps,m,r));

vector<P> qs; rep(i,l,r) if(abs(ps[i].x-ps[m].x)<= ret) qs.pb(ps[i]);

sort(qs.begin(), qs.end(),[](P a,P b) -> bool {return a.y<b.y; });

rep(i,1,qs.size()) for(int j=i-1;j>=0&&qs[j].y>=qs[i].y-ret;--j) ret = min(ret,qs[i].distTo(qs[j]));

return ret;

}

int type(P o1,db r1,P o2,db r2){

db d = o1.distTo(o2);

if(cmp(d,r1+r2) == 1) return 4;

if(cmp(d,r1+r2) == 0) return 3;

if(cmp(d,abs(r1-r2)) == 1) return 2;

if(cmp(d,abs(r1-r2)) == 0) return 1;

return 0;

}

vector<P> isCL(P o,db r,P p1,P p2){

db x = (p1-o).dot(p2-p1), y = (p2-p1).abs2(), d = x \* x - y \* ((p1-o).abs2() - r\*r);

if(sign(d) < 0) return {};

d = max(d,0.0); P m = p1 - (p2-p1)\*(x/y), dr = (p2-p1)\*(sqrt(d)/y);

return {m-dr,m+dr}; //along dir: p1->p2

}

vector<P> isCC(P o1, db r1, P o2, db r2) { //need to check whether two circles are the same

db d = o1.distTo(o2);

if (cmp(d, r1 + r2) == 1) return {};

d = min(d, r1 + r2);

db y = (r1 \* r1 + d \* d - r2 \* r2) / (2 \* d), x = sqrt(r1 \* r1 - y \* y);

P dr = (o2 - o1).unit();

P q1 = o1 + dr \* y, q2 = dr.rot90() \* x;

return {q1-q2,q1+q2};//along circle 1

}

vector<P> tanCP(P o, db r, P p) {//点到圆的两个切点

db x = (p - o).abs2(), d = x - r \* r;

if (sign(d) <= 0) return {}; // on circle => no tangent

P q1 = o + (p - o) \* (r \* r / x);

P q2 = (p - o).rot90() \* (r \* sqrt(d) / x);

return {q1-q2,q1+q2}; //counter clock-wise

}

vector<L> extanCC(P o1, db r1, P o2, db r2) {

vector<L> ret;

if (cmp(r1, r2) == 0) {

P dr = (o2 - o1).unit().rot90() \* r1;

ret.pb({o1 + dr, o2 + dr}), ret.pb({o1 - dr, o2 - dr});

} else {

P p = (o2 \* r1 - o1 \* r2) / (r1 - r2);

vector<P> ps = tanCP(o1, r1, p), qs = tanCP(o2, r2, p);

rep(i,0,min(ps.size(),qs.size())) ret.pb({ps[i], qs[i]}); //c1 counter-clock wise

}

return ret;

}

vector<L> intanCC(P o1, db r1, P o2, db r2) {

vector<L> ret;

P p = (o1 \* r2 + o2 \* r1) / (r1 + r2);

vector<P> ps = tanCP(o1,r1,p), qs = tanCP(o2,r2,p);

rep(i,0,min(ps.size(),qs.size())) ret.pb({ps[i], qs[i]}); //c1 counter-clock wise

return ret;

}

db areaCT(db r, P p1, P p2){

vector<P> is = isCL(P(0,0),r,p1,p2);

if(is.empty()) return r\*r\*rad(p1,p2)/2;

bool b1 = cmp(p1.abs2(),r\*r) == 1, b2 = cmp(p2.abs2(), r\*r) == 1;

if(b1 && b2){

if(sign((p1-is[0]).dot(p2-is[0])) <= 0 &&

sign((p1-is[0]).dot(p2-is[0])) <= 0)

return r\*r\*(rad(p1,is[0]) + rad(is[1],p2))/2 + is[0].det(is[1])/2;

else return r\*r\*rad(p1,p2)/2;

}

if(b1) return (r\*r\*rad(p1,is[0]) + is[0].det(p2))/2;

if(b2) return (p1.det(is[1]) + r\*r\*rad(is[1],p2))/2;

return p1.det(p2)/2;

}

bool parallel(L l0, L l1) { return sign( l0.dir().det( l1.dir() ) ) == 0; }//判断两直线是否平行

bool sameDir(L l0, L l1) { return parallel(l0, l1) && sign(l0.dir().dot(l1.dir()) ) == 1; }//判断两射线是否同向

bool cmp (P a, P b) {

if (a.quad() != b.quad()) {

return a.quad() < b.quad();

} else {

return sign( a.det(b) ) > 0;

}

}

bool operator < (L l0, L l1) {

if (sameDir(l0, l1)) {

return l1.include(l0[0]);

} else {

return cmp( l0.dir(), l1.dir() );

}

}

bool check(L u, L v, L w) {

return w.include(isLL(u,v));

}

vector<P> halfPlaneIS(vector<L> &l) {//求半平面交

sort(l.begin(), l.end());

deque<L> q;

for (int i = 0; i < (int)l.size(); ++i) {

if (i && sameDir(l[i], l[i - 1])) continue;

while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) q.pop\_back();

while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop\_front();

q.push\_back(l[i]);

}

while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop\_back();

while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop\_front();

vector<P> ret;

for (int i = 0; i < (int)q.size(); ++i) ret.push\_back(isLL(q[i], q[(i + 1) % q.size()]));

return ret;

}

int main(){

return 0;

}

1. K次圆

db sqr(db x)

{

return x\*x;

}

int dcmp(double x)

{

if(fabs(x) < EPS) return 0;

else return x < 0 ? -1 : 1;

}

struct Circle

{

double x, y, r, angle;

int d;

Circle(){}

Circle(double xx, double yy, double ang = 0, int t = 0)

{

x = xx; y = yy; angle = ang; d = t;

}

void get()

{

scanf("%lf%lf%lf", &x, &y, &r);

d = 1;

}

};

Circle cir[maxn],tp[maxn\*2];

double area[maxn];

double dis(Circle a,Circle b)

{

return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));

}

double cross(Circle p0,Circle p1,Circle p2)

{

return (p1.x - p0.x) \* (p2.y - p0.y) - (p1.y - p0.y) \* (p2.x - p0.x);

}

//圆相交

int CirCrossCir(Circle p1, double r1,Circle p2, double r2,Circle &cp1,Circle &cp2)

{

double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx \* mx;

double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my \* my;

double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) \* (sq - sqr(r1 + r2));

if (d + eps < 0) return 0; if (d < eps) d = 0; else d = sqrt(d);

double x = mx \* ((r1 + r2) \* (r1 - r2) + mx \* sx) + sx \* my2;

double y = my \* ((r1 + r2) \* (r1 - r2) + my \* sy) + sy \* mx2;

double dx = mx \* d, dy = my \* d; sq \*= 2;

cp1.x = (x - dy) / sq; cp1.y = (y + dx) / sq;

cp2.x = (x + dy) / sq; cp2.y = (y - dx) / sq;

if (d > eps) return 2; else return 1;

}

bool circmp(const Circle& u, const Circle& v)

{

return dcmp(u.r - v.r) < 0;

}

bool cmp(const Circle& u, const Circle& v)

{

if (dcmp(u.angle - v.angle)) return u.angle < v.angle;

return u.d > v.d;

}

//0.5\*r\*r\*(K-sin(K))

double calc(Circle cir,Circle cp1,Circle cp2)

{

double ans = (cp2.angle - cp1.angle) \* sqr(cir.r)

- cross(cir, cp1, cp2) + cross(Circle(0, 0), cp1, cp2);

return ans / 2;

}

void CirUnion(Circle cir[], int n)

{

Circle cp1, cp2;

sort(cir, cir + n, circmp);

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

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

if (dcmp(dis(cir[i], cir[j]) + cir[i].r - cir[j].r) <= 0)

cir[i].d++;

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

{

int tn = 0, cnt = 0;

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

{

if (i == j) continue;

if (CirCrossCir(cir[i], cir[i].r, cir[j], cir[j].r,

cp2, cp1) < 2) continue;

cp1.angle = atan2(cp1.y - cir[i].y, cp1.x - cir[i].x);

cp2.angle = atan2(cp2.y - cir[i].y, cp2.x - cir[i].x);

cp1.d = 1; tp[tn++] = cp1;

cp2.d = -1; tp[tn++] = cp2;

if (dcmp(cp1.angle - cp2.angle) > 0) cnt++;

}

tp[tn++] = Circle(cir[i].x - cir[i].r, cir[i].y, pi, -cnt);

tp[tn++] = Circle(cir[i].x - cir[i].r, cir[i].y, -pi, cnt);

sort(tp, tp + tn, cmp);

int p, s = cir[i].d + tp[0].d;

for (int j = 1; j < tn; ++j)

{

p = s; s += tp[j].d;

area[p] += calc(cir[i], tp[j - 1], tp[j]);

}

}

}

//最小圆覆盖

struct Point {

double x, y;

};

Point p[505];

double dist(Point A,Point B) {

return sqrt((A.x-B.x)\*(A.x-B.x)+(A.y-B.y)\*(A.y-B.y));

}

/\*\*\*返回三角形的外心 \*/

Point circumcenter(Point A,Point B,Point C) {

Point ret;

double a1 = B.x - A.x, b1 = B.y - A.y, c1 = (a1 \* a1 + b1 \* b1) / 2;

double a2 = C.x - A.x, b2 = C.y - A.y, c2 = (a2 \* a2 + b2 \* b2) / 2;

double d = a1 \* b2 - a2 \* b1;

ret.x = A.x + (c1 \* b2 - c2 \* b1) / d;

ret.y = A.y + (a1 \* c2 - a2 \* c1) / d;

return ret;

}

/\*\*\*c为圆心，r为半径 \*/

void min\_cover\_circle(Point \*p, int n, Point &c, double &r) {

random\_shuffle(p, p + n);

c=p[0]; r=0;

for(int i=1;i<n;i++) {

if(dist(p[i], c) > r + EPS) { //第一个点

c=p[i]; r=0;

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

if(dist(p[j], c) > r + EPS) { //第二个点

c.x = (p[i].x + p[j].x) / 2;

c.y = (p[i].y + p[j].y) / 2;

r = dist(p[j], c);

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

if(dist(p[k], c)> r + EPS) { //第三个 //求外接圆圆心，三点必不共线

c = circumcenter(p[i], p[j], p[k]);

r = dist(p[i], c);

}

}

}

}

}

int main() {

int n;

Point c;

double r;

while(~scanf("%d", &n) && n) {

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

scanf("%lf%lf", &p[i].x, &p[i].y);

min\_cover\_circle(p,n,c,r);

printf("%.2lf %.2lf %.2lf\n",c.x, c.y, r);

}

return 0;

}

//三维凸包

/\*

HDU 3662

求凸包表面多边形个数

\*/

const int MAXN=550;

const double eps=1e-8;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// \*是代码中的叉积

// ^是代码中的点积

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct CH3D {

struct face {

//表示凸包一个面上的三个点的编号

int a,b,c;

//表示该面是否属于最终凸包上的面

bool ok;

};

//初始顶点数

int n;

//初始顶点

Point P[MAXN];

//凸包表面的三角形数

int num;

//凸包表面的三角形

face F[8\*MAXN];

//凸包表面的三角形

int g[MAXN][MAXN];

//向量长度

double vlen(Point a)

{return sqrt(a.x\*a.x+a.y\*a.y+a.z\*a.z);

}

//叉乘

Point cross(const Point &a,const Point &b,const Point &c) {

return Point((b.y-a.y)\*(c.z-a.z)-(b.z-a.z)\*(c.y-a.y),

(b.z-a.z)\*(c.x-a.x)-(b.x-a.x)\*(c.z-a.z),

(b.x-a.x)\*(c.y-a.y)-(b.y-a.y)\*(c.x-a.x)

);

}

//三角形面积\*2

double area(Point a,Point b,Point c) {

return vlen((b-a)\*(c-a));

}

//四面体有向体积\*6

double volume(Point a,Point b,Point c,Point d) {

return (b-a)\*(c-a)^(d-a);

}

//正：点在面同向

double dblcmp(Point &p,face &f) {

Point m=P[f.b]-P[f.a];

Point n=P[f.c]-P[f.a];

Point t=p-P[f.a];

return (m\*n)^t;

}

void deal(int p,int a,int b) {

int f=g[a][b];//搜索与该边相邻的另一个平面

face add;

if(F[f].ok) {

if(dblcmp(P[p],F[f])>eps)

dfs(p,f);

else {

add.a=b;

add.b=a;

add.c=p;//这里注意顺序，要成右手系

add.ok=true;

g[p][b]=g[a][p]=g[b][a]=num;

F[num++]=add;

}

}

}

void dfs(int p,int now) {//递归搜索所有应该从凸包内删除的面

F[now].ok=0;

deal(p,F[now].b,F[now].a);

deal(p,F[now].c,F[now].b);

deal(p,F[now].a,F[now].c);

}

bool same(int s,int t) {

Point &a=P[F[s].a];

Point &b=P[F[s].b];

Point &c=P[F[s].c];

return fabs(volume(a,b,c,P[F[t].a]))<eps &&

fabs(volume(a,b,c,P[F[t].b]))<eps &&

fabs(volume(a,b,c,P[F[t].c]))<eps;

}

//构建三维凸包

void create() {//有用 {

int i,j,tmp;

face add;

num=0;

if(n<4)return;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//此段是为了保证前四个点不共面

bool flag=true;

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

if(vlen(P[0]-P[i])>eps) {

swap(P[1],P[i]);

flag=false;

break;

}

}

if(flag)return;

flag=true;

//使前三个点不共线

for(i=2;i<n;i++) {

if(vlen((P[0]-P[1])\*(P[1]-P[i]))>eps) {

swap(P[2],P[i]);

flag=false;

break;

}

}

if(flag)return;

flag=true;

//使前四个点不共面

for(int i=3;i<n;i++) {

if(fabs((P[0]-P[1])\*(P[1]-P[2])^(P[0]-P[i]))>eps) {

swap(P[3],P[i]);

flag=false;

break;

}

}

if(flag)return;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(i=0;i<4;i++) {

add.a=(i+1)%4;

add.b=(i+2)%4;

add.c=(i+3)%4;

add.ok=true;

if(dblcmp(P[i],add)>0)swap(add.b,add.c);

g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;

F[num++]=add;

}

for(i=4;i<n;i++) {

for(j=0;j<num;j++) {

if(F[j].ok&&dblcmp(P[i],F[j])>eps) {

dfs(i,j);

break;

}

}

}

tmp=num;

for(i=num=0;i<tmp;i++)

if(F[i].ok)

F[num++]=F[i];

}

//表面积

double area() {

double res=0;

if(n==3) {

Point p=cross(P[0],P[1],P[2]);

res=vlen(p)/2.0;

return res;

}

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

res+=area(P[F[i].a],P[F[i].b],P[F[i].c]);

return res/2.0;

}

double volume() {

double res=0;

Point tmp(0,0,0);

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

res+=volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);

return fabs(res/6.0);

}

//表面三角形个数

int triangle() {

return num;

}

//表面多边形个数

int polygon() {//计算多边形个数 {

int i,j,res,flag;

for(i=res=0;i<num;i++) {

flag=1;

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

if(same(i,j)) {

flag=0;

break;

}

res+=flag;

}

return res;

}

//三维凸包重心

Point barycenter() {

Point ans(0,0,0),o(0,0,0);

double all=0;

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

double vol=volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);

ans=ans+(o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0\*vol;

all+=vol;

}

ans=ans/all;

return ans;

}

//点到面的距离

double ptoface(Point p,int i) {

return fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p)

/vlen((P[F[i].b]-P[F[i].a])\*(P[F[i].c]-P[F[i].a])));

}

};

CH3D hull;

int main() {

while(scanf("%d",&hull.n)==1) {

for(int i=0;i<hull.n;i++) {

scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i].z);

}

hull.create();

printf("%d\n",hull.polygon());

}

return 0;

}

//最近点对

const double eps = 1e-8;

int dcmp(double x) {

if (fabs(x) < eps) return 0;

return x < 0 ? -1 : 1;

}

double Min(double x, double y) {

return dcmp(x - y) <= 0 ? x : y;

}

double dis(Point a, Point b) {

Vector u = a - b;

return sqrt(u \* u);

}

int arr[maxn];

Point p[maxn];

bool cmp(int a, int b) {

return p[a].y < p[b].y;

}

double close\_pair(int l, int r) {

// 判断两个点和三个点的情况

if(r == l + 1)

return dis(p[l], p[r]);

else if(r == l + 2)

return Min(dis(p[l],p[r]), Min(dis(p[l],p[l + 1]), dis(p[l + 1],p[r])));

int mid = (l + r) >> 1;

double ans = Min(close\_pair(l, mid), close\_pair(mid + 1,r));

int cnt = 0;

// 如果 当前p[i]点 横坐标位于 范围（中点横坐标-ans，中点横坐标+ans）位置内，则记录点的序号

for(int i = l; i <= r; i ++)

if (dcmp(p[i].x - p[mid].x + ans) >= 0 && dcmp(p[i].x - p[mid].x - ans) <= 0)

arr[cnt ++] = i;

// 按照纵坐标由小到大 对于arr数组内点进行排序

sort(arr, arr + cnt, cmp);

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

for(int j = i + 1; j < cnt; j ++) {

if(dcmp(p[arr[j]].y - p[arr[i]].y - ans) >= 0) break;

ans = Min(ans, dis(p[arr[i]],p[arr[j]]));

}

return ans;

}

//多边形面积交

const int maxn = 555;

const int maxisn = 10;

const double eps = 1e-8;

const double pi = acos(-1.0);

int dcmp(double x) {

if (fabs(x) < eps) return 0;

return x < 0 ? -1 : 1;

}

inline double sqr(double x) {return x \* x;}

struct Point {

double x, y;

Point () {}

Point (double x, double y) : x(x), y(y) {}

Point operator +(const Point& rhs) {

return Point(x + rhs.x, y + rhs.y);

}

Point operator -(const Point& rhs) {

return Point(x - rhs.x, y - rhs.y);

}

double operator \*(const Point& rhs) const {

return x \* rhs.x + y \* rhs.y;

}

double operator ^(const Point& rhs) const {

return x \* rhs.y - y \* rhs.x;

}

};

typedef Point Vector;

Point LineCross(Point a, Point b, Point c, Point d) {

double u = ((b - a) ^ (c - a)), v = ((a - b) ^ (d - b));

return Point((c.x \* v + d.x \* u) / (u + v), (c.y \* v + d.y \* u) / (u + v));

}

double PolygonArea(Point p[], int n) {

if (n < 3) return 0.0;

double s = p[0].y \* (p[n - 1].x - p[1].x);

p[n] = p[0];

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

s += p[i].y \* (p[i - 1].x - p[i + 1].x);

return fabs(s \* 0.5);

}

double CPIA(Point a[], Point b[], int na, int nb) {

Point p[maxisn], tmp[maxisn];

int tn, sflag, eflag;

a[na] = a[0], b[nb] = b[0];

memcpy(p, b, sizeof(Point) \* (nb + 1));

for (int i = 0; i < na && nb > 2; ++ i) {

sflag = dcmp((a[i + 1] - a[i]) ^ (p[0] - a[i]));

for (int j = tn = 0; j < nb; ++ j, sflag = eflag) {

if (sflag >= 0) tmp[tn ++] = p[j];

eflag = dcmp((a[i + 1] - a[i]) ^ (p[j + 1] - a[i]));

if ((sflag ^ eflag) == -2)

tmp[tn ++] = LineCross(a[i], a[i + 1], p[j], p[j + 1]);

}

memcpy(p, tmp, sizeof(Point) \* tn);

nb = tn, p[nb] = p[0];

}

if (nb < 3) return 0.0;

return PolygonArea(p, nb);

}

// a, b两个多边形 ，na a的点个数，nb 点个数 无论顺逆时针

//实际解法是两多边形面积和减去面积交，这个板子可以经过调整变成面积交

double SPIA(Point a[], Point b[], int na, int nb) {

Point t1[4], t2[4];

double res = 0, if\_clock\_t1, if\_clock\_t2;

a[na] = t1[0] = a[0], b[nb] = t2[0] = b[0];

for (int i = 2; i < na; i ++) {

t1[1] = a[i - 1], t1[2] = a[i];

if\_clock\_t1 = dcmp((t1[1] - t1[0]) ^ (t1[2] - t1[0]));

if (if\_clock\_t1 < 0) swap(t1[1], t1[2]);

for (int j = 2; j < nb; j ++) {

t2[1] = b[j - 1], t2[2] = b[j];

if\_clock\_t2 = dcmp((t2[1] - t2[0]) ^ (t2[2] - t2[0]));

if (if\_clock\_t2 < 0) swap(t2[1], t2[2]);

res += CPIA(t1, t2, 3, 3) \* if\_clock\_t1 \* if\_clock\_t2;

}

}

//return res; 面积交

return PolygonArea(a, na) + PolygonArea(b, nb) - res;

}

//多项式算法

#include <cmath>

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

typedef long long ll;

template<typename T>inline void read(T &x){

x=0;char ch;bool flag = false;

while(ch=getchar(),ch<'!');if(ch == '-') ch=getchar(),flag = true;

while(x=10\*x+ch-'0',ch=getchar(),ch>'!');if(flag) x=-x;

}

const int pri\_rt = 3;

const int maxn=600010;

const int mod=998244353;

const int inv\_2 = 499122177;

int n,k,N,C,len;

int rev[maxn],w[maxn];

int f[maxn];

int Inv[maxn],Ln[maxn],Exp[maxn],Sqrt[maxn];

inline int qpow(int x,int p){

int ret = 1;

for(;p;x=1LL\*x\*x%mod,p>>=1) if(p&1) ret=1LL\*ret\*x%mod;

return ret;

}

inline int check(int &x){

if(x < 0) x += mod;

if(x >= mod) x -= mod;

}

void FNT(int n,int \*x,int flag){

for(int i=0,t=0;i<n;++i){

if(i > t) swap(x[i],x[t]);

for(int j=n>>1;(t^=j) < j;j>>=1);

}

for(int m=2;m<=n;m<<=1){

int k = m>>1;

int wn = qpow(pri\_rt,flag == 1 ? (mod - 1)/m : (mod-1) - (mod-1)/m);

w[0] = 1;

for(int i=1;i<k;++i) w[i] = 1LL\*w[i-1]\*wn % mod;

for(int i=0;i<n;i+=m){

for(int j=0;j<k;++j){

int u = 1LL\*x[i+j+k]\*w[j] % mod;

x[i+j+k] = x[i+j] - u;check(x[i+j+k]);

x[i+j] = x[i+j] + u;check(x[i+j]);

}

}

}

if(flag == -1){

int inv = qpow(n,mod-2);

for(int i=0;i<n;++i) x[i] = 1LL\*x[i]\*inv%mod;

}

}

inline void get\_dao(int n,int \*f){

for(int i=0;i<n;++i) f[i] = 1LL\*f[i+1]\*(i+1) % mod;

f[n] = 0;

}

inline void get\_fen(int n,int \*f){

for(int i=n-1;i>=0;--i) f[i] = 1LL\*f[i-1]\*qpow(i,mod-2) % mod;

f[0] = 0;

}

void get\_inv(int n,int \*f){

static int g[maxn];

if(n == 1){

Inv[0] = qpow(f[0],mod-2);

return;

}

get\_inv((n+1)>>1,f);

int len = n<<1;

for(int i=0;i<n;++i) g[i] = f[i];

fill(g+n,g+len,0);

FNT(len,g,1);FNT(len,Inv,1);

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

Inv[i] = 1LL\*Inv[i]\*(2LL - 1LL\*g[i]\*Inv[i]%mod + mod) % mod;

}FNT(len,Inv,-1);fill(Inv+n,Inv+len,0);

}

void get\_ln(int n,int \*f){

int len = n<<1;

fill(Inv,Inv+(len<<1),0);

get\_inv(n,f);get\_dao(n,f);

FNT(len,f,1);FNT(len,Inv,1);

for(int i=0;i<len;++i) Ln[i] = 1LL\*f[i]\*Inv[i] % mod;

FNT(len,Ln,-1);fill(Ln+n,Ln+len,0);

get\_fen(n,Ln);

}

void get\_exp(int n,int \*f){

static int g[maxn];

if(n == 1){

Exp[0] = 1;

return;

}

get\_exp(n>>1,f);

int len = n<<1;

for(int i=0;i<n;++i) g[i] = Exp[i];

fill(g+n,g+len,0);

get\_ln(n,g);

for(int i=0;i<n;++i) Ln[i] = ((i == 0) - Ln[i] + f[i] + mod) % mod;

FNT(len,Ln,1);FNT(len,Exp,1);

for(int i=0;i<len;++i) Exp[i] = 1LL\*Exp[i]\*Ln[i] % mod;

FNT(len,Exp,-1);fill(Exp+n,Exp+len,0);

}

void get\_sqrt(int n,int \*f){

static int g[maxn];

if(n == 1){

Sqrt[0] = sqrt(f[0]);

return;

}

get\_sqrt(n>>1,f);

int len = n<<1;

fill(Inv,Inv+(len<<1),0);get\_inv(n,Sqrt);

for(int i=0;i<n;++i) g[i] = f[i];

fill(g+n,g+len,0);

FNT(len,g,1);FNT(len,Inv,1);

for(int i=0;i<len;++i) g[i] = 1LL\*g[i]\*Inv[i] % mod;

FNT(len,g,-1);

for(int i=0;i<n;++i) Sqrt[i] = 1LL\*(Sqrt[i] + g[i])\*inv\_2%mod;

}

void get\_pow(int len,int \*f,int p){

get\_ln(len,f);

for(int i=0;i<len;++i) f[i] = 1LL\*p\*Ln[i]%mod;

fill(Exp,Exp+(len<<1),0);

get\_exp(len,f);

}

int main(){

int n,k;read(n);read(k);

for(int i=0;i<n;++i) read(f[i]);

for(len=1;len<=n;len<<=1);

get\_sqrt(len,f);

fill(Inv,Inv+(len<<1),0);get\_inv(len,Sqrt);

for(int i=0;i<len;++i) f[i] = Inv[i];

get\_fen(len,f);fill(f+n,f+len,0);

get\_exp(len,f);

for(int i=0;i<len;++i) f[i] = Exp[i];

fill(Inv,Inv+(len<<1),0);get\_inv(len,f);

for(int i=0;i<len;++i) f[i] = Inv[i];

++f[0];

get\_ln(len,f);

for(int i=0;i<len;++i) f[i] = Ln[i];

++f[0];

fill(f+len+1,f+(len<<1),0);

get\_pow(len,f,k);

for(int i=1;i<n;++i) printf("%d ",1LL\*Exp[i]\*i % mod);

puts("0");

getchar();getchar();

return 0;

}

//单纯形

# include <cstdio>

# include <algorithm>

using namespace std;

# define FOR(i, a, b) for (int i = a; i <= b; ++ i)

# define REP(i, n) FOR (i, 1, n)

# define REP\_0N(i, n) FOR (i, 0, n)

# define NR 4005

typedef double ld;

const ld eps = 1e-8, inf = 1e9;

int n, m, type, id[NR], tp[NR]; ld a[NR][205];

void pivot (int r, int c) {

swap (id[r + n], id[c]);

ld t = -a[r][c]; a[r][c] = -1; REP\_0N (i, n) a[r][i] /= t;

REP\_0N (i, m) if (a[i][c] && r != i) { t = a[i][c]; a[i][c] = 0; REP\_0N (j, n) a[i][j] += t \* a[r][j]; }

}

void solve () {

ld t; REP (i, n) id[i] = i;

while (true) {

int i = 0, j = 0; ld \_w = -eps;

REP (k, m) if (a[k][0] < \_w) \_w = a[i = k][0]; if (!i) break;

REP (k, n) if (a[i][k] > eps) { j = k; break; } if (!j) { printf ("Infeasible"); return; }

pivot (i, j);

}

while (true) {

int i = 0, j = 0; ld \_w = eps;

REP (k, n) if (a[0][k] > \_w) \_w = a[0][j = k]; if (!j) break; \_w = inf;

REP (k, m) if (a[k][j] < -eps && (t = -a[k][0] / a[k][j]) < \_w) \_w = t, i = k; if (!i) { printf ("Unbounded"); return; }

pivot (i, j);

}

printf ("%.0f\n", a[0][0]);

FOR (i, n + 1, n + m) tp[id[i]] = i - n;

if (type) REP (i, n) printf ("%.9f ", tp[i] ? a[tp[i]][0] : 0);

}

int main(){

scanf("%d%d",&n,&m);

for(int i = 1;i <= n;i++){

scanf("%lf",&a[0][i]);

}

for(int i = 1;i <= m;i++){

int l,r;

scanf("%d%d",&l,&r);

scanf("%lf",&a[i][0]);

for(int j = l;j <= r;j++){

a[i][j] = -1;

}

}

solve();

return 0;

}

//NTT

const int MOD = 998244353;

typedef long long LL;

LL pow\_mod(LL a,LL b,LL p) {

LL ret = 1;

while (b) {

if (b & 1) ret = ret \* a % p;

a = a \* a % p;

b >>= 1;

}

return ret;

}

const int maxn = (1 << 18) + 5;

namespace NTT

{

const int ort = 3; // original root in MOD

// this code works only when b == 2

LL a[maxn],b[maxn],c[maxn];

void ntt(LL A[],int n,int inv) {

// inv == 1 : ntt, == -1 : intt

LL w = 1,d = pow\_mod(ort,(MOD - 1) / n,MOD),t;

int i,j,c,s;

if (inv == -1) {

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

std::swap(A[i],A[j]);

for (t = pow\_mod(n,MOD - 2,MOD),i = 0; i < n; ++ i)

A[i] = A[i] \* t % MOD;

}

for (s = n >> 1; s; s >>= w = 1,d = d \* d % MOD) {

for (c = 0; c < s; ++ c,w = w \* d % MOD) {

for (i = c; i < n; i += s << 1) {

A[i | s] = (A[i] + MOD - (t = A[i | s])) \* w % MOD;

A[i] = (A[i] + t) % MOD;

}

}

}

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

for (j = 0,s = i,c = n >> 1; c; c >>= 1,s >>= 1)

j = j << 1 | s & 1;

if (i < j) std::swap(A[i],A[j]);

}

}

void solve(LL \*A,LL \*B,LL \* C,int n){

int N = 1;

while(N <= (n << 1)) N <<= 1;

for(int i = n + 1;i < N;i++) A[i] = 0;

for(int i = n + 1;i < N;i++) B[i] = 0;

memcpy(b,B,sizeof(\*B) \* N);

memcpy(a,A,sizeof(\*A) \* N);

ntt(a,N,1);

ntt(b,N,1);

for(int i = 0;i < N;i++) c[i] = (LL) a[i] \* b[i] % MOD;

ntt(c,N,-1);

for(int i = 0;i < N;i++) C[i] = c[i];

}

}

//fastIO

namespace IO {

const int MT = 50 \* 1024 \* 1024; /// 50MB 请注意输入数据的大小！！！

char IO\_BUF[MT];

int IO\_PTR, IO\_SZ;

/// 要记得把这一行添加到main函数第一行！！！

void begin() {

IO\_PTR = 0;

IO\_SZ = fread (IO\_BUF, 1, MT, stdin);

}

template<typename T>

inline bool scan\_d (T & t) {

while (IO\_PTR < IO\_SZ && IO\_BUF[IO\_PTR] != '-' && (IO\_BUF[IO\_PTR] < '0' || IO\_BUF[IO\_PTR] > '9'))

IO\_PTR ++;

if (IO\_PTR >= IO\_SZ) return false;

bool sgn = false;

if (IO\_BUF[IO\_PTR] == '-') sgn = true, IO\_PTR ++;

for (t = 0; IO\_PTR < IO\_SZ && '0' <= IO\_BUF[IO\_PTR] && IO\_BUF[IO\_PTR] <= '9'; IO\_PTR ++)

t = t \* 10 + IO\_BUF[IO\_PTR] - '0';

if (sgn) t = -t;

return true;

}

inline bool scan\_s (char s[]) {

while (IO\_PTR < IO\_SZ && (IO\_BUF[IO\_PTR] == ' ' || IO\_BUF[IO\_PTR] == '\n') ) IO\_PTR ++;

if (IO\_PTR >= IO\_SZ) return false;

int len = 0;

while (IO\_PTR < IO\_SZ && IO\_BUF[IO\_PTR] != ' ' && IO\_BUF[IO\_PTR] != '\n')

s[len ++] = IO\_BUF[IO\_PTR], IO\_PTR ++;

s[len] = '\0';

return true;

}

template<typename T>

void print(T x) {

static char s[33], \*s1; s1 = s;

if (!x) \*s1++ = '0';

if (x < 0) putchar('-'), x = -x;

while(x) \*s1++ = (x % 10 + '0'), x /= 10;

while(s1-- != s) putchar(\*s1);

}

template<typename T>

void println(T x) {

print(x); putchar('\n');

}

};

//模拟退火

#include <iostream>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <math.h>

#define N 1005

#define eps 1e-8 //搜索停止条件阀值

#define INF 1e99

#define delta 0.98 //温度下降速度

#define T 100 //初始温度

using namespace std;

int dx[4] = {0, 0, -1, 1};

int dy[4] = {-1, 1, 0, 0}; //上下左右四个方向

struct Point

{

double x, y;

};

Point p[N];

double dist(Point A, Point B)

{

return sqrt((A.x - B.x) \* (A.x - B.x) + (A.y - B.y) \* (A.y - B.y));

}

double GetSum(Point p[], int n, Point t)

{

double ans = 0;

while(n--)

ans += dist(p[n], t);

return ans;

}

//其实我觉得这玩意儿根本不叫模拟退火

double Search(Point p[], int n)

{

Point s = p[0]; //随机初始化一个点开始搜索

double t = T; //初始化温度

double ans = INF; //初始答案值

while(t > eps)

{

bool flag = 1;

while(flag)

{

flag = 0;

for(int i = 0; i < 4; i++) //上下左右四个方向

{

Point z;

z.x = s.x + dx[i] \* t;

z.y = s.y + dy[i] \* t;

double tp = GetSum(p, n, z);

if(ans > tp)

{

ans = tp;

s = z;

flag = 1;

}

}

}

t \*= delta;

}

return ans;

}

int main()

{

int n;

while(scanf("%d", &n) != EOF)

{

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

scanf("%lf %lf", &p[i].x, &p[i].y);

printf("%.0lf\n", Search(p, n));

}

return 0;

}

//vim配置

syntax on

set cindent

set nu

set tabstop=4

set shiftwidth=4

set background=dark

map <C-A> ggVG"+y

map <F5> :call Run()<CR>

func! Run()

exec "w"

exec "!g++ -Wall % -o %<"

exec "!./%<"

endfunc

配置2

//syntax on

set cindent

set backspace=indent,eol,start

set nu

set tabstop=4

set shiftwidth=4

set background=dark

colo evening

inoremap {<CR> {<CR>}<ESC>O

imap ^H <Left><Del>

map <C-A> ggVG"+y

map <F5> :call CompileCpp()<CR>

func! CompileCpp()

exec "w"

exec "!g++ % -std=c++11 -Wall -o /tmp/a"

endfunc

map <F9> :call SaveInputData()<CR>

func! SaveInputData()

exec "tabnew"

exec 'normal "+gP'

exec "w! /tmp/input\_data"

endfunc

map <F6> :call ExecCpp()<CR>

func! ExecCpp()

exec "!/tmp/a < /tmp/input\_data"

endfunc