











题意：平面上有n个点集，现在把他们分成k个集合，使得每个集合中的每个点都至少有一个本集合的点之间的曼哈顿距离不大于X，求最小的X。

题目要求划分集合之后，每个集合的曼哈顿最小生成树的最长边不超过X，那么容易想到就是整个点集的曼哈顿最小生成树的第n-k条边。

那么怎么求出这个曼哈顿最小生成树呢

如果暴力加边再用Kruskal，边太多会超时。这里用一个算法来减少有效边的加入。

边权值为点间曼哈顿距离，那么每个点的有效加边选择应该是和他最近的4个象限方向的点。这里用一个树状数组维护以y-x为索引的y+x的值，然后这个数组所储存的就是一个点的第一象限方向的距离他最近的点。这样我们每次查找只要看（i，N）这个区间是否有一个点的距离比现在的更小（因为以y-x为索引，所以I>i就表示I这个点在i的第一象限方向）。最后每个方向的边都加完后，只要用Kruskal算法加边，加到第K就输出权值。

#define ll long long

const int N = 10000+5;

const int INIT = 1061109567;

using namespace std;

int n,k,cnt,Id[N],v[N],f[N]; //id为v值的id序号，v为储存以y-x为索引的y+x值

struct node{

int x,y,id;

friend bool operator < (node a,node b){

return a.x == b.x? a.y < b.y : a.x < b.x;

}

}p[N];

struct edge{

int u,v,w;

friend bool operator < (edge a,edge b) {

return a.w < b.w;

}

}e[N<<2];

int lowbit(int x){

return x&(-x);

}

void query(int id,int pos,int val){

pos += 1000;

int u = -1,ret = INT\_MAX;

for(int i = pos;i < N;i += lowbit(i)){

if(v[i] < ret && v[i] != INIT){ //找曼哈顿距离最短的

ret = v[i];

u = Id[i];

}

}

if(u != -1){ //找到这种点就加一个边

e[cnt].u = id;

e[cnt].v = u;

e[cnt++].w = ret - val;

}

}

void update(int id,int pos,int val){ //id,y-x,y+x

pos += 1000;

for(int i = pos;i > 0;i -= lowbit(i)){

if(val < v[i]){ //更换最佳选择

v[i] = val;

Id[i] = id;

}

}

}

int find(int x){

if(x == f[x]) return x;

else return f[x] = find(f[x]);

}

void kruskal(){

sort(e,e+cnt);

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

int num = n;

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

int x = find(e[i].u);

int y = find(e[i].v);

if(x != y){

f[x] = f[y];

num--;

if(num == k){

printf("%d\n",e[i].w);

return;

}

}

}

}

void addEdge(){

memset(v,63,sizeof(v)); //v == 1061109567

sort(p,p+n);

for(int i = n-1;i >= 0;i--){

int index = p[i].y - p[i].x;

int val = p[i].y + p[i].x;

query(p[i].id,index,val);

update(p[i].id,index,val);

}

}

int main(){

cnt = 0;

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

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

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

p[i].id = i;

}

addEdge();

for(int i = 0;i < n;i++) swap(p[i].x,p[i].y);

addEdge();

for(int i = 0;i < n;i++) p[i].y = -p[i].y;

addEdge();

for(int i = 0;i < n;i++) swap(p[i].x,p[i].y);

addEdge();

kruskal();

return 0;

}