

GGBeng

编程，首先要多敲，其次才是学！

博客园 新随笔 管理

随笔- 551 文章- 0 评论- 21

昵称: GGBeng
园龄: 2年3个月
粉丝: 166
关注: 9
[+加关注](#)

曼哈顿距离最小生成树

一、前人种树

博客: [曼哈顿距离最小生成树与莫队算法](#)

博客: [学习总结: 最小曼哈顿距离生成树](#)

二、知识梳理

曼哈顿距离: 给定二维平面上的 N 个点, 在两点之间连边的代价。 (即 $\text{distance}(P1, P2) = |x1 - x2| + |y1 - y2|$)

曼哈顿距离最小生成树问题求什么? 求使所有点连通的最小代价。

最小生成树的“环切”性质: 在图 $G = (V, E)$ 中, 如果存在一个环, 那么把环上的**最大边** e 删除后得到的图 $G' = (V, E - \{e\})$ 的最小生成树的边权和与 G 相同。

三、难点剖析

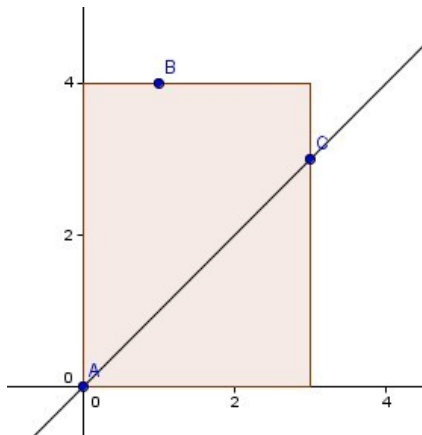
【废话定理神马的, 很难懂只要记住就是了】

朴素的算法可以用 $O(N^2)$ 的Prim, 或者处理出所有边做Kruskal, 但在这里总边数有 $O(N^2)$ 条, 所以Kruskal的复杂度变成了 $O(N^2 \log N)$ 。

但是事实上, 真正有用的边远没有 $O(N^2)$ 条。我们考虑每个点会和其他一些什么样的点连边。

可以得出这样一个结论: 以一个点为原点建立直角坐标系, 在每45度内只会**向距离该点最近的一个点连边**。

证明结论: 假设我们以点A为原点建系, 考虑在y轴向右45度区域内的任意两点 $B(x1, y1)$ 和 $C(x2, y2)$, 不妨设 $|AB| \leq |AC|$ (这里的距离为曼哈顿距离), 如下图:



$|AB| = x1 + y1$, $|AC| = x2 + y2$, $|BC| = |x1 - x2| + |y1 - y2|$ 。而由于B和C都在y轴向右45度的区域内, 有 $y - x > 0$ 且 $x > 0$ 。下面我们分情况讨论:

1. $x1 > x2$ 且 $y1 > y2$ 。这与 $|AB| \leq |AC|$ 矛盾;
2. $x1 \leq x2$ 且 $y1 > y2$ 。此时 $|BC| = x2 - x1 + y1 - y2$, $|AC| - |BC| = x2 + y2 - x2 + x1 - y1 + y2 = x1 - y1 + 2 * y2$ 。由前面各种关系可得 $y1 > y2 > x2 > x1$ 。假设 $|AC| < |BC|$ 即

2019年11月						
<	日	一	二	三	四	五
	27	28	29	30	31	1
	3	4	5	6	7	8
	10	11	12	13	14	15
	17	18	19	20	21	22
	24	25	26	27	28	29
	1	2	3	4	5	6

搜索

积分与排名

积分 - 113814
排名 - 4827

随笔分类

[C++ Primer习题\(20\)](#)
[C++学习\(81\)](#)
[Effective C++\(5\)](#)
[Java学习之路\(15\)](#)
[Linux入门\(32\)](#)
[Windows编程——MFC方式\(13\)](#)
[笔试题\(18\)](#)
[递归\(3\)](#)
[动态规划\(1\)](#)
[二叉树\(8\)](#)
[计算机网络\(6\)](#)
[计算机知识基础\(6\)](#)
[每天一笔\(28\)](#)
[排序\(6\)](#)
[散列表\(1\)](#)
[数据结构\(16\)](#)
[数据结构 \(基础篇\) \(20\)](#)
[数据库学习\(1\)](#)
[数据库语言-SQL\(16\)](#)
[思想聚焦\(21\)](#)
[图算法\(1\)](#)
[网络编程\(7\)](#)
[系统编程进阶\(2\)](#)
[系统编程入门\(15\)](#)
[系统编程习题\(3\)](#)

阅读排行榜

1. C++中substr函数的用法(62770)
2. C++STL——优先队列(38298)
3. 数据结构4——并查集 (入门) (24623)
4. C++STL——队列(13656)

$y_1 > 2 * y_2 + x_1$, 那么 $|AB| = x_1 + y_1 > 2 * x_1 + 2 * y_2$, $|AC| = x_2 + y_2 < 2 * y_2 < |AB|$ 与前提矛盾, 故 $|AC| \geq |BC|$;

3. $x_1 > x_2$ 且 $y_1 \leq y_2$. 与2同理;

4. $x_1 \leq x_2$ 且 $y_1 \leq y_2$. 此时显然有 $|AB| + |BC| = |AC|$, 即有 $|AC| > |BC|$ 。

综上有 $|AC| \geq |BC|$, 也即在这个区域内只需选择距离A最近的点向A连边。

这种连边方式可以保证边数是 $O(N)$ 的, 那么如果能高效处理出这些边, 就可以用 Kruskal 在 $O(N \log N)$ 的时间内解决问题。下面我们就考虑怎样高效处理边。

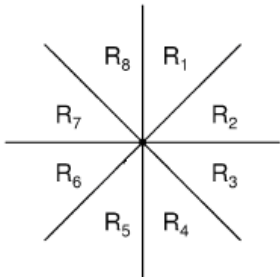
我们只需考虑在一块区域内的点, 其他区域内的点可以通过坐标变换“移动”到这个区域内。为了方便处理, 我们考虑在 y 轴向右45度的区域。在某个点 $A(x_0, y_0)$ 的这个区域内的点 $B(x_1, y_1)$ 满足 $x_1 \geq x_0$ 且 $y_1 - x_1 > y_0 - x_0$ 。这里对于边界我们只取一边, 但是操作中两边都取也无所谓。那么 $|AB| = y_1 - y_0 + x_1 - x_0 = (x_1 + y_1) - (x_0 + y_0)$ 。在A的区域内距离A最近的点也即满足条件的点中 $x + y$ 最小的点。因此我们可以将所有点按 x 坐标排序, 再按 $y - x$ 离散, 用线段树或者树状数组维护大于当前点的 $y - x$ 的最小的 $x + y$ 对应的点。时间复杂度 $O(N \log N)$ 。

至于坐标变换, 一个比较好处理的方法是第一次直接做; 第二次沿直线 $y = x$ 翻转, 即交换 x 和 y 坐标; 第三次沿直线 $x = 0$ 翻转, 即将 x 坐标取相反数; 第四次再沿直线 $y = x$ 翻转。注意只需要做4次, 因为边是双向的。

至此, 整个问题就可以在 $O(N \log N)$ 的复杂度内解决了。

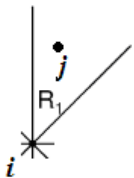
【回到正题】

一个点把平面分成了8个部分:



由上面的废话可知, 我们只需要让这个点与每个部分里距它最近的点连边。

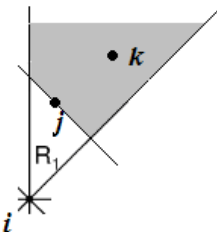
拿R1来说吧:



如图, i 的 R_1 区域里距 i 最近的点是 j 。也就是说, 其他点 k 都有:

$$x_j + y_j \leq x_k + y_k$$

那么 k 将落在如下阴影部分:



显然, 边 (i, j) , (j, k) , (i, k) 构成一个环 $\langle i, j, k \rangle$, 而 (i, k) 一定是最长边, 可以被删去。所以我们只连边 (i, j) 。

为了避免重复加边, 我们只考虑 $R_1 \sim R_4$ 这4个区域。(总共加了 $4N$ 条边)

这4个区域的点 (x, y) 要满足什么条件?

- 如果点 (x, y) 在 R_1 , 它要满足: $x \geq x_i, y - x \geq y_i - x_i$ (最近点的 $x + y$ 最小)
- 如果点 (x, y) 在 R_2 , 它要满足: $y \geq y_i, y - x \leq y_i - x_i$ (最近点的 $x + y$ 最小)

5. 制作一个简易计算器——基于Android Studio实现(11451)
6. 递归从入门到精通(11305)
7. 【转】C++后台开发之我见(8652)
8. 最短路径——SPFA算法(8310)
9. 并查集 (进阶) (4995)
10. 最小树形图——朱刘算法(4734)

- 如果点(x,y)在R3, 它要满足: $y \leq y_i, y + x \geq y_i + x_i$ (最近点的 $y - x$ 最小)
- 如果点(x,y)在R4, 它要满足: $x \geq x_i, y + x \leq y_i - x_i$ (最近点的 $y - x$ 最小)

其中一个条件用排序, 另一个条件用数据结构(这种方法很常用), 在数据结构上询问, 找最近点。因为询问总是前缀或后缀, 所以可以用树状数组。

四、代码模板

```

1 //离散化:
2 scanf("%d", &N);
3 for (int i=1; i<=N; ++i)
4 {
5     scanf("%d%d", &P[i].x, &P[i].y);
6     P[i].id = i;
7     P[i].d = P[i].y - P[i].x;
8     P[i].s = P[i].y + P[i].x;
9 }
10 //对x,y离散化
11 int totxy = 0;
12 for (int i=1; i<=N; ++i)
13 {
14     xy[totxy++] = P[i].x;
15     xy[totxy++] = P[i].y;
16 }
17 sort(xy, xy+totxy);
18 for (int i=1; i<=N; ++i)
19 {
20     P[i].idx = lower_bound(xy, xy+totxy, P[i].x) - xy + 1;
21     P[i].idy = lower_bound(xy, xy+totxy, P[i].y) - xy + 1;
22 }

1 //树状数组:
2 struct BIT
3 {
4     pii a[maxN * 2];
5     int N;
6     void Init(int _N)
7     {
8         N = _N;
9         for (int i=0; i<=N; ++i) a[i] = pii(oo, 0);
10    }
11    pii ask(int x)
12    {
13        return x == 0 ? pii(oo, 0) : min(a[x], ask(x - (x & (-x))));
14    }
15    void update(int x, const pii &v)
16    {
17        if (x > N) return ;
18        a[x] = min(a[x], v);
19        update(x + (x & (-x)), v);
20    }
21
22    pii ask_front(int x) {return ask(x);}
23    pii ask_back(int x) {return ask(N - x + 1);}
24    void update_front(int x, const pii &v) {update(x, v);}
25    void update_back(int x, const pii &v) {update(N - x + 1, v);}
26 } tree;

1 //构图:
2 bool cmp1(const Tpoint &A, const Tpoint &B)
3 {
4     //return A.x < B.x || (A.x == B.x && A.y < B.y);
5     return (A.y - A.x > B.y - B.x || A.y - A.x == B.y - B.x && A.x > B.x);
6 }
7 bool cmp2(const Tpoint &A, const Tpoint &B)

```

```

8  {
9      //return A.x < B.x || (A.x == B.x && A.y > B.y);
10     return (A.y + A.x < B.y + B.x || A.y + A.x == B.y + B.x && A.x > B.x);
11 }
12 bool cmp3(const Tpoint &A, const Tpoint &B)
13 {
14     //return A.y < B.y || (A.y == B.y && A.x < B.x);
15     return A.y - A.x < B.y - B.x || A.y - A.x == B.y - B.x && A.y > B.y;
16 }
17 bool cmp4(const Tpoint &A, const Tpoint &B)
18 {
19     //return A.y < B.y || (A.y == B.y && A.x > B.x);
20     return A.s > B.s || A.s == B.s && A.y < B.y;;
21 }
22
23 bool cmpE(const E_arr &A, const E_arr &B) {return A.v < B.v;}
24
25 void Make_Graph()
26 {
27     #define Connect(i,j) E[++tot_E].Init(P[i].id,P[j].id,getdis(i,j))
28
29     int LL, RR;
30
31     tree.Init(2 * N);
32     sort(P+1, P+N+1, cmp1);
33     for (int i=1; i<=N; ++i)
34     {
35         pii tmp = tree.ask_back(P[i].idx);
36         if (tmp.first < oo) Connect(i, tmp.second);
37         tree.update_back(P[i].idx, pii(P[i].x + P[i].y, i));
38     }
39
40     sort(P+1, P+N+1, cmp2);
41     tree.Init(2 * N);
42     for (int i=1; i<=N; ++i)
43     {
44         pii tmp = tree.ask_back(P[i].idx);
45         if (tmp.first < oo) Connect(i, tmp.second);
46         tree.update_back(P[i].idx, pii(P[i].x - P[i].y, i));
47     }
48
49     sort(P+1, P+N+1, cmp3);
50     tree.Init(2 * N);
51     for (int i=1; i<=N; ++i)
52     {
53         pii tmp = tree.ask_back(P[i].idy);
54         if (tmp.first < oo) Connect(i, tmp.second);
55         tree.update_back(P[i].idy, pii(P[i].x + P[i].y, i));
56     }
57
58     sort(P+1, P+N+1, cmp4);
59     tree.Init(2 * N);
60     for (int i=1; i<=N; ++i)
61     {
62         pii tmp = tree.ask_front(P[i].idy);
63         if (tmp.first < oo) Connect(i, tmp.second);
64         tree.update_front(P[i].idy, pii(P[i].x - P[i].y, i));
65     }
66 }

```

五、沙场练兵

[POJ 3241 Object Clustering 求曼哈顿距离最小生成树上第k大的边](#)

```
1 //POJ3241; Object Clustering; Manhattan Distance MST
```

```

2  #include <stdio>
3  #include <stdlib>
4  #include <algorithm>
5  #define N 100000
6  #define INFI 123456789
7
8  struct point
9  {
10     int x, y, n;
11     bool operator < (const point &p) const
12     { return x == p.x ? y < p.y : x < p.x; }
13 }p[N + 1];
14 struct inedged
15 {
16     int a, b, w;
17     bool operator < (const inedged &x) const
18     { return w < x.w; }
19 }e[N << 3 | 1];
20 struct BITnode
21 {
22     int w, p;
23 }arr[N + 1];
24 int n, k, tot = 0, f[N + 1], a[N + 1], *l[N + 1], ans;
25
26 template <typename T>
27 inline T abs(T x)
28 { return x < (T)0 ? -x : x; }
29
30 int find(int x)
31 { return x == f[x] ? x : f[x] = find(f[x]); }
32
33 inline bool cmp(int *a, int *b)
34 { return *a < *b; }
35
36 inline int query(int x)
37 {
38     int r = INFI, p = -1;
39     for (; x <= n; x += x & -x)
40         if (arr[x].w < r) r = arr[x].w, p = arr[x].p;
41     return p;
42 }
43
44 inline void modify(int x, int w, int p)
45 {
46     for (; x > 0; x -= x & -x)
47         if (arr[x].w > w) arr[x].w = w, arr[x].p = p;
48 }
49
50 inline void addedge(int a, int b, int w)
51 {
52     ++tot;
53     e[tot].a = a, e[tot].b = b, e[tot].w = w;
54     // printf("%d %d %d\n", a, b, w);
55 }
56
57 inline int dist(point &a, point &b)
58 { return abs(a.x - b.x) + abs(a.y - b.y); }
59
60 int main()
61 {
62     //Initialize
63     scanf("%d%d", &n, &k);
64     for (int i = 1; i <= n; ++i)
65     {
66         scanf("%d%d", &p[i].x, &p[i].y);
67         p[i].n = i;

```

```

68     }
69     //Solve
70     for (int dir = 1; dir <= 4; ++dir)
71     {
72         //Coordinate transform - reflect by y=x and reflect by x=0
73         if (dir == 2 || dir == 4)
74             for (int i = 1; i <= n; ++i) p[i].x ^= p[i].y ^= p[i].x ^= p[i].y;
75         else if (dir == 3)
76             for (int i = 1; i <= n; ++i) p[i].x = -p[i].x;
77         //Sort points according to x-coordinate
78         std::sort(p + 1, p + n + 1);
79         //Discretize
80         for (int i = 1; i <= n; ++i) a[i] = p[i].y - p[i].x, l[i] = &a[i];
81         std::sort(l + 1, l + n + 1, cmp);
82         /*
83         int cnt = 1;
84         for (int i = 2; i <= n; ++i)
85             if (*l[i] != *l[i - 1]) *l[i - 1] = cnt++;
86             else *l[i - 1] = cnt;
87         *l[n] = cnt;
88         */
89         for (int i = 1; i <= n; ++i) *l[i] = i;
90         //Initialize BIT
91         for (int i = 1; i <= n; ++i) arr[i].w = INFI, arr[i].p = -1;
92         //Find points and add edges
93         for (int i = n; i > 0; --i)
94         {
95             int pos = query(a[i]);
96             if (pos != -1)
97                 addedge(p[i].n, p[pos].n, dist(p[i], p[pos]));
98             modify(a[i], p[i].x + p[i].y, i);
99         }
100     }
101     //Kruskal
102     std::sort(e + 1, e + tot + 1);
103     for (int i = 1; i <= n; ++i) f[i] = i;
104     for (int i = 1, ec = n; ec > k && i <= tot; ++i)
105         if (find(e[i].a) != find(e[i].b))
106         {
107             f[find(e[i].a)] = find(e[i].b);
108             if (--ec == k) ans = e[i].w;
109         }
110     printf("%d\n", ans);
111     return 0;
112 }

```

好文要顶

关注我

收藏该文



GGBeng

关注 - 9

粉丝 - 166

+加关注

1

0

« 上一篇: [LCA \(最近公共祖先\) ——离线 Tarjan 算法](#)» 下一篇: [一般图匹配问题: 带花树](#)

posted @ 2017-07-26 00:21 GGBeng 阅读(2295) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问](#) 网站首页。

【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【活动】京东云服务器_云主机低于1折, 低价高性能产品备战双11

【培训】马士兵老师强势回归! Java线下课程全免费, 双十一大促!

【推荐】天翼云双十一翼降到底, 云主机11.11元起, 抽奖送大礼!

【推荐】流程自动化专家UiBot, 体系化教程成就高薪RPA工程师

【福利】个推四大热门移动开发SDK全部免费用一年，限时抢！
【推荐】阿里云双11冰点钜惠，热门产品低至一折等你来抢！

相关博文：

- [曼哈顿距离最小生成树与莫队算法（总结）](#)
 - [曼哈顿距离的最小生成树](#)
 - [\[题解\] poj 3241 Object Clustering \(kruskal曼哈顿距离最小生成树+树状数组\)](#)
 - [最小生成树模板加例题分析（最小生成树类型汇总）](#)
 - [POJ-3241 Object Clustering 曼哈顿最小生成树](#)
- » [更多推荐...](#)

最新 IT 新闻：

- [白宫警告重返月球计划需要更多资金](#)
 - [Pixel 4让人大失所望 谷歌犯了这么几个错误](#)
 - [三星内存生产设备污染发生在器兴工厂 专家：损失远超10亿韩元](#)
 - [专业软件的强制订阅让开源替代更有吸引力](#)
 - [尼安德特人可能死于现代人类带来的疾病](#)
- » [更多新闻...](#)