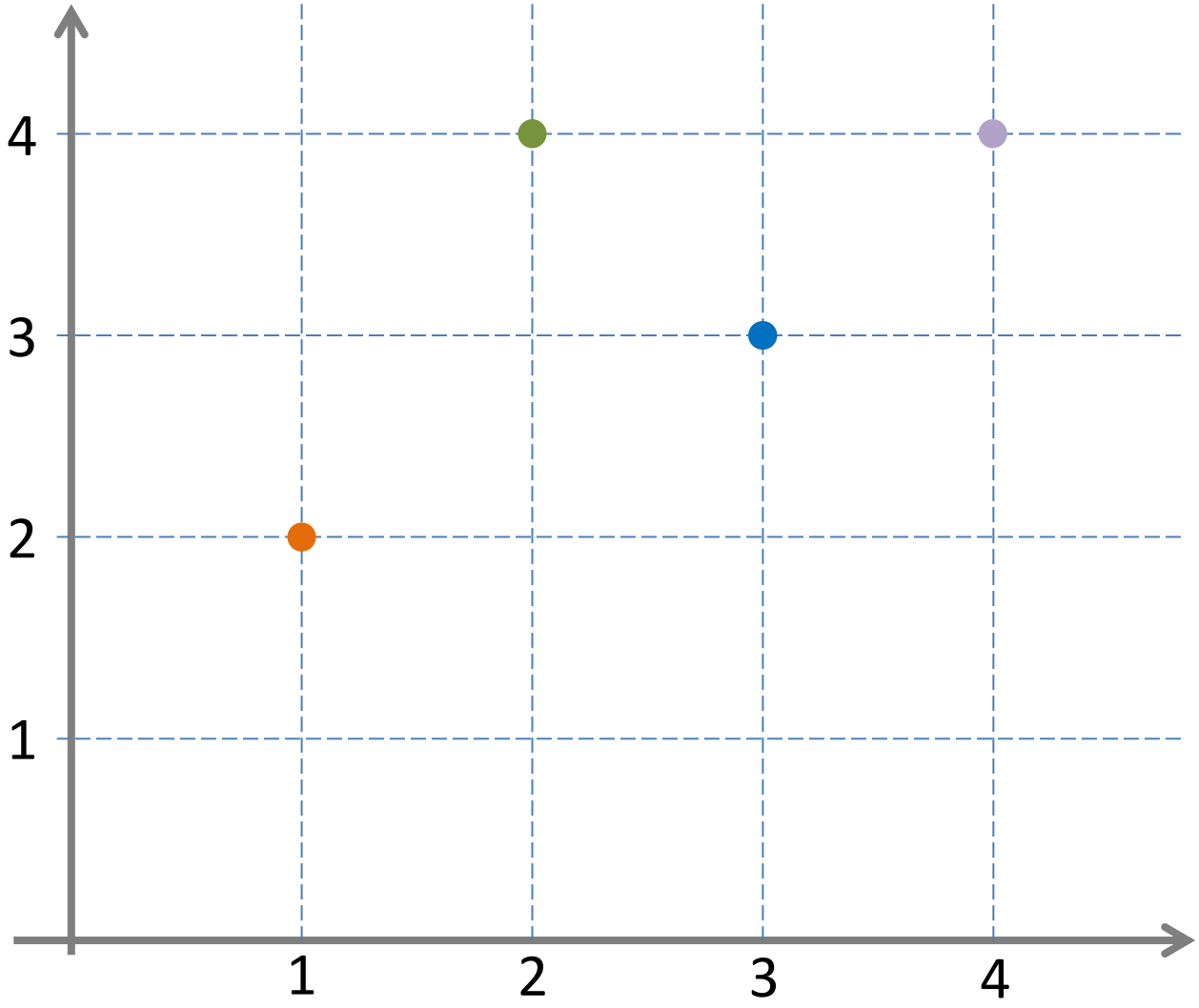
信奥算法

二维平面点

区间[1,r]
对应二维平面点(1,r)

[1,2],[2,4],
[3,3],[4,4].

并不是所有点
都对应区间

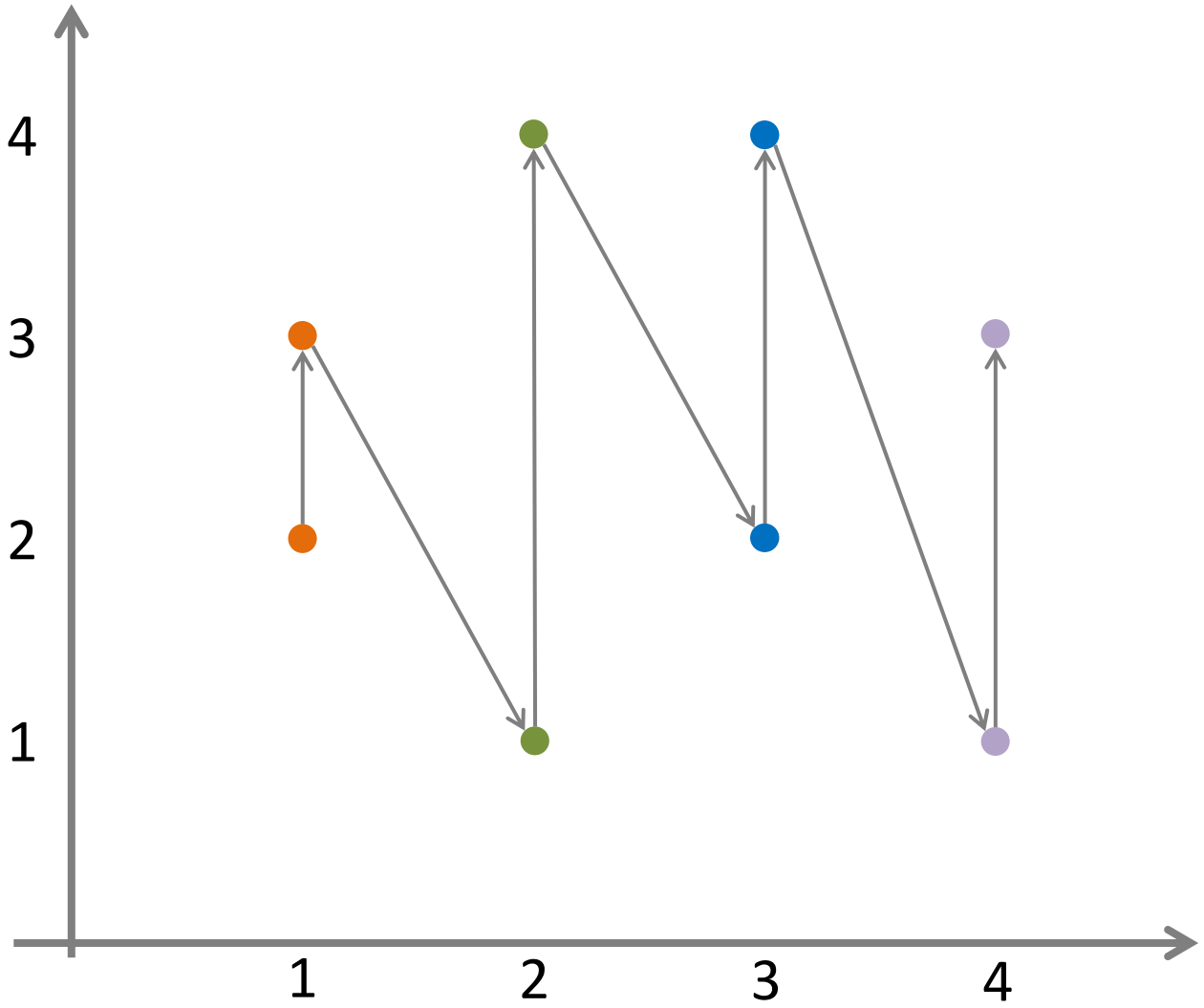


二维平面点排序

先比横坐标
后比纵坐标

从小到大

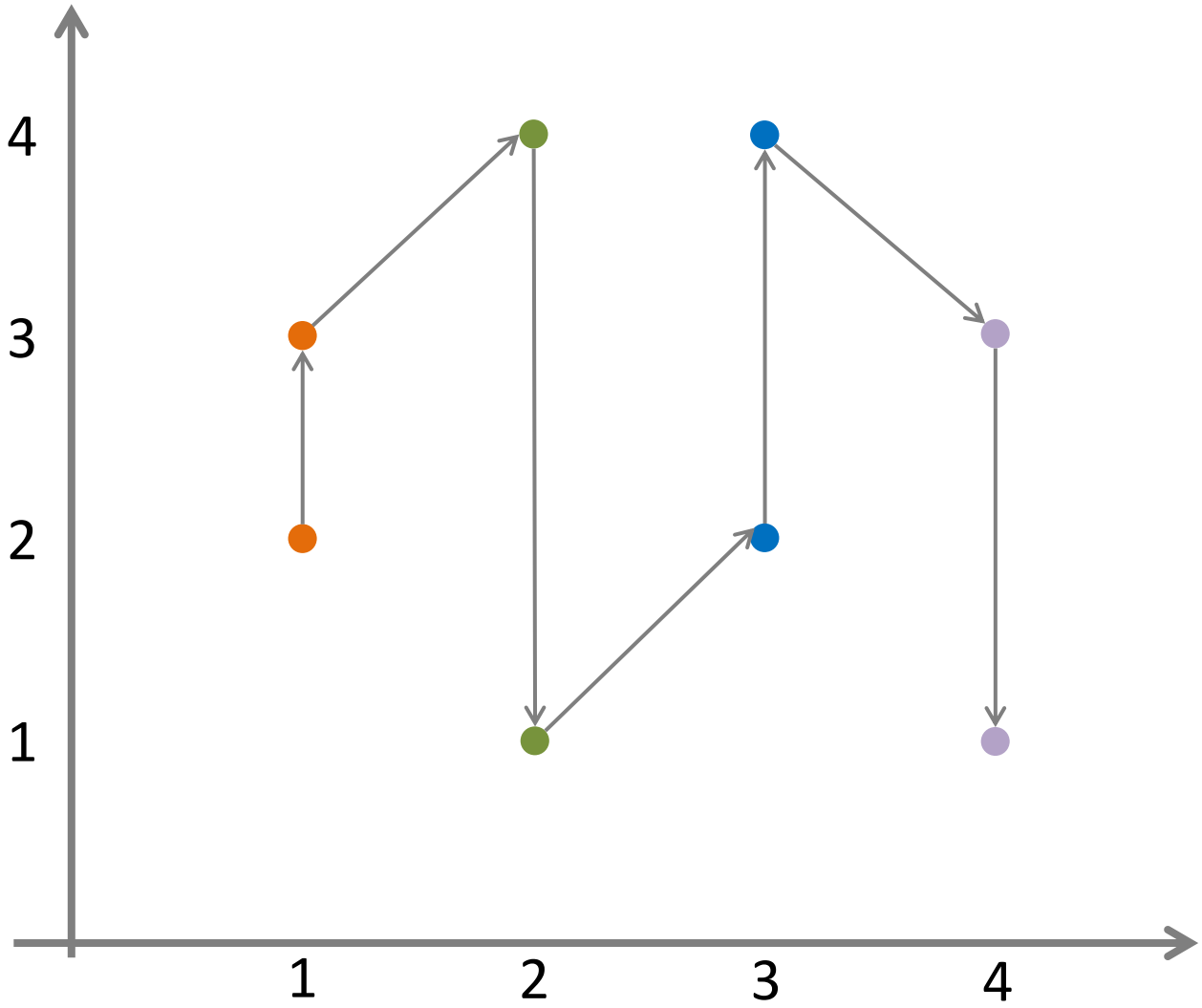
```
struct Point{int x,y;};
bool cmp(const Point&a,const Point&b){
        return a.x<b.x||a.x==b.x&&a.y<b.y;
}
```



先比横坐标
后比纵坐标

蛇形排序

```
bool cmp(const Point&a,const Point&b){
        if(a.x!=b.x) return a.x<b.x;
        if(a.x&1) return a.y<b.y;
        return a.y>b.y;
}
```



675

纯暴力解法

暴力

```
 9      cin>>n>>m>>k;
10      for(int i=1;i<=n;++i) cin>>x[i];
11      for(int i=1;i<=m;++i){
12          int l,r;
13          cin>>l>>r;
14          fill(cnt+1,cnt+1+k,0);
15          int cUnq=0;
16          for(int j=l;j<=r;++j){
17              ++cnt[x[j]];
18              if(cnt[x[j]]==1)
19                  cUnq++;
20          }
21          cout<<cUnq<<endl;
22      }
```

```
cUnq+=(++cnt[x[j]]==1);
```

暴力为什么慢?

增量式更新

增量式更新

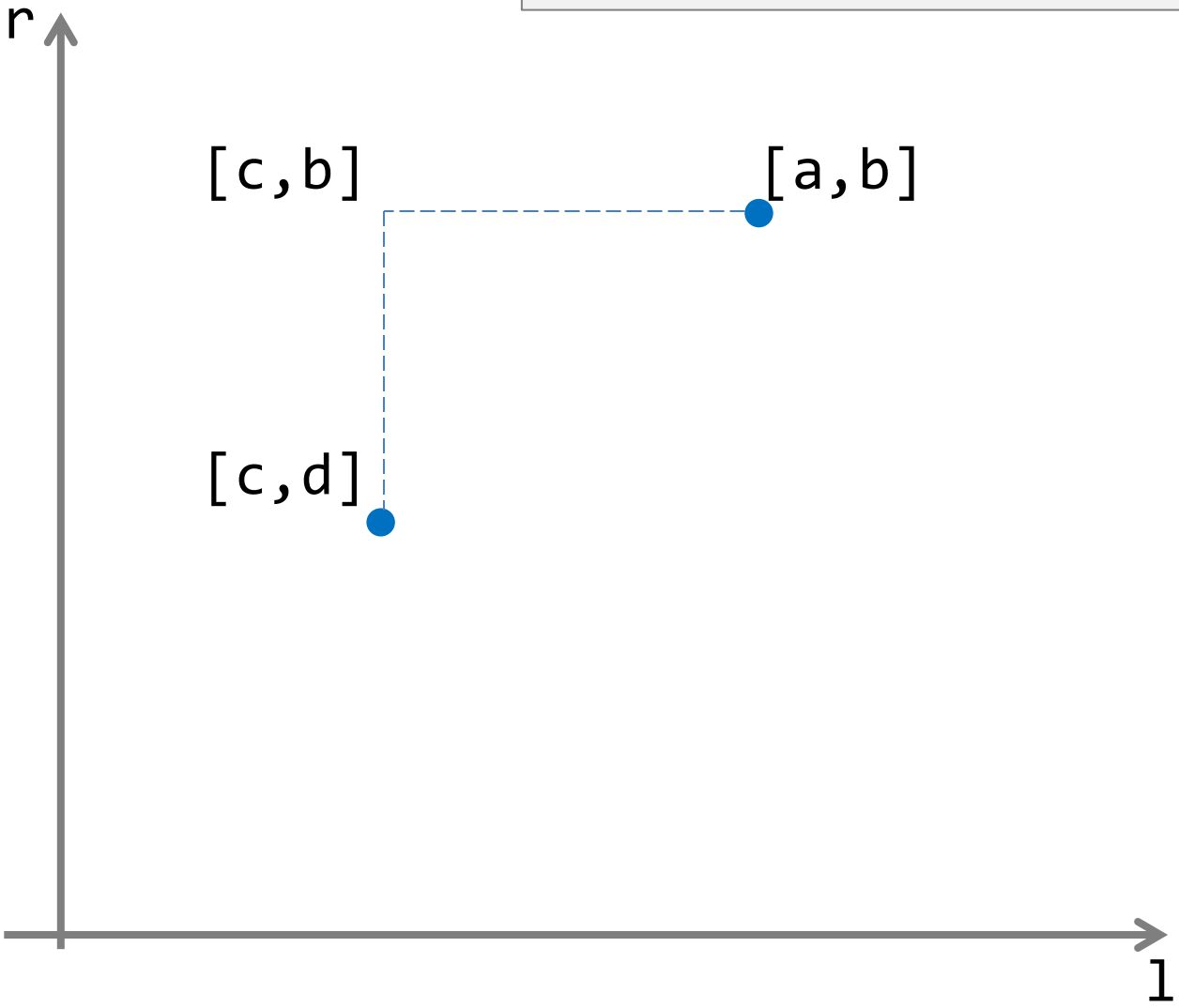
```
22      cin>>n>>m>>k;
23      for(int i=1;i<=n;++i) cin>>x[i];
24      for(int i=1;i<=m;++i) cin>>q[i].l>>q[i].r;

31      int l=0,r=0,cUnq=0;
32      for(int i=1;i<=m;++i){
33          while(l>q[i].l) cUnq+=(++cnt[x[--l]]==1);
34          while(l<q[i].l) cUnq-=(--cnt[x[l++]]==0);
35          while(r<q[i].r) cUnq+=(++cnt[x[++r]]==1);
36          while(r>q[i].r) cUnq-=(--cnt[x[r--]]==0);
37          ans[q[i].id]=cUnq;
38      }
```

增量式更新 曼哈顿距离

从区间[a,b]增量式渐变成区间[c,d]

计算量=|a-c|+|b-d|

区间对应二维点



增量式更新

```
22      cin>>n>>m>>k;
23      for(int i=1;i<=n;++i) cin>>x[i];
24      for(int i=1;i<=m;++i) cin>>q[i].l>>q[i].r;

31      int l=0,r=0,cUnq=0;
32      for(int i=1;i<=m;++i){
33          while(l>q[i].l) cUnq+=(++cnt[x[--l]]==1);
34          while(l<q[i].l) cUnq-=(--cnt[x[l++]]==0);
35          while(r<q[i].r) cUnq+=(++cnt[x[++r]]==1);
36          while(r>q[i].r) cUnq-=(--cnt[x[r--]]==0);
37          ans[q[i].id]=cUnq;
38      }
```

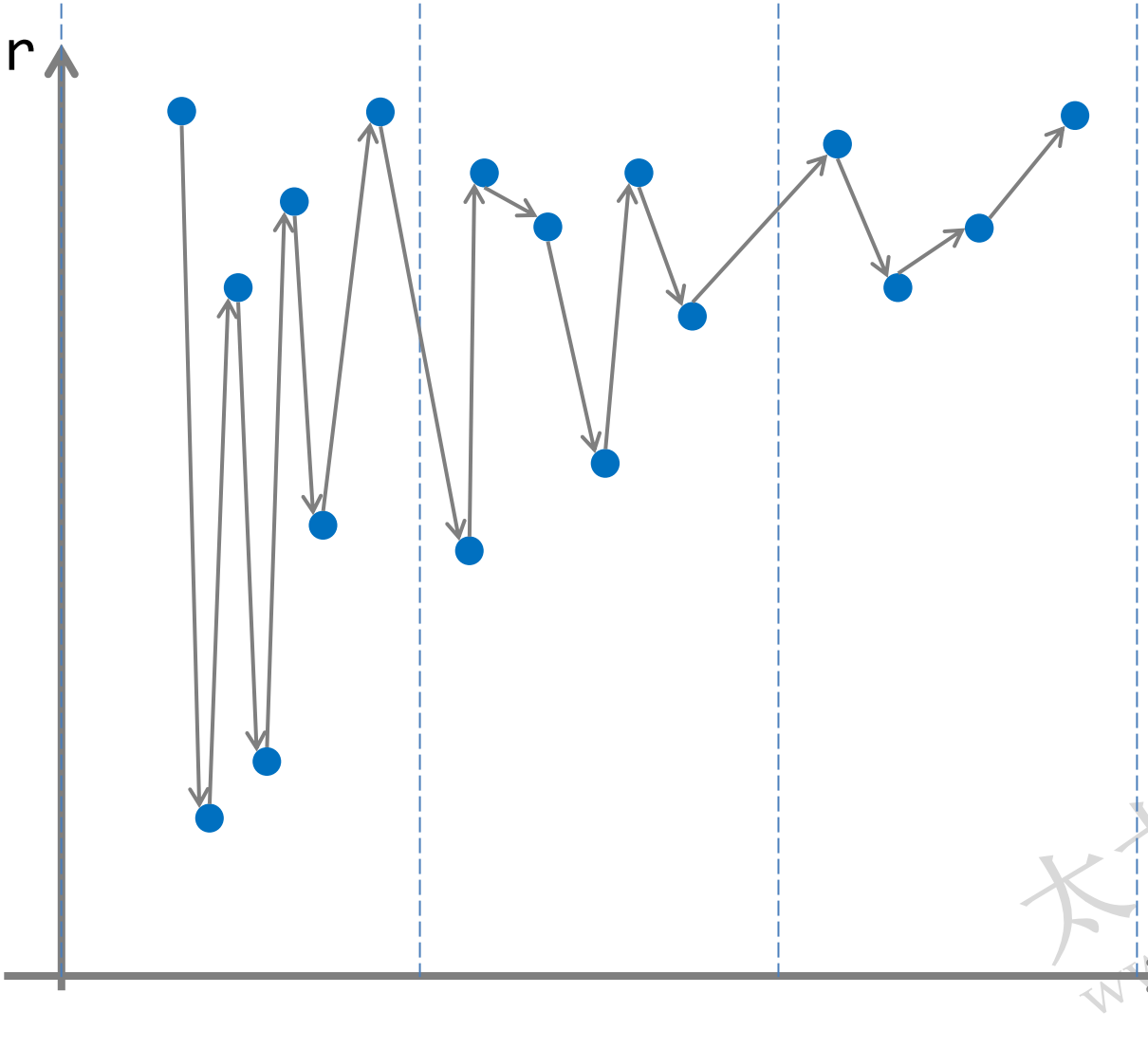
最差情况 时间复杂度O(mn)	请构造 最差情况

增量式更新

离线问询

区间排序

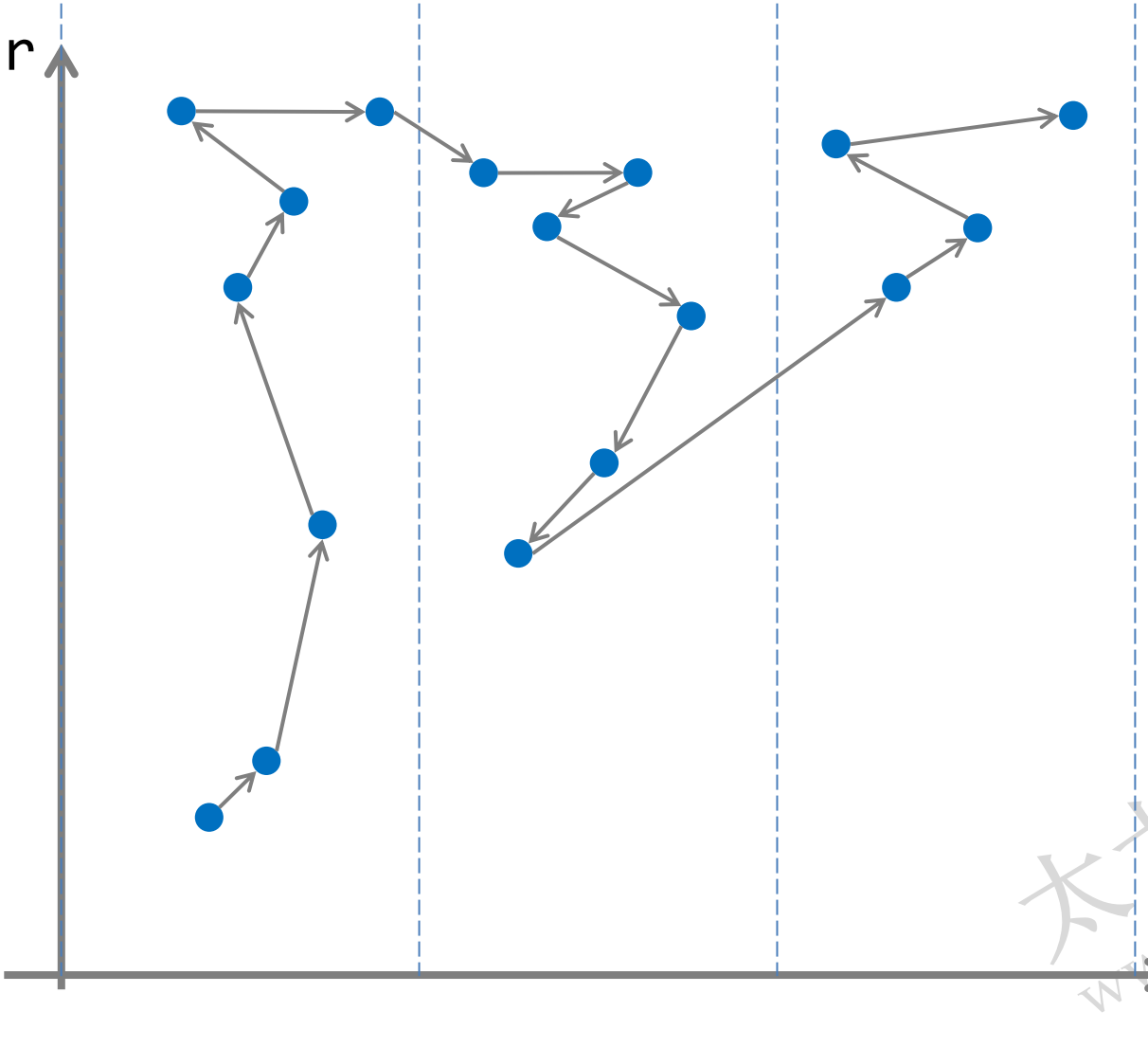
先比左端点
再比右端点



两点之间求
曼哈顿距离

增量式更新
离线问询
区间排序

左端点分块	先比左端点所在块号 块内再比右端点



两点之间求曼哈顿距离

增量式更新

离线问询

区间排序

左端点
分块

先比左端点所在块号
块内再比右端点

左端抖动
右端移动

增量式更新
离线问询
区间排序

左端点分块	先比左端点所在块号 块内再比右端点

```
 9  bool cmp(const Query&a,const Query&b){
10      if(a.blockID!=b.blockID) return a.blockID<b.blockID;
11      if(a.blockID&1) return a.r<b.r;
12      return a.r>b.r;
13  }
```

```
20      int L=n/sqrt(2*m);
21      for(int i=1;i<=m;++i){
22          q[i].id=i;
23          q[i].blockID=(q[i].l-1)/L+1;
24      }
25      sort(q+1,q+m+1,cmp);
```

最优分块大小

左端点块号

莫队算法

Mo’s algorithm

以上就是莫队算法,由莫涛提出,因其常做队长,人称莫队
主要是用分块思想+离线问询对区间排序
来解决不带修改的区间问题

排序规则	先比区间左端点所在块号 块内再比区间右端点

莫队算法

复杂度分析+最优分块大小

序列编号分组每组大小L,共n/L组

左端点

每两个问询间:左端点抖动距离不超过L
左端点抖动总距离不超过mL

右端点

1号块内:右端点移动距离不超过n
2号块内:右端点移动距离不超过n-L
……
右端点移动总距离不超过

$$n + (n - L) + (n - 2L) + \cdots \approx \frac{n^2}{2L}$$

总复杂度

$$mL + \frac{n^2}{2L} \geq n\sqrt{2m} = O(n\sqrt{m})$$

额外排序
O(mlogm)

等号条件

$$mL = \frac{n^2}{2L}, \qquad L^2 = \frac{n^2}{2m}, \qquad L = \frac{n}{\sqrt{2m}}$$

711

```
21      int L=
22      for(int i=1;i<=m;++i){
23          q[i].id=i;
24          q[i].blockID=
25      }
26      sort(q+1,q+m+1,cmp);
27      int l=0,r=0,cUnq=0;
28      for(int i=1;i<=m;++i){
29          while(l>q[i].l) 
30          while(l<q[i].l) 
31          while(r<q[i].r) cUnq+=(++cnt[x[++r]]==1);
32          while(r>q[i].r) cUnq-=(--cnt[x[r--]]==0);
33          ans[q[i].id]=
34      }
```

易错点

```
3 const int N=
4 const int M=
5 const int K=
6 int n,m,k,x[N],cnt[K],ans[M];
```

677

识别突破口：
k<=100

```
27      int l=0,r=0;
28      for(int i=1;i<=m;++i){
29          while(l>q[i].l)
30          while(l<q[i].l)
31          while(r<q[i].r)
32          while(r>q[i].r)
33          ans[q[i].id]=
34      }
```

总复杂度	$O(n\sqrt{m}+mk)$

太戈编程

675

711

677

拓展题	712