题解

A. Array's Hash

题意

给定一个长度为n的数组，这么定义该数组的哈希值：每次从数组开头取出两个数，将后一个数减去前一个数得到的数值放入数组开头，如此重复，直到数组中只剩下一个数，最后这个数便为数组的哈希值。现在每次操作把一段区间的数加上v要求输出每次操作后数组的哈希值

题解

显然数组的哈希值为$\sum_{i=1}^n{\left(-1\right)^\left(n-i\right)\times a_i}$

因此当区间左右端点奇偶性相同时对哈希值无贡献，奇偶性不同时如果区间左端点与n奇偶性相同则哈希值加v否则减v

时间复杂度$O\left(n\right)$

B. Bonuses on a Line

C. Manhattan Distance

题意

在直角坐标系中给定n整点$\left(-10^8\le x_i,y_i\le 10^8\right)$可以得到$\frac{n\times \left(n+1\right)}{2}$个点对，将所有点对的哈密顿距离排序，要求输出第k大的哈密顿距离$\left(2\le n\le 100000,1\le k\le \frac{n\times \left(n+1\right)}{2}\right)$

题解

大概思路为二分答案d统计哈密顿距离$\le d$的点对个数

首先，将坐标系顺时针旋转45度，放大$\sqrt{2}$倍，所以所有点坐标变为$\left(x-y,x+y\right)$与某个点哈密顿距离$\le d$转化为在以该点为中心的边与坐标轴平行且长度为$2d$的正方形中

考虑用滑动窗口+树状树组统计答案，具体过程见代码

代码

```
#include <iostream>
```

```cpp
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cstring>
#include <cctype>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
#define lowbit(x) (x)&(-x)
const int MAXN=1e5+5;
struct Node{
    int x,y;
    bool operator < (const Node &b)const{
        return x<b.x||(x==b.x&&y<b.y);
    }
}node[MAXN];
int n,m,c[MAXN],Y[MAXN];
LL k;
void add(int pos,int v){
    while(pos<=n){
        c[pos]+=v;
        pos+=lowbit(pos);
    }
}
int query(int pos){
    int ans=0;
    while(pos){
        ans+=c[pos];
        pos-=lowbit(pos);
    }
    return ans;
}
LL Count(int d){
    mem(c,0);
    LL ans=0;
    for(int i=1,j=1;i<=n;i++){
```

```
        while(j<i&&node[i].x-node[j].x>d){
            int pos=lower_bound(Y+1,Y+m,node[j].y)-Y;
            add(pos,-1);
            j++;
        }
        int pos1=upper_bound(Y+1,Y+m,node[i].y+d)-Y-1;
        int pos2=lower_bound(Y+1,Y+m,node[i].y-d)-Y-1;
        int pos3=lower_bound(Y+1,Y+m,node[i].y)-Y;
        ans+=query(pos1)-query(pos2);
        add(pos3,1);
    }
    return ans;
}
int main()
{
    n=read_int(),k=read_LL();
    int x,y;
    _rep(i,1,n){
        x=read_int(),y=read_int();
        node[i].x=x-y,node[i].y=x+y;
        Y[i]=x+y;
    }
    sort(node+1,node+n+1);
    sort(Y+1,Y+n+1);
    m=unique(Y+1,Y+n+1)-Y;
    int lef=1,rig=4e8,mid,ans=-1;
    while(lef<=rig){
        mid=lef+rig>>1;
        if(Count(mid)<k)
        lef=mid+1;
        else{
            ans=mid;
            rig=mid-1;
        }
    }
    cout<<ans;
    return 0;
}
```

D. Lexicographically Minimal Shortest Path

E. Fluctuations of Mana

F. Moving Target

G. Nuts and Bolts

H. Tree Painting

I. Sorting Colored Array

J. The Battle of Mages

K. Table

L. The Dragon Land

M. Notifications

签到题

总结