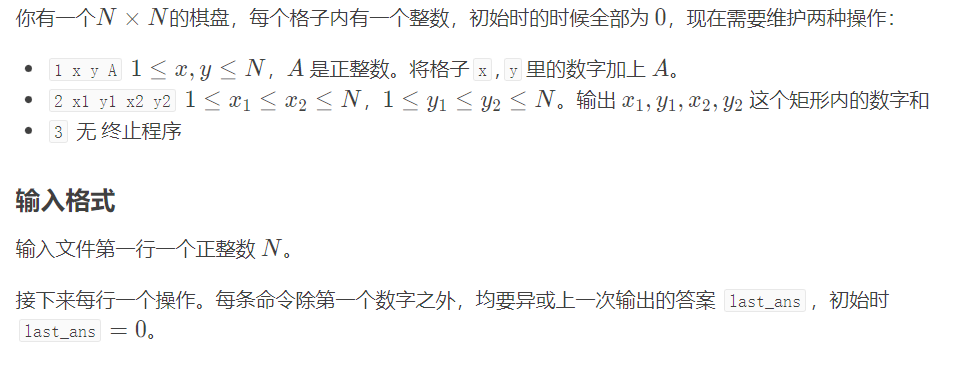
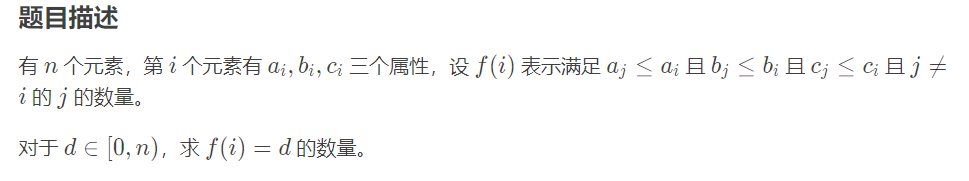
P4148 简单题



这个题目很有借鉴性，基本是三维偏序的模型（可以用kd树，cdq，树套树来写），

也就是有一种添加操作，拥有三种属性tj,xj,yj,(这里我们将操作的时间单独抽象出来作为一种属性)，而又有一种查询操作，拥有5个属性ti,x1i,x2i,y1i,y2i,查询有多少添加操作，其满足tj<ti,x1i<=xj<=x2i,y1i<=yj<=y2i,这种模型适用于一些是数据结构体，比如经典的cdq三维（陌上花开）如下，具体做法就是先按照某一个维度排序（比方说t），然后再套上面的这个板子



下面为第一道题的代码

#include<bits/stdc++.h>

using namespace std;

int read() {

int q=0,w=1;char ch=' ';

while(ch!='-'&&(ch<'0'||ch>'9')) ch=getchar();

if(ch=='-') w=-1,ch=getchar();

while(ch>='0'&&ch<='9') q=q\*10+ch-'0',ch=getchar();

return q\*w;

}

const int N=200005;

struct point{int x[2],w;}p[N];

struct node{int mi[2],mx[2],sum,ls,rs,sz;point tp;}tr[N];

int n,ans,rt,WD,top,cur,rub[N];

int operator < (point a,point b) {return a.x[WD]<b.x[WD];}

int newnode() {//建立一个新节点,rub:垃圾回收栈

if(top) return rub[top--];

else return ++cur;

}

void up(int k) {//提取子树最大/最小值,子树大小,权值和

int l=tr[k].ls,r=tr[k].rs;

for(int i=0;i<=1;++i) {

tr[k].mi[i]=tr[k].mx[i]=tr[k].tp.x[i];

if(l) tr[k].mi[i]=min(tr[k].mi[i],tr[l].mi[i]);

if(r) tr[k].mi[i]=min(tr[k].mi[i],tr[r].mi[i]);

if(l) tr[k].mx[i]=max(tr[k].mx[i],tr[l].mx[i]);

if(r) tr[k].mx[i]=max(tr[k].mx[i],tr[r].mx[i]);

}

tr[k].sum=tr[l].sum+tr[r].sum+tr[k].tp.w,tr[k].sz=tr[l].sz+tr[r].sz+1;

}

int build(int l,int r,int wd) {//重新建树

if(l>r) return 0;

int mid=(l+r)>>1,k=newnode();

WD=wd,nth\_element(p+l,p+mid,p+r+1),tr[k].tp=p[mid];

tr[k].ls=build(l,mid-1,wd^1),tr[k].rs=build(mid+1,r,wd^1);

up(k);return k;

}

void pia(int k,int num) {//把树还原成序列

if(tr[k].ls) pia(tr[k].ls,num);

p[tr[tr[k].ls].sz+num+1]=tr[k].tp,rub[++top]=k;

if(tr[k].rs) pia(tr[k].rs,num+tr[tr[k].ls].sz+1);

}

void check(int &k,int wd) {//检查树是否依然平衡,不平衡则重建

if(tr[k].sz\*0.75<tr[tr[k].ls].sz||tr[k].sz\*0.75<tr[tr[k].rs].sz)

pia(k,0),k=build(1,tr[k].sz,wd);

}

void ins(int &k,point tmp,int wd) {//插入

if(!k) {k=newnode(),tr[k].ls=tr[k].rs=0,tr[k].tp=tmp,up(k);return;}

if(tmp.x[wd]<=tr[k].tp.x[wd]) ins(tr[k].ls,tmp,wd^1);

else ins(tr[k].rs,tmp,wd^1);

up(k),check(k,wd);

}

int in(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2) {

return (X1>=x1&&X2<=x2&&Y1>=y1&&Y2<=y2);

}//检查是否在查询矩形内

int out(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2) {

return (x1>X2||x2<X1||y1>Y2||y2<Y1);

}//检查是否在查询矩形外

int query(int k,int x1,int y1,int x2,int y2) {

if(!k) return 0;

int re=0;

if(in(x1,y1,x2,y2,tr[k].mi[0],tr[k].mi[1],tr[k].mx[0],tr[k].mx[1])) return tr[k].sum;

if(out(x1,y1,x2,y2,tr[k].mi[0],tr[k].mi[1],tr[k].mx[0],tr[k].mx[1])) return 0;

if(in(x1,y1,x2,y2,tr[k].tp.x[0],tr[k].tp.x[1],tr[k].tp.x[0],tr[k].tp.x[1]))

re+=tr[k].tp.w;//对当前点做判断

re+=query(tr[k].ls,x1,y1,x2,y2)+query(tr[k].rs,x1,y1,x2,y2);

return re;//递归求解

}

int main() {

int bj,x1,y1,x2,y2;

n=read();

while("STO Cai Orz") {

bj=read(); if(bj==3) break;

if(bj==1) ins(rt,(point){read()^ans,read()^ans,read()^ans},0);

else {

x1=read()^ans,y1=read()^ans,x2=read()^ans,y2=read()^ans;

ans=query(rt,x1,y1,x2,y2),printf("%d\n",ans);

}

}

return 0;

}