線段樹模板

综述

线段树的原理：将[1,n]分解成若干特定的子区间(数量不超过4\*n),然后，将每个区间[L,R]都分解为少量特定的子区间，通过对这些少量子区间的修改或者统计，来实现快速对[L,R]的修改或者统计。

作用：对编号连续的一些点的区间信息进行修改或者统计操作

主要操作：区间查询、点更新、区间更新

时间复杂度：修改和统计的复杂度都是O(log(N))

由原理可以看出线段树维护的信息必须满足区间加法

如：

数字之和——总数字之和 = 左区间数字之和 + 右区间数字之和

最大公因数(GCD)——总GCD = gcd( 左区间GCD , 右区间GCD );

最大值——总最大值=max(左区间最大值，右区间最大值)

线段树原理的详细分析及应用可以参考一篇写得特别好的博文：线段树详解

这篇博客完全可以作为学习线段树的指南及训练线段树的参考。

模板

为了规范自己的写法，所以就整理一下模板。

以下模板ans[]存的是区间和，若存其他符合区间加法的信息，需要相应改代码。

(0)定义

const int MAXN=50010;

int a[MAXN],ans[MAXN<<2],lazy[MAXN<<2];

//a[]为原序列信息，ans[]模拟线段树维护区间和，lazy[]为懒惰标记

(1)更新结点信息

void PushUp(int rt)

{

ans[rt]=ans[rt<<1]+ans[rt<<1|1];

}

(2)建树

void Build(int l,int r,int rt)

{

if (l==r)

{

ans[rt]=a[l];

return;

}

int mid=(l+r)>>1;

Build(l,mid,rt<<1);

Build(mid+1,r,rt<<1|1);

PushUp(rt);

}

(3) 下推懒惰标记

void PushDown(int rt,int ln,int rn)//ln表示左子树元素结点个数，rn表示右子树结点个数

{

if (lazy[rt])

{

lazy[rt<<1]+=lazy[rt];

lazy[rt<<1|1]+=lazy[rt];

ans[rt<<1]+=lazy[rt]\*ln;

ans[rt<<1|1]+=lazy[rt]\*rn;

lazy[rt]=0;

}

}

(4)点更新

void Add(int L,int C,int l,int r,int rt)

{

if (l==r)

{

ans[rt]+=C;

return;

}

int mid=(l+r)>>1;

//PushDown(rt,mid-l+1,r-mid); 若既有点更新又有区间更新，需要这句话

if (L<=mid)

Add(L,C,l,mid,rt<<1);

else

Add(L,C,mid+1,r,rt<<1|1);

PushUp(rt);

}

(5)区间更新

void Update(int L,int R,int C,int l,int r,int rt)

{

if (L<=l&&r<=R)

{

ans[rt]+=C\*(r-l+1);

lazy[rt]+=C;

return;

}

int mid=(l+r)>>1;

PushDown(rt,mid-l+1,r-mid);

if (L<=mid) Update(L,R,C,l,mid,rt<<1);

if (R>mid) Update(L,R,C,mid+1,r,rt<<1|1);

PushUp(rt);

}

(6)区间查询

LL Query(int L,int R,int l,int r,int rt)

{

if (L<=l&&r<=R)

return ans[rt];

int mid=(l+r)>>1;

PushDown(rt,mid-l+1,r-mid);//若更新只有点更新，不需要这句

LL ANS=0;

if (L<=mid) ANS+=Query(L,R,l,mid,rt<<1);

if (R>mid) ANS+=Query(L,R,mid+1,r,rt<<1|1);

return ANS;

}

(7)调用函数

//建树

Build(1,n,1);

//点更新

Add(L,C,1,n,1);

//区间修改

Update(L,R,C,1,n,1);

//区间查询

int ANS=Query(L,R,1,n,1);

注：若只涉及点更新的题，只需用(1)(2)(4)(6)

若只涉及区间更新的题，需用(1)(2)(3)(5)(6)

若为两种更新都有，则在所有向子区间查询或更新前，都需PushDown()