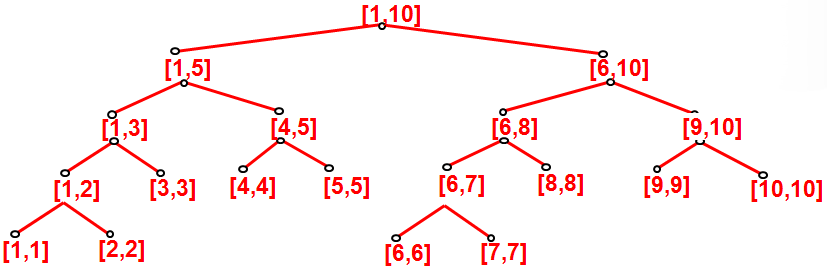
# 线段树

## **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](http://blog.csdn.net/zearot/article/details/48299459" \t "https://blog.csdn.net/yyt330507870/article/details/_blank)   
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sum[]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],sum[MAXN<<2],lazy[MAXN<<2];//a[]为原序列信息，sum[]模拟线段树维护区间和，lazy[]为懒惰标记

****(1)更新结点信息****

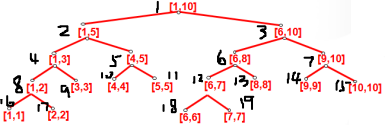
void PushUp(int rt){//子树上推

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

}

****(2)建树****

在线段树中建树一般采用的是递归的方式建树,在建树的过程中保留每个点的信息(区间左端点,右端点,区间和等),在线段树中的每个节点就是一个区间







void Build(int l,int r,int rt){//建树，参数：节点rt、区间[l,r]

if (l==r){//叶子节点

sum[rt]=a[l];

return;

}

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

Build(l,mid,rt<<1);//建左子树

Build(mid+1,r,rt<<1|1);//建右子树

PushUp(rt);//更新树根

}

****(3) 下推懒惰标记****

azy[]数组,懒惰标记.

lazy数组用于保存修改在某个区间上的值,但不即时修改赋值到节点上,而是在之后查询时需要查询它子树的值的时候再将懒惰标记下移.

那么下移lazy标记就是将当前区间的修改细化到每个小区间上。

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];////左右子树加上上面节点的标记

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

sum[rt<<1]+=lazy[rt]\*ln;//将区间和加上子树的个数乘标记的大小

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

lazy[rt]=0; //消去节点的懒惰标记

}

}

****(4)点更新****

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

if (l==r){ //叶子节点

sum[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);//更新根

}

****(5)区间更新****

修改时是寻找一个能全部包含于修改范围的区间,并将它打上lazy标记.

一个需要修改的范围可以看做是一个个小的范围的集合,如:

1到8的区间可以看做是[1,5]和[6,8];5到6的区间可以看做是[5,5]和[6,6];

如果是将1~8加5,则将区间[1,5]和区间[6,8]节点的懒惰标记加上5就可以

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

if (L<=l&&r<=R){ //修改完全被包含的区间

sum[rt]+=C\*(r-l+1); //修改区间(子树rt)信息

lazy[rt]+=C;

return;

}

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

PushDown(rt,mid-l+1,r-mid); //根节点的信息，存入左右子树

if (L<=mid) Update(L,R,C,l,mid,rt<<1);//更新左子树

if (R>mid) Update(L,R,C,mid+1,r,rt<<1|1);//更新右子树

PushUp(rt);//更新根

}

****(6)区间查询****

查询和修改操作比较像,也是将一个大区间细化为一个个小区间,找能被完全包含的区间进行查询

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

if (L<=l&&r<=R)//完全被包含的区间直接返回区间和

return sum[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);//有在左子树

if (R>mid) ANS+=Query(L,R,mid+1,r,rt<<1|1);//有在右子树

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()

# 线段树模板题

1. 1.HDU 1754 (点修改)

#include <cstdio>

#include <iostream>

#include <cstring>

#include <algorithm>

#define ll long long

#define mod 1000000007

#define mem(a) memset(a,0,sizeof(a))

using namespace std;

const int maxn = 200000 + 5 , inf = 0x3f3f3f3f ;

int sum[maxn\*4],A[maxn];

void PushUp(int rt){sum[rt] = max(sum[rt\*2],sum[rt\*2+1]);}

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

if(l==r){

sum[rt]=A[l];

return;

}

int m = (l+r)/2;

Build(l,m,rt\*2);

Build(m+1,r,rt\*2+1);

PushUp(rt);

}

void Updata(int L,int c,int l,int r,int rt){

if(r==l){

sum[rt] = c;

return;

}

int m = (l+r)/2;

if(L<=m) Updata(L,c,l,m,rt\*2);

else Updata(L,c,m+1,r,rt\*2+1);

PushUp(rt);

}

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

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

return sum[rt];

}

int m = (l+r)/2;

int ans = 0;

if(L<=m) ans = max(ans,Query(L,R,l,m,rt\*2));

if(R>m ) ans = max(ans,Query(L,R,m+1,r,rt\*2+1));

return ans;

}

int main(){

char c;

int n,m,a,b;

while(scanf("%d %d",&n,&m)!=EOF){

mem(sum);mem(A);

for(int i=1;i<=n;i++)scanf("%d",&A[i]);

Build(1,n,1);

while(m--){

scanf(" %c",&c);

if(c=='Q'){

scanf("%d %d",&a,&b);

printf("%d\n",Query(a,b,1,n,1));

}else{

scanf("%d %d",&a,&b);

Updata(a,b,1,n,1);

}

}

}

return 0;

}

2.POJ 3468 （区间修改）

#include <cstdio>

#include <iostream>

#include <cstring>

#include <algorithm>

#define ll long long

#define mod 1000000007

#define mem(a) memset(a,0,sizeof(a))

using namespace std;

const int maxn = 100000 + 7 , inf = 0x3f3f3f3f ;

ll sum[maxn<<2],add[maxn<<2];

ll A[maxn];

int n,sum\_cmd;

void Pushup(int rt){sum[rt]=sum[rt<<1]+sum[rt<<1|1];}

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

if(l==r){

sum[rt] = A[l];

return;

}

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

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

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

Pushup(rt);

}

void PushDown(int rt,int ln,int rn){

if(add[rt]){

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

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

sum[rt<<1]+=add[rt]\*ln;

sum[rt<<1|1]+=add[rt]\*rn;

add[rt] = 0;

}

}

void Updata(int L,int R,int C,int l,int r,int rt){

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

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

add[rt]+=C;

return;

}

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

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

if(L<=m) Updata(L,R,C,l,m,rt<<1);

if(R> m) Updata(L,R,C,m+1,r,rt<<1|1);

Pushup(rt);

}

ll Query(int L,int R,int l,int r,int rt){

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

return sum[rt];

}

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

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

ll ans = 0;

if(L<=m) ans+=Query(L,R,l,m,rt<<1);

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

return ans;

}

int main(){

char cmd;

ll a,b,c;

while(~scanf("%d %d",&n,&sum\_cmd)){

mem(A);mem(sum);mem(add);

for(int i = 1 ; i <= n ; i ++ )scanf("%lld",&A[i]);

Build(1,n,1);

while(sum\_cmd--){

scanf(" %c",&cmd);

if(cmd=='Q'){

scanf("%lld %lld",&a,&b);

printf("%lld\n",Query(a,b,1,n,1));

}else if(cmd=='C'){

scanf("%lld %lld %lld",&a,&b,&c);

Updata(a,b,c,1,n,1);

}

}

}

return 0;

}

3.HDU 1754（区间求最大值）

#include <cstdio>

#include <iostream>

#include <cstring>

#include <algorithm>

#define ll long long

#define mod 1000000007

#define mem(a) memset(a,0,sizeof(a))

using namespace std;

const int maxn = 200000 + 5 , inf = 0x3f3f3f3f ;

int sum[maxn\*4],A[maxn];

void PushUp(int rt){sum[rt] = max(sum[rt\*2],sum[rt\*2+1]);}

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

if(l==r){

sum[rt]=A[l];

return;

}

int m = (l+r)/2;

Build(l,m,rt\*2);

Build(m+1,r,rt\*2+1);

PushUp(rt);

}

void Updata(int L,int c,int l,int r,int rt){

if(r==l){

sum[rt] = c;

return;

}

int m = (l+r)/2;

if(L<=m) Updata(L,c,l,m,rt\*2);

else Updata(L,c,m+1,r,rt\*2+1);

PushUp(rt);

}

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

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

return sum[rt];

}

int m = (l+r)/2;

int ans = 0;

if(L<=m) ans = max(ans,Query(L,R,l,m,rt\*2));

if(R>m ) ans = max(ans,Query(L,R,m+1,r,rt\*2+1));

return ans;

}

int main(){

char c;

int n,m,a,b;

while(scanf("%d %d",&n,&m)!=EOF){

mem(sum);mem(A);

for(int i=1;i<=n;i++)scanf("%d",&A[i]);

Build(1,n,1);

while(m--){

scanf(" %c",&c);

if(c=='Q'){

scanf("%d %d",&a,&b);

printf("%d\n",Query(a,b,1,n,1));

}else{

scanf("%d %d",&a,&b);

Updata(a,b,1,n,1);

}

}

}

return 0;

}