原创 HDU 4578（Transformation）线段树区间更新及查询

2019-09-10 16:10:26 Laaahu_ 阅读数 26 更多

题目： 客官进来看看啊
题意：
四种操作：
①对给定区间的所有值加上 一个数 c;
②对给定区间的所有数乘上一个数 c;
③将给定区间的所有数变为 c ;
④输出一个区间的所有数的和，平方和，立方和 值 mod 10007。
这道题第一眼就知道是一个线段树的题目， 这个很容易就可以看出来，最主要的是怎么将这些操作联系起来。
思路：
因为有三种更新操作，所以需要三个 懒惰标记 ， 分别表示 加 ， 乘 ， 覆盖。
用三个值去表示 这个区间的 和 ， 平方和，立方和；
len 表示区间长度。
加一个数 c ：
立方和 : sum [3] = sum [3]+3 x c^2 x sum[1]+3 x c x sum[2] + len x c^3;
平方和 : sum [2] = sum [2]+2 x sum [1] x c +len x c^2;
和 ： sum[1] = sum[1] + len*c;
式子中的每个 sum 都是之前的sum 所以一定要按照 3，2，1 的顺序更新。
乘一个数 c ：
sum[1] = c x sum[1];
sum[2] =c^2x sum[2];
sum[3] = c^3x sum[3];
覆盖 ； 就是让所有值直接改变，这个比较简单。
因为要取模 所以每次只要做运算就对他的结果进行一次取模 ， 让结果始终在 int 范围内。
细节部分看代码，并自己推一下公式啊（利用简单的平方和，立方和公式）。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
const int maxn = 1e5+10;
const int mod = 10007;
struct node{
    int b,e,v[5];
    int lazy[5];
}t[maxn<<2];
void create(int bb,int ee,int node){
    t[node].b=bb;
    t[node].e=ee;
    t[node].v[1]=0,t[node].v[2]=0,t[node].v[3]=0;
    t[node].lazy[1]=0,t[node].lazy[2]=1,t[node].lazy[3]=0;
    if(bb==ee)return ;
    int mid = (bb+ee)>>1;
    create(bb,mid,node<<1);
    create(mid+1,ee,node<<1|1);
}
void push_up(int node){
    t[node].v[1]=(t[node<<1].v[1]+t[node<<1|1].v[1])%mod;
    t[node].v[2]=(t[node<<1].v[2]+t[node<<1|1].v[2])%mod;
    t[node].v[3]=(t[node<<1].v[3]+t[node<<1|1].v[3])%mod;
}
void push_down(int node){
    int mid = (t[node].b+t[node].e)>>1,len;
    if(t[node].lazy[3]){
        t[node<<1].lazy[3]=t[node<<1|1].lazy[3]=t[node].lazy[3];
        t[node<<1].lazy[2]=t[node<<1|1].lazy[2]=1;
```

```
        t[node<<1].lazy[1]=t[node<<1|1].lazy[1]=0;

        len=mid-t[node].b+1;
        t[node<<1].v[1]=t[node].lazy[3]*len%mod;
        t[node<<1].v[2]=t[node].lazy[3]*t[node].lazy[3]%mod*len%mod;
        t[node<<1].v[3]=t[node].lazy[3]*t[node].lazy[3]%mod*t[node].lazy[3]%mod*len%mod;

        len=t[node].e-mid;
        t[node<<1|1].v[1]=t[node].lazy[3]*len%mod;
        t[node<<1|1].v[2]=t[node].lazy[3]*t[node].lazy[3]%mod*len%mod;
        t[node<<1|1].v[3]=t[node].lazy[3]*t[node].lazy[3]%mod*t[node].lazy[3]%mod*len%mod;
        t[node].lazy[3]=0;
    }
    if(t[node].lazy[2]!=1){
        t[node<<1].lazy[2]=t[node<<1].lazy[2]*t[node].lazy[2]%mod;
        t[node<<1].lazy[1]=t[node<<1].lazy[1]*t[node].lazy[2]%mod;

        t[node<<1|1].lazy[2]=t[node<<1|1].lazy[2]*t[node].lazy[2]%mod;
        t[node<<1|1].lazy[1]=t[node<<1|1].lazy[1]*t[node].lazy[2]%mod;

        t[node<<1].v[1]=t[node<<1].v[1]*t[node].lazy[2]%mod;
        t[node<<1].v[2]=t[node<<1].v[2]*t[node].lazy[2]%mod*t[node].lazy[2]%mod;
        t[node<<1].v[3]=t[node<<1].v[3]*t[node].lazy[2]%mod*t[node].lazy[2]%mod*t[node].lazy[2]%mod;

        t[node<<1|1].v[1]=t[node<<1|1].v[1]*t[node].lazy[2]%mod;
        t[node<<1|1].v[2]=t[node<<1|1].v[2]*t[node].lazy[2]%mod*t[node].lazy[2]%mod;
        t[node<<1|1].v[3]=t[node<<1|1].v[3]*t[node].lazy[2]%mod*t[node].lazy[2]%mod*t[node].lazy[2]%mod;
        t[node].lazy[2]=1;

    }
    if(t[node].lazy[1]){
        t[node<<1].lazy[1]=(t[node<<1].lazy[1]+t[node].lazy[1])%mod;
        t[node<<1|1].lazy[1]=(t[node<<1|1].lazy[1]+t[node].lazy[1])%mod;

        len=mid-t[node].b+1;
        t[node<<1].v[3]=(t[node<<1].v[3]+3*t[node<<1].v[2]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1].v[3]=(t[node<<1].v[3]+3*t[node<<1].v[1]%mod*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1].v[3]=(t[node<<1].v[3]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;

        t[node<<1].v[2]=(t[node<<1].v[2]+2*t[node<<1].v[1]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1].v[2]=(t[node<<1].v[2]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;

        t[node<<1].v[1]=(t[node<<1].v[1]+len*t[node].lazy[1]%mod)%mod;
        len=t[node].e-mid;
        t[node<<1|1].v[3]=(t[node<<1|1].v[3]+3*t[node<<1|1].v[2]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1|1].v[3]=(t[node<<1|1].v[3]+3*t[node<<1|1].v[1]%mod*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1|1].v[3]=(t[node<<1|1].v[3]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;

        t[node<<1|1].v[2]=(t[node<<1|1].v[2]+2*t[node<<1|1].v[1]%mod*t[node].lazy[1]%mod)%mod;
        t[node<<1|1].v[2]=(t[node<<1|1].v[2]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;

        t[node<<1|1].v[1]=(t[node<<1|1].v[1]+len*t[node].lazy[1]%mod)%mod;

        t[node].lazy[1]=0;
    }
}
void update(int bb ,int ee,int node,int c,int opt){
    if(bb<=t[node].b&&t[node].e<=ee){
        int len=t[node].e-t[node].b+1;
        if(opt==1){
            t[node].lazy[1]=(t[node].lazy[1]+c)%mod;

            t[node].v[3]=(t[node].v[3]+3*t[node].v[2]%mod*c%mod)%mod;
            t[node].v[3]=(t[node].v[3]+3*t[node].v[1]%mod*c%mod*c%mod)%mod;
            t[node].v[3]=(t[node].v[3]+len*c%mod*c%mod*c%mod)%mod;

            t[node].v[2]=(t[node].v[2]+2*t[node].v[1]%mod*c%mod)%mod;
            t[node].v[2]=(t[node].v[2]+len*c%mod*c%mod)%mod;

            t[node].v[1]=(t[node].v[1]+len*c%mod)%mod;

```

```
        }
        if(opt==2){
            t[node].lazy[2]=t[node].lazy[2]*c%mod;
            t[node].lazy[1]=t[node].lazy[1]*c%mod;

            t[node].v[1]=t[node].v[1]*c%mod;
            t[node].v[2]=t[node].v[2]*c%mod*c%mod;
            t[node].v[3]=t[node].v[3]*c%mod*c%mod*c%mod;
        }
        if(opt==3){
            t[node].lazy[2]=1;
            t[node].lazy[1]=0;
            t[node].lazy[3]=c;

            t[node].v[1]=c*len%mod;
            t[node].v[2]=c*c%mod*len%mod;
            t[node].v[3]=c*c%mod*c%mod*len%mod;
        }
        return ;
    }
    int mid=(t[node].b+t[node].e)>>1;
    push_down(node);
    if(bb>mid) update(bb,ee,node<<1|1,c,opt);
    else if(ee<=mid) update(bb,ee,node<<1,c,opt);
    else{
        update(bb,mid,node<<1,c,opt);
        update(mid+1,ee,node<<1|1,c,opt);
    }
    push_up(node);
}
int query(int bb,int ee,int node,int opt){
    if(bb<=t[node].b&&ee>=t[node].e){
        return t[node].v[opt];
    }
    int mid=(t[node].b+t[node].e)>>1;
    push_down(node);
    if(mid<bb)return query(bb,ee,node<<1|1,opt);
    else if(ee<=mid) return query(bb,ee,node<<1,opt);
    else return (query(bb,mid,node<<1,opt)+query(mid+1,ee,node<<1|1,opt))%mod;
}
int main()
{
    int  n ,m ;
    while(scanf("%d %d", &n, &m) && n+m){
        create(1,n,1);
        for(int i=0; i<m; i++){
            int opt,x,y,k;
            scanf("%d%d%d%d",&opt,&x,&y,&k);
            if(opt<=3){
                update(x,y,1,k,opt);
            }
            if(opt==4){
                printf("%d\n",query(x,y,1,k));
            }
        }
    }
    return 0;
}
```


文章最后发布于: 2019-09-10 16:44:23