

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

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/laaahu/article/details/100702480>

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： $\text{sum}[3] = \text{sum}[3] + 3 \times c^2 \times \text{sum}[1] + 3 \times c \times \text{sum}[2] + \text{len} \times c^3$;

平方和： $\text{sum}[2] = \text{sum}[2] + 2 \times \text{sum}[1] \times c + \text{len} \times c^2$;

和： $\text{sum}[1] = \text{sum}[1] + \text{len} \times c$;

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 ：

$\text{sum}[1] = c \times \text{sum}[1]$;

$\text{sum}[2] = c^2 \times \text{sum}[2]$;

$\text{sum}[3] = c^3 \times \text{sum}[3]$;

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）。

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <cmath>
6  using namespace std;
7  const int maxn = 1e5+10;
8  const int mod = 10007;
9  struct node{
10     int b,e,v[5];
11     int lazy[5];
12 }t[maxn<<2];
13 void create(int bb,int ee,int node){
14     t[node].b=bb;
15     t[node].e=ee;
16     t[node].v[1]=0,t[node].v[2]=0,t[node].v[3]=0;
17     t[node].lazy[1]=0,t[node].lazy[2]=1,t[node].lazy[3]=0;
18     if(bb==ee)return ;
19     int mid = (bb+ee)>>1;
20     create(bb,mid,node<<1);
21     create(mid+1,ee,node<<1|1);
22 }
23 void push_up(int node){
24     t[node].v[1]=(t[node<<1].v[1]+t[node<<1|1].v[1])%mod;
25     t[node].v[2]=(t[node<<1].v[2]+t[node<<1|1].v[2])%mod;
26     t[node].v[3]=(t[node<<1].v[3]+t[node<<1|1].v[3])%mod;
27 }
28 void push_down(int node){
29     int mid = (t[node].b+t[node].e)>>1,len;
30     if(t[node].lazy[3]){
31         t[node<<1].lazy[3]=t[node<<1|1].lazy[3]=t[node].lazy[3];
32         t[node<<1].lazy[2]=t[node<<1|1].lazy[2]=1;
```

```

33     t[node<<1].lazy[1]=t[node<<1|1].lazy[1]=0;
34
35     len=mid-t[node].b+1;
36     t[node<<1].v[1]=t[node].lazy[3]*len%mod;
37     t[node<<1].v[2]=t[node].lazy[3]*t[node].lazy[3]%mod*len%mod;
38     t[node<<1].v[3]=t[node].lazy[3]*t[node].lazy[3]%mod*t[node].lazy[3]%mod*len%mod;
39
40     len=t[node].e-mid;
41     t[node<<1|1].v[1]=t[node].lazy[3]*len%mod;
42     t[node<<1|1].v[2]=t[node].lazy[3]*t[node].lazy[3]%mod*len%mod;
43     t[node<<1|1].v[3]=t[node].lazy[3]*t[node].lazy[3]%mod*t[node].lazy[3]%mod*len%mod;
44     t[node].lazy[3]=0;
45 }
46 if(t[node].lazy[2]!=1){
47     t[node<<1].lazy[2]=t[node<<1].lazy[2]*t[node].lazy[2]%mod;
48     t[node<<1].lazy[1]=t[node<<1].lazy[1]*t[node].lazy[2]%mod;
49
50     t[node<<1|1].lazy[2]=t[node<<1|1].lazy[2]*t[node].lazy[2]%mod;
51     t[node<<1|1].lazy[1]=t[node<<1|1].lazy[1]*t[node].lazy[2]%mod;
52
53     t[node<<1].v[1]=t[node<<1].v[1]*t[node].lazy[2]%mod;
54     t[node<<1].v[2]=t[node<<1].v[2]*t[node].lazy[2]%mod*t[node].lazy[2]%mod;
55     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;
56
57     t[node<<1|1].v[1]=t[node<<1|1].v[1]*t[node].lazy[2]%mod;
58     t[node<<1|1].v[2]=t[node<<1|1].v[2]*t[node].lazy[2]%mod*t[node].lazy[2]%mod;
59     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;
60     t[node].lazy[2]=1;
61
62 }
63 if(t[node].lazy[1]){
64     t[node<<1].lazy[1]=(t[node<<1].lazy[1]+t[node].lazy[1])%mod;
65     t[node<<1|1].lazy[1]=(t[node<<1|1].lazy[1]+t[node].lazy[1])%mod;
66
67     len=mid-t[node].b+1;
68     t[node<<1].v[3]=(t[node<<1].v[3]+3*t[node<<1].v[2]%mod*t[node].lazy[1]%mod)%mod;
69     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;
70     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;
71
72     t[node<<1].v[2]=(t[node<<1].v[2]+2*t[node<<1].v[1]%mod*t[node].lazy[1]%mod)%mod;
73     t[node<<1].v[2]=(t[node<<1].v[2]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;
74
75     t[node<<1].v[1]=(t[node<<1].v[1]+len*t[node].lazy[1]%mod)%mod;
76     len=t[node].e-mid;
77     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;
78     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;
79     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;
80
81     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;
82     t[node<<1|1].v[2]=(t[node<<1|1].v[2]+len*t[node].lazy[1]%mod*t[node].lazy[1]%mod)%mod;
83
84     t[node<<1|1].v[1]=(t[node<<1|1].v[1]+len*t[node].lazy[1]%mod)%mod;
85
86     t[node].lazy[1]=0;
87 }
88 }
89 void update(int bb,int ee,int node,int c,int opt){
90     if(bb<=t[node].b&&t[node].e==ee){
91         int len=t[node].e-t[node].b+1;
92         if(opt==1){
93             t[node].lazy[1]=(t[node].lazy[1]+c)%mod;
94
95             t[node].v[3]=(t[node].v[3]+3*t[node].v[2]%mod*c%mod)%mod;
96             t[node].v[3]=(t[node].v[3]+3*t[node].v[1]%mod*c%mod*c%mod)%mod;
97             t[node].v[3]=(t[node].v[3]+len*c%mod*c%mod*c%mod)%mod;
98
99             t[node].v[2]=(t[node].v[2]+2*t[node].v[1]%mod*c%mod)%mod;
100            t[node].v[2]=(t[node].v[2]+len*c%mod*c%mod)%mod;
101
102            t[node].v[1]=(t[node].v[1]+len*c%mod)%mod;
103

```

猿衣酷

专属于程序员的卫衣

关闭

```
104     }
105     if(opt==2){
106         t[node].lazy[2]=t[node].lazy[2]*c%mod;
107         t[node].lazy[1]=t[node].lazy[1]*c%mod;
108
109         t[node].v[1]=t[node].v[1]*c%mod;
110         t[node].v[2]=t[node].v[2]*c%mod*c%mod;
111         t[node].v[3]=t[node].v[3]*c%mod*c%mod*c%mod;
112     }
113     if(opt==3){
114         t[node].lazy[2]=1;
115         t[node].lazy[1]=0;
116         t[node].lazy[3]=c;
117
118         t[node].v[1]=c*len%mod;
119         t[node].v[2]=c*c%mod*len%mod;
120         t[node].v[3]=c*c%mod*c%mod*len%mod;
121     }
122     return ;
123 }
124 int mid=(t[node].b+t[node].e)>>1;
125 push_down(node);
126 if(bb>mid) update(bb,ee,node<<1|1,c,opt);
127 else if(ee<=mid) update(bb,ee,node<<1,c,opt);
128 else{
129     update(bb,mid,node<<1,c,opt);
130     update(mid+1,ee,node<<1|1,c,opt);
131 }
132 push_up(node);
133 }
134 int query(int bb,int ee,int node,int opt){
135     if(bb<=t[node].b&&ee>=t[node].e){
136         return t[node].v[opt];
137     }
138     int mid=(t[node].b+t[node].e)>>1;
139     push_down(node);
140     if(mid<bb)return query(bb,ee,node<<1|1,opt);
141     else if(ee<=mid) return query(bb,ee,node<<1,opt);
142     else return (query(bb,mid,node<<1,opt)+query(mid+1,ee,node<<1|1,opt))%mod;
143 }
144 int main()
145 {
146     int n,m ;
147     while(scanf("%d %d", &n, &m) && n+m){
148         create(1,n,1);
149         for(int i=0; i<m; i++){
150             int opt,x,y,k;
151             scanf("%d%d%d", &opt,&x,&y,&k);
152             if(opt<=3){
153                 update(x,y,1,k,opt);
154             }
155             if(opt==4){
156                 printf("%d\n",query(x,y,1,k));
157             }
158         }
159     }
160     return 0;
161 }
162 }
```

有 0 个人打赏

文章最后发布于: 2019-09-10 16:44:23

©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客

猿衣酷

专属于程序员的卫衣

关闭