可持久化字典树

算法简介

一种可以维护历史版本的字典树，实现方面与可持久化线段树类似，主要用于维护区间异或和。

时间复杂度和空间复杂度均为 $O\left(m\log n\right)$

算法例题

洛谷p4735

题意

给定序列 $a_1,a_2\cdots a_n$支持下述操作：

1. 在序列末尾添加一个数 x
2. 给定 l,r,x询问 $\max_{l \le p\le r} x\oplus(a_p\oplus a_{p+1}\oplus \cdots \oplus a_N)$其中 N 表示当前序列长度

题解

设 $S_i=a_1\oplus a_2\oplus \cdots \oplus a_i$于是询问操作变为 $\max_{l-1\le p\le r-1} x\oplus S_N\oplus S_p$

考虑对序列 ${S_0,S_1,S_2\cdots S_n}$ 建立可持久化字典树，修改操作等价于在最后版本基础上插入新元素。

查询操作用类似主席树的方法查询区间中每个数与 $x\oplus S_N$ 异或的最大值即可。

```
const int MAXN=3e5+5,MAXM=24;
int root[MAXN<<1],ch[MAXN*MAXM*2][2],val[MAXN*MAXM*2],cnt;
void Insert(int &k,int p,int pos,int v){
    k=++cnt;
    val[k]=val[p]+1;
    if(pos<0)return;
    int dir=(v>>pos)&1;
    ch[k][!dir]=ch[p][!dir];
    Insert(ch[k][dir],ch[p][dir],pos-1,v);
}
int query(int k1,int k2,int v){
    int ans=0,pos=MAXM-1;
    while(~pos){
        int dir=(v>>pos)&1;
        if(val[ch[k2][!dir]]-val[ch[k1][!dir]])
```

```
        ans|=(1<<pos),k1=ch[k1][!dir],k2=ch[k2][!dir];
        else
        k1=ch[k1][dir],k2=ch[k2][dir];
        pos--;
    }
    return ans;
}
int s[MAXN];
int main()
{
    int n=read_int(),m=read_int(),pre=0;
    Insert(root[1],root[0],MAXM-1,0);
    n++;
    _rep(i,2,n){
        pre=pre^read_int();
        Insert(root[i],root[i-1],MAXM-1,pre);
    }
    char opt;
    int l,r,x;
    while(m--){
        opt=get_char();
        if(opt=='A'){
            pre=pre^read_int();
            Insert(root[n+1],root[n],MAXM-1,pre);
            n++;
        }
        else{
            l=read_int(),r=read_int(),x=read_int();
            enter(query(root[l-1],root[r],pre^x));
        }
    }
    return 0;
}
```