

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 $O(m \log n)$

算法例题

[洛谷p4735](#)

题意

给定序列 $a_1, a_2, \dots, a_n$ 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 $\max_{l \leq p \leq r} x \oplus (a_p \oplus a_{p+1} \oplus \dots \oplus a_N)$ 其中 N 表示当前序列长度

题解

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

考虑对序列 $\{S_0, S_1, S_2, \dots, S_n\}$ 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*50][2],val[MAXN*50],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;
}
```

算法练习

习题一

[洛谷p4098](#)

题意

给定 n 个互异的数。求 $a_i \oplus a_j$ 的最大值，其中 $i, j \in [l, r]$ 且 a_i 为 $a_l, a_{l+1} \cdots a_r$ 的次大值。

题解

枚举每个次大值，显然次大值确定时区间尽可能向两边拓展可以取到最优解。

考虑对 a_i 进行排序，然后通过双向链表确定最优区间(至多存在两个)。区间查询通过可持久化字典树即可完成。

时空间复杂度 $O(n \log v)$

```
const int MAXN=5e4+5,MAXM=30;
int root[MAXN],ch[MAXN*32][2],val[MAXN*32],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 lef,int rig,int v){
    int ans=0,pos=MAXM-1,k1=root[lef-1],k2=root[rig];
    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 pre[MAXN],nxt[MAXN];
pair<int,int> a[MAXN];
int main()
{
    int n=read_int(),head=0,tail=n+1;
    nxt[head]=1,pre[tail]=n;
    _rep(i,1,n){
        nxt[i]=i+1,pre[i]=i-1;
        a[i]=make_pair(read_int(),i);
        Insert(root[i],root[i-1],MAXM-1,a[i].first);
    }
    sort(a+1,a+n+1);
    int ans=0;
    _rep(i,1,n){
        int l=pre[a[i].second],r=nxt[a[i].second];
        nxt[l]=r,pre[r]=l;
        if(l!=head)ans=max(ans,query(pre[l]+1,r-1,a[i].first));
        if(r!=tail)ans=max(ans,query(l+1,nxt[r]-1,a[i].first));
    }
}
```

```
enter(ans);  
return 0;  
}
```

习题二

洛谷p5283

题意

给定 n 个互异的数，求前 k 大的区间异或和。

题解

前缀异或和处理，问题转化为求前 k 大的点对异或和。

考虑用优先队列维护每个固定点 i 对配对区间 $[0, i-1]$ 的最佳答案，设 i 的最大答案位置为 k_i 则全局最大答案一定为某个 (k_i, i)

弹出全局最大答案，固定点 i 的第二大答案位置一定在区间 $[0, k_i-1]$ 和 $[k_i+1, i-1]$ 将其加入优先队列。

重复 k 次即可得到全局前 k 大答案。

```
const int MAXN=5e5+5, MAXM=32;  
int root[MAXN], ch[MAXN*35][2], val[MAXN*35], idx[MAXN*35], cnt;  
void Insert(int &k, int p, int id, LL v, int pos){  
    k=++cnt;  
    val[k]=val[p]+1;  
    if(pos<0){  
        idx[k]=id;  
        return;  
    }  
    int dir=(v>>pos)&1;  
    ch[k][!dir]=ch[p][!dir];  
    Insert(ch[k][dir], ch[p][dir], id, v, pos-1);  
}  
int query(int lef, int rig, LL v){  
    int pos=MAXM-1, k1=lef?root[lef-1]:0, k2=root[rig];  
    while(~pos){  
        int dir=(v>>pos)&1;  
        if(val[ch[k2][!dir]]-val[ch[k1][!dir]])  
            k1=ch[k1][!dir], k2=ch[k2][!dir];  
        else  
            k1=ch[k1][dir], k2=ch[k2][dir];  
        pos--;  
    }
```

```

    }
    return idx[k2];
}
LL a[MAXN];
struct Node{
    int ql,qr,x,nxt;
    LL v;
    bool operator < (const Node &b)const{
        return v<b.v;
    }
    Node(int ql=0,int qr=0,int x=0){
        this->ql=ql,this->qr=qr,this->x=x;
        nxt=query(ql,qr,a[x]);
        v=a[x]^a[nxt];
    }
};
priority_queue<Node> q;
int main()
{
    int n=read_int(),k=read_int();
    Insert(root[0],root[0],0,0,MAXM-1);
    _rep(i,1,n)a[i]=read_LL()^a[i-1],Insert(root[i],root[i-1],i,a[i],MAXM-1);
    _rep(i,1,n)q.push(Node(0,i-1,i));
    LL ans=0;
    while(k--){
        Node temp=q.top();q.pop();
        ans+=temp.v;
        if(temp.nxt>temp.ql)q.push(Node(temp.ql,temp.nxt-1,temp.x));
        if(temp.nxt<temp.qr)q.push(Node(temp.nxt+1,temp.qr,temp.x));
    }
    enter(ans);
    return 0;
}

```

习题三

[洛谷p5795](#)

题意

给定一个长度为 n 的序列 X 和长度为 m 的序列 Y 定义矩阵元素 $A_{i,j}=X_i \oplus Y_j$ 接下来 q 个询问。

每次询问 $A_{i,j} (l_1 \leq i \leq r_1, l_2 \leq j \leq r_2)$ 第 k 大。数据保证 $n \leq 1000, m \leq 300000, q \leq 500$

题解

对序列 S 建字典树，差分维护区间结点个数。每次询问时同时对序列 $X_i(l_1 \leq r_1)$ 跳字典树。

考虑按位贪心，先查询异或结果为 S 的结点个数，根据结点个数考虑跳边方向。时间复杂度 $O((nq+m)\log v)$

```
const int MAXN=1e3+5,MAXM=3e5+5,MAXL=31;
struct Node{
    int ch[2],sz;
}node[MAXM*40];
int root[MAXM],node_cnt;
void Insert(int &k,int p,int v){
    int pos=k++node_cnt;
    for(int i=MAXL-1;~i;i--){
        int dir=(v>>i)&1;
        node[pos].ch[!dir]=node[p].ch[!dir];
        node[pos].ch[dir]=++node_cnt;
        pos=node[pos].ch[dir];
        p=node[p].ch[dir];
        node[pos].sz=node[p].sz+1;
    }
}
int x[MAXN],y[MAXM],cur_pos[MAXN][2];
int query(int l1,int r1,int l2,int r2,int rk){
    int ans=0;
    _rep(i,l1,r1){
        cur_pos[i][0]=root[r2];
        cur_pos[i][1]=root[l2-1];
    }
    for(int i=MAXL-1;~i;i--){
        int cnt=0;
        _rep(j,l1,r1){
            int dir=((x[j]>>i)&1)^1;
            cnt+=node[node[cur_pos[j][0]].ch[dir]].sz-
node[node[cur_pos[j][1]].ch[dir]].sz;
        }
        int flag=cnt>=rk;
        if(flag)
            ans|=1<<i;
        else
            rk-=cnt;
        _rep(j,l1,r1){
            int dir=((x[j]>>i)&1)^flag;
            cur_pos[j][0]=node[cur_pos[j][0]].ch[dir];
            cur_pos[j][1]=node[cur_pos[j][1]].ch[dir];
        }
    }
    return ans;
}
int main()
{
```

```
int n=read_int(),m=read_int();
_rep(i,1,n)x[i]=read_int();
_rep(i,1,m){
    y[i]=read_int();
    Insert(root[i],root[i-1],y[i]);
}
int q=read_int();
while(q--){
    int l1=read_int(),r1=read_int(),l2=read_int(),r2=read_int();
    enter(query(l1,r1,l2,r2,read_int()));
}
return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%8F%AF%E6%8C%81%E4%B9%85%E5%8C%96%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84_3

Last update: 2020/09/20 19:30