字典树 Trie

前k大区间异或和

给一个长度为n的数组a，求前k大的区间$[l_i,r_i]$异或和$\bigoplus_{i=l_i}^{r_i} a_i$，并输出他们的和。

思路：Trie+优先队列

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N=5e5+7;
const int mod=1e9+7;

int n,k,a[N],ans=0;

struct Trie{
    int ch[2],sz;
}t[N<<5];
int tot=1;

struct node{
    int id,rk,v; //编号id，第rk大异或和，答案为v
    bool operator < (node x)const{return v<x.v;}
};
priority_queue<node>q; //堆

void insert(int v){ //Trie的插入操作
    int p=1;
    t[p].sz++; //记录结点子树下有多少个插入数字
    for(int i=31;i>=0;i--){
        int c=(v>>i)&1ll;
        if(!t[p].ch[c]) t[p].ch[c]=++tot;
        p=t[p].ch[c];
        t[p].sz++;
    }
}

int query(int v,int k){ //查询v的在Trie上的第k大异或值
    int p=1,ans=0;
    for(int i=31;i>=0;i--){
        int c=(v>>i)&1ll;
```

```
        if(t[t[p].ch[c]].sz>=k) p=t[p].ch[c];
        else k-=t[t[p].ch[c]].sz,p=t[p].ch[c^1],ans|=(1ll<<i);
    }
    return ans;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>k;
    k*=2; //考虑到[l,r]两端重复计算
    insert(0);
    for(int i=1;i<=n;i++){
        cin>>a[i];
        a[i]^=a[i-1]; //处理区间异或前缀和
        insert(a[i]);
    }
    for(int i=0;i<=n;i++) q.push({i,n+1,query(a[i],n+1)});
    while(k--){ //前2*k个区间异或最大值
        node fro=q.top();
        q.pop();
        ans+=fro.v;
        if(fro.rk) q.push({fro.id,fro.rk-1,query(a[fro.id],fro.rk-1)});
    }
    cout<<ans/2<<"\n";
    return 0;
}
```

融合树(Fusion Tree)

在$O(logn/logw+logw)$时间复杂度下支持插入、删除、前驱、后继、min/max，整数排序。

luoguP6018 [Ynoi2010] Fusion tree

```
给出一棵树，每个节点都有一个权值。
有三种操作：
1. 把点x周围的点的点权+1
2. 把点x的点权-v
3. 求点x周围的点的权值异或和
这里定义“周围”为树上与一个节点 x 距离为 1 的节点集合
```

每个结点建立一棵 trie 维护其儿子的权值，trie 应该支持全局加一。 可以使用在每一个结点上设置懒标记来标记儿子的权值的增加量。

```
#include<bits/stdc++.h>
using namespace std;
const int N = 5e5+10;
namespace trie {
    int rt[N],ch[N<<5][2];
    int w[N<<5];  //w[o]指节点o到其父亲节点这条边上数值的数量（权值）。
    int xorv[N<<5];
    int tot=0;
```

```
    void maintain(int o){  //维护w数组和xorv（权值的异或）数组
        w[o]=xorv[o]=0;
        if(ch[o][0]){
            w[o]+=w[ch[o][0]];
            xorv[o]^=xorv[ch[o][0]]<<1;
        }
        if(ch[o][1]){
            w[o]+=w[ch[o][1]];
            xorv[o]^=(xorv[ch[o][1]]<<1)|(w[ch[o][1]]&1);
        }
    }
    inline int mknode(){  //创造一个新的节点
        ++tot;
        ch[tot][0]=ch[tot][1]=0;
        w[tot]=0;
        return tot;
    }
    void insert(int &o, int x, int dep) {  // x是权重，dp是深度
        if(!o) o=mknode();
        if(dep>20) return (void)(w[o]++);
        insert(ch[o][x&1],x>>1,dep+1);
        maintain(o);
    }
    void erase(int o, int x, int dep){
        if(dep>20) return (void)(w[o]--);
        erase(ch[o][x&1],x>>1,dep+1);
        maintain(o);
    }
    void addall(int o){  //对所有节点+1即将所有节点的ch[o][1]和ch[o][0]交换
        swap(ch[o][1],ch[o][0]);
        if(ch[o][0]) addall(ch[o][0]);
        maintain(o);
    }
}  // namespace trie

int head[N];
struct Edge{
    int v,n;
}e[N<<1];
int tot=0;
void addedge(int u, int v) {
    e[++tot]=(Edge){v,head[u]};head[u]=tot;
    e[++tot]=(Edge){u,head[v]};head[v]=tot;
}

int n,m,u,v,op,rt,lztar[N],fa[N],V[N],res;

void dfs(int o, int f) {  //得到fa数组
    fa[o]=f;
    for(int i=head[o];i;i=e[i].n){  //遍历子节点
        int to=e[i].v;
        if(to==f) continue;
        dfs(to,o);
    }
}

inline int get(int x) {
    return (fa[x] == -1 ? 0 : lztar[fa[x]]) + V[x];
```

```
}  //权值函数

int main() {
    cin>>n>>m;
    for(int i=1;i<n;i++){
        cin>>u>>v;
        addedge(u,rt=v);  //双向建边
    }
    dfs(rt,-1);  // rt是随机的一个点
    for(int i=1;i<=n;i++){
        cin>>V[i];
        if(fa[i]!=-1) trie::insert(trie::rt[fa[i]],V[i],0);
    }
    while(m--){
        cin>>op>>u;
        if(op==1){
            lztar[u]++;
            if(u!= rt){
                if(fa[fa[u]]!=-1) trie::erase(trie::rt[fa[fa[u]]],
get(fa[u]),0);
                V[fa[u]]++;
                if(fa[fa[u]]!=-1) trie::insert(trie::rt[fa[fa[u]]],
get(fa[u]),0);  //重新插入
            }
            trie::addall(trie::rt[u]);  //对所有节点+1
        }else if(op==2){
            cin>>v;
            if(u!=rt) trie::erase(trie::rt[fa[u]],get(u),0);
            V[u]-=v;
            if(u!=rt) trie::insert(trie::rt[fa[u]],get(u),0);  //重新插入
        }else{
            res=trie::xorv[trie::rt[u]];
            res^=get(fa[u]);
            printf("%d\n",res);
        }
    }
    return 0;
}
```

luoguP6623 【省选联考 2020 A 卷】 树

给定一棵以1为根的有根树，记 $d(u,v)$为 $u \to v$简单路径上的边数， $w(u)$ 为点 u的权值，$val(u)=\sum_v d(u,v)+w(v)$。其中， v 是 u 的子树内的点（包括点 u ）。求 $\sum_{i=1}^{n} val(i)$

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N = 526010;
namespace trie {
    int ch[N<<5][2];
    int w[N<<5];  //w[o]指节点o到其父亲节点这条边上数值的数量（权值）。
    int xorv[N<<5];
    int tot=0;
    void maintain(int o){  //维护w数组和xorv（权值的异或）数组
        w[o]=xorv[o]=0;
```

```
        if(ch[o][0]){
            w[o]+=w[ch[o][0]];
            xorv[o]^=xorv[ch[o][0]]<<1;
        }
        if(ch[o][1]){
            w[o]+=w[ch[o][1]];
            xorv[o]^=(xorv[ch[o][1]]<<1)|(w[ch[o][1]]&1);
        }
        w[o]=w[o]&1;
    }
    inline int mknode(){  //创造一个新的节点
        ++tot;
        ch[tot][0]=ch[tot][1]=w[tot]=xorv[tot]=0;
        return tot;
    }
    void insert(int &o, int x, int dep) {  // x是权重，dp是深度
        if(!o) o=mknode();
        if(dep>21) return (void)(w[o]++);
        insert(ch[o][x&1],x>>1,dep+1);
        maintain(o);
    }
    int merge(int a,int b){
        if(!a) return b;
        if(!b) return a;
        w[a]=w[a]+w[b];
        xorv[a]^=xorv[b];
        ch[a][0]=merge(ch[a][0],ch[b][0]);
        ch[a][1]=merge(ch[a][1],ch[b][1]);
        return a;
    }

    void addall(int o){  //对所有节点+1即将所有节点的ch[o][1]和ch[o][0]交换
        swap(ch[o][1],ch[o][0]);
        if(ch[o][0]) addall(ch[o][0]);
        maintain(o);
    }
}  // namespace trie

int head[N];
struct Edge{
    int v,n;
}e[N<<1];
int tot=0;
void addedge(int u,int v){
    e[++tot]=(Edge){v,head[u]};head[u]=tot;
}

int n,m,u,v,op,rt[N],V[N],ans;

void dfs(int o){
    for(int i=head[o];i;i=e[i].n){
        int to=e[i].v;
        dfs(to);
        rt[o]=trie::merge(rt[o],rt[to]);
    }
    trie::addall(rt[o]);
    trie::insert(rt[o],V[o],0);
    ans+=trie::xorv[rt[o]];
```

```
}

signed main(){
    cin>>n;
    for(int i=1;i<=n;i++) cin>>v[i];
    for(int i=2;i<=n;i++){
        cin>>u;
        addedge(u,i);
    }
    dfs(1);
    printf("%lld",ans);
    return 0;
}
```

参考资料

OI-wiki

https://www.luogu.com.cn/problem/P6018 题目背景