异或与字典树的应用

0.异或的相关性质.

1.$a \oplus a = 0, 0 \oplus a = a \oplus 0 = a, a \in N$

2.$a \oplus b = b \oplus a$

3.$a \oplus b = c \Leftrightarrow a \oplus c = b \Leftrightarrow b \oplus c = a$

4.记$a_1 \oplus a_2 \oplus a_3 \cdots \oplus a_i = pre[i]$

有$a_l \oplus a_{l+1} \cdots \oplus a_{r-1} \oplus a_r = pre[l-1] \oplus pre[r]$

5.$(a \oplus b) \oplus c = a \oplus (b \oplus c)$

1.异或在字典树的优化.

考虑下列问题:

一.给定序列$a_1, a_2 \ldots, a_n$**，求**$max\{a_i \oplus a_j\}, (1 \le i < j \le n)$
$n \le 1e5$。

1.暴力枚举i, j位置，复杂度为$O(n^2)$ 显然不行。

2.考虑对内层循环优化.

对于前$i-1$个数建立一棵从高位到低位的01字典树，然后贪心地从最高位开始选，根据异或的(不同为1，相同为0)的原则，显然优先考虑不同的位，若该位没有结点，则只能选择当前已有的结点，一直走到叶子结点结束的值就是最大值。

每次插入字符和查询都是$O(31) = log(int)$

总时间复杂度：$O(n^2) \to O(31n)$

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N=1e5+5,M=3e6+5;
int n,a[N],son[M][2],cnt;
void insert(int x){
    int p=0;
    for(int i=30;~i;i--){
        int &s=son[p][x>>i&1];
        if(!s) s=++cnt;
        p=s;
    }
}
int query(int x){
    int ans=0,p=0;
    for(int i=30;~i;i--){
        int s=x>>i&1;
        if(son[p][!s]) ans+=1<<i,p=son[p][!s];
        else p=son[p][s];
```

```
    }
    return ans;
}
int main(){
    cin>>n;
    int ans=0;
    for(int i=0;i<n;i++)
        cin>>a[i],ans=max(ans,query(a[i])),insert(a[i]);
    cout<<ans<<endl;
}
```

二.给定序列$a_1,a_2\ldots,a_n$，求最大区间异或和.

显然预处理前缀和，然后方法同上。

三.给定序列$a_1,a_2\ldots,a_n$，求两段不相交区间最大异或和.

显然预处理前缀和与后缀和，然后取$max\{f_{pre}[i]+f_{suf}[i+1]\},(i\in[1,n))$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=4e5+5,M=8e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
int a[N],son[N][2],n,pre[N],suf[N],f[N],cnt=0;
void insert(int x){
    int p=0;
    for(int i=30;~i;i--){
        int &s=son[p][x>>i&1];
        if(!s) s=++cnt;
        p=s;
    }
}
int query(int x){
    int p=0,ans=0;
    for(int i=30;~i;i--){
        int s=x>>i&1;
        if(son[p][!s]) ans+=1<<i,p=son[p][!s];
        else p=son[p][s];
    }
    return ans;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++ )scanf("%d",&a[i]),pre[i]=pre[i-1]^a[i];
    for(int i=n;i;i--) suf[i]=suf[i+1]^a[i];
    for(int i=1;i<=n;i++){
        f[i]=max(f[i-1],query(pre[i]));
        insert(pre[i]);
```

```
    }
    int ans=0;mst(son),cnt=0;
    for(int i=n;i;i--){
        ans=max(ans,f[i-1]+query(suf[i]));
        insert(suf[i]);
    }
    printf("%d\n",ans);
    return 0;
}
```

四.给定一棵树,求树上路径的最大异或和.

考虑以1为根进行dfs，求出所有节点i到1的路径异或和$f[i]$，然后根据树的性质可知：

$Xor_{u-v}=f[u]\oplus f[v]$

然后就可以用01字典树的板子了。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
typedef long long ll;
const int N=1e5+5,M=3e6+5,inf=0x3f3f3f3f,mod=1e9+7;
#define mst(a) memset(a,0,sizeof a)
#define lx x<<1
#define rx x<<1|1
#define reg register
#define PII pair<int,int>
#define fi first
#define se second
#define pb push_back
int n,son[M][2],f[N],h[N],cnt,tot;
struct node{
    int to,nt,w;
}e[N<<1];
void add(int u,int v,int w){
    e[++tot]={v,h[u],w},h[u]=tot;
    e[++tot]={u,h[v],w},h[v]=tot;
}
void dfs(int u,int fa){
    for(int i=h[u];i;i=e[i].nt){
         if(e[i].to==fa) continue;
         int v=e[i].to;
         f[v]=f[u]^e[i].w;
         dfs(v,u);
    }
}
void ins(int x){
    int p=0;
    for(int i=30;~i;i--){
        int &s=son[p][x>>i&1];
        if(!s) s=++cnt;
        p=s;
    }
}
```

```
int que(int x){
    int p=0,ans=0;
    for(int i=30;~i;i--){
        int s=x>>i&1;
        if(son[p][!s]) ans+=1<<i,p=son[p][!s];
        else p=son[p][s];
    }
    return ans;
}
int main(){
    while(~scanf("%d",&n)){
    for(int i=0;i<=n;i++) h[i]=f[i]=0;tot=cnt=0;
    mst(son);
    for(int i=1,u,v,w;i<n;i++){
        scanf("%d%d%d",&u,&v,&w);
        add(u,v,w);
    }
    dfs(0,-1);int ans=0;
    for(int i=0;i<=n;i++){
        ans=max(ans,que(f[i]));
        ins(f[i]);
    }
    printf("%d\n",ans);
    }
    return 0;

}
```