**Trie目录**

[零、大致模板：](#_Toc14179_WPSOffice_Level1) [1](#_Toc14179_WPSOffice_Level1)

[一、 前缀统计 （trie树，节点记录cnt）](#_Toc188_WPSOffice_Level1) [1](#_Toc188_WPSOffice_Level1)

[二、 The Xor Largest Pair （求N个数中最大的异或对）](#_Toc11346_WPSOffice_Level1) [2](#_Toc11346_WPSOffice_Level1)

[三、 The Xor Longest Path（最长异或值路径）](#_Toc11076_WPSOffice_Level1) [3](#_Toc11076_WPSOffice_Level1)

# **零、大致模板：**

**int** trie[Size][26] , tot = 1; //初始化，假设字符串由小写字母构成

**int** end[Size];

**void** Insert(**char**\* str){ //插入一个字符串

**int** len = **strlen**(str) , p = 1;

**for**(**int** k = 0 ; k < len ; ++ k){

**int** ch = str[k] - 'a';

**if**(trie[p][ch] == 0) trie[p][ch] = ++ tot;

        p = trie[p][ch];

    }

    end[p] = **true**; //标记p是一个字符串的末尾

}

**bool** search(**char** \*str){

**int** len = **strlen**(str) , p = 1;

**for**(**int** k = 0 ; k < len ; ++ k){

        p = trie[p][str[k] - 'a'];

**if**(p == 0) **return** **false**;

    }

**return** end[p]; //是否有以p为结尾的单词

}

### 前缀统计 （trie树，节点记录cnt）

节点上存储一个cnt，记录该节点是多少个字符串的末尾节点。（为了处理插入重复字符串的情况）对于每个询问，在Trie树种检索T，在检索过程种累加途径的每个节点的cnt值，就是该询问的答案。

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 1000005;

**char** s[maxn];

**int** trie[maxn][26] , tot = 1 , cnt[maxn];

**void** Insert(**char**\* str){

**int** len = **strlen**(str) , p = 1;

**for**(**int** k = 0 ; k < len ; ++ k){

**int** ch = str[k] - 'a';

**if**(trie[p][ch] == 0) trie[p][ch] = ++ tot;

        p = trie[p][ch];

    }

    cnt[p] ++; //以p结尾的单词个数加1

}

**int** search(**char** \*str){

**int** len = **strlen**(str) , p = 1 , res = 0;

**for**(**int** k = 0 ; k < len ; ++ k){

        p = trie[p][str[k] - 'a'];

        res += cnt[p]; //将路径上的个数加起来

**if**(p == 0) **return** res;

    }

**return** res; //返回的res就是str的前缀的个数

}

**int** main(){

**int** n , m;

**scanf**("%d%d",&n,&m);

**for**(**register** **int** i = 0 ; i < n ; ++ i){

**scanf**("%s",s);

        Insert(s);

    }

**for**(**register** **int** i = 0 ; i < m ; ++ i){

**scanf**("%s",s);

**printf**("%d\n",search(s));

    }

**return** 0;

}

### The Xor Largest Pair （求N个数中最大的异或对）

**注意空间复杂度：O(NC),N为节点个数，C是字符集大小。**

**此题节点个数：1e5个数，每个数30位，节点开3e6个。**

**字符集大小C：2。**

#include <iostream>

**using** **namespace** std;

**const** **int** maxn = 3000005;

**int** trie[maxn][2] , tot = 0;

**void** Insert(**int** x){ //

**int** p = 0;

**for**(**int** i = 30 ; i >= 0 ; -- i){

**if**(trie[p][x >> i & 1] == 0) trie[p][x >> i & 1] = ++ tot;

        p = trie[p][x >> i & 1];

    }

}

**int** search(**int** x){

**int** p = 0 , res = 0;

**for**(**int** i = 30 ; i >= 0 ; -- i){

**int** state = x >> i & 1;

**if**(trie[p][!state]){ //如果有相反的指针

            res += 1 << i; //当前位为1，答案增加

            p = trie[p][!state]; //转移

        }**else** p = trie[p][state];

    }

**return** res;

}

**int** main(){

    ios::sync\_with\_stdio(**false**);

    cin.tie(0) , cout.tie(0);

**int** n;

    cin >> n;

**int** res = 0;

**for**(**int** i = 0 ; i < n ; ++ i){

**int** x;

        cin >> x;

        res = max(res , search(x));

        Insert(x);

    }

    cout << res << '\n';

**return** 0;

}

### The Xor Longest Path（最长异或值路径）

1. dfs求出所有节点到根的异或值D[]

又a 到 b的路径异或值 = D[a] ^ D[b] （a到根的异或值 Xor b到根的异或值）

1. 求出的D[]中的最大异或对就是所求答案，即转换为上一道题目。

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 100010;

**const** **int** maxm = 3000005;

**int** ver[maxn << 1] , edge[maxn << 1] , Next[maxn << 1];

**int** head[maxn] , dis[maxn] , tot = 0;

**int** trie[maxm][2] , idx = 0;

**void** add(**int** u , **int** v ,**int** w){

    ver[++ tot] = v , edge[tot] = w;

    Next[tot] = head[u] , head[u] = tot;

}

**void** dfs(**int** u , **int** fa , **int** sum){

    dis[u] = sum;

**for**(**int** i = head[u] ; i != -1 ; i = Next[i]){

**int** v = ver[i];

**if**(v != fa) dfs(v , u , sum ^ edge[i]);

    }

}

**void** Insert(**int** x){

**int** p = 0;

**for**(**int** i = 30 ; i >= 0 ; -- i){

**bool** state = x >> i & 1;

**if**(trie[p][state] == 0) trie[p][state] = ++ idx;

        p = trie[p][state];

    }

}

**int** search(**int** x){

**int** p = 0 , res = 0;

**for**(**int** i = 30 ; i >= 0 ; -- i){

**bool** state = x >> i & 1;

**if**(trie[p][!state]){

            res += 1 << i;

            p = trie[p][!state];

        }**else** p = trie[p][state];

    }

**return** res;

}

**int** main(){

**int** n;

**scanf**("%d",&n);

**memset**(head , -1 , **sizeof** head);

**for**(**int** i = 0 ; i < n - 1; ++ i){

**int** u , v, w;

**scanf**("%d%d%d",&u,&v,&w);

        add(u , v, w);

        add(v , u, w);

    }

    dfs(0 , -1 , 0);  //1.dfs求每个点到根节点0的距离

    //2.求dis[]中的最大异或和对就是答案

**int** res = 0;

**for**(**int** i = 0 ; i < n; ++ i){

        res = max(res , search(dis[i]));

        Insert(dis[i]);

    }

**printf**("%d\n",res);

**return** 0;

}