# 题目

给你一个整数数组nums，返回nums[i] XOR nums[j]的最大运算结果，其中0≤i≤j< n。

示例 1：

输入：nums = [3,10,5,25,2,8]

输出：28

解释：最大运算结果是 5 XOR 25 = 28.

示例 2：

输入：nums = [14,70,53,83,49,91,36,80,92,51,66,70]

输出：127

提示：

1 <= nums.length <= 2 \* 105

0 <= nums[i] <= 231 - 1

# 分析

## 方法一：哈希表

我们可以通过迭代求解。具体思路如下：

1、首先，我们从最高位开始逐位确定异或结果的每一位。假设当前我们要确定第k位，我们可以先将所有数字的第k位放入一个集合prefixes中。

2、然后，假设当前已确定的异或结果为x，我们尝试将x的第k位设为1，然后查看是否存在两个前缀a和b，使得a ^ b == x。如果存在，则说明x的第k位可以为1，否则为0。

3、重复以上步骤直到确定了所有位，最终得到最大的异或结果。

代码：

class Solution {

public:

int findMaximumXOR(vector<int>& nums) {

int maxXor = 0;

int mask = 0;

unordered\_set<int> prefixes;

for (int i = 31; i >= 0; --i) {

mask |= (1 << i);

prefixes.clear();

for (int num : nums) {

prefixes.insert(num & mask);

}

int nextXor = maxXor | (1 << i);

for (int prefix : prefixes) {

if (prefixes.count(prefix ^ nextXor)) {

maxXor = nextXor;

break;

}

}

}

return maxXor;

}

};

这个实现中，我们首先定义了一个变量maxXor用于记录当前已确定的异或结果，一个变量mask用于掩码来确定当前要确定的位，一个集合prefixes用于存放当前所有数字的前缀。然后，我们从最高位开始逐位确定异或结果的每一位，更新maxXor，直到确定了所有位。最终返回maxXor。

## 方法二：字典树

这个问题可以使用位运算和字典树（Trie）来解决。首先，我们可以观察到对于任意两个数字的异或运算，其结果的最高位在原数字中的最高位之后。因此，我们可以从最高位开始逐位确定异或结果的每一位。

具体步骤如下：

1、首先，我们将所有数字转换为二进制形式，并将它们插入到字典树中。

2、然后，我们从最高位开始逐位确定异或结果的每一位。假设当前我们要确定第 k 位，我们可以从字典树中查找是否存在两个数字，它们的第 k 位异或为 1。如果存在，则该位为 1，否则为 0。

3、最终，我们将所有确定的位组合起来，即为最大的异或结果。

代码：

class TrieNode {

public:

TrieNode\* children[2];

TrieNode() {

children[0] = nullptr;

children[1] = nullptr;

}

};

class Trie {

public:

TrieNode\* root;

Trie() {

root = new TrieNode();

}

void insert(int num) {

TrieNode\* node = root;

for (int i = 31; i >= 0; --i) {

int bit = (num >> i) & 1;

if (node->children[bit] == nullptr) {

node->children[bit] = new TrieNode();

}

node = node->children[bit];

}

}

int findMaximumXOR(vector<int>& nums) {

int maxXor = 0;

Trie\* trie = new Trie();

for (int num : nums) {

trie->insert(num);

}

for (int num : nums) {

TrieNode\* node = trie->root;

int currXor = 0;

for (int i = 31; i >= 0; --i) {

int bit = (num >> i) & 1;

if (node->children[1 - bit] != nullptr) {

currXor += (1 << i);

node = node->children[1 - bit];

} else {

node = node->children[bit];

}

}

maxXor = max(maxXor, currXor);

}

return maxXor;

}

};

class Solution {

public:

int findMaximumXOR(vector<int>& nums) {

Trie trie;

return trie.findMaximumXOR(nums);

}

};

这个实现中，我们首先定义了一个 Trie 类和 TrieNode 类，用于表示字典树。然后，我们在 Trie 类中实现了插入数字和查找最大异或结果的方法。最终，在主函数中创建了一个 Trie 对象，并调用了 `findMaximumXOR` 方法来求解最大异或结果。