1004 左右脑互搏题解

Gospel_rock

1 题意

给定一个包含 n（$1 \le n \le 20$）个正整数的多重集合 S。左脑（先手）和右脑（后手）轮流操作：每次必须从集合中删除一个元素 x，且 x 必须**严格大于**删除后剩余元素的异或和。若集合中只剩一个元素，可以直接删去（剩余异或和为 0）。无法操作者判负。双方均采用最优策略，求谁获胜。

2 分析

2.1 博弈建模

这是一个标准的**组合博弈**问题。每个局面（当前集合的状态）在双方最优策略下有确定的胜负：

- **必胜态**：存在至少一种合法操作，使得对手进入**必败态**。
- **必败态**：所有合法操作都使对手进入**必胜态**；或者根本无法操作。

空集是必败态（当前玩家无法操作）。我们从空集出发，自底向上递推出所有状态的胜负。

2.2 状态表示——为什么需要状压？

朴素做法是对多重集合进行 DFS 回溯搜索。但问题在于：**不同的删除顺序可以到达相同的剩余集合**。例如先删 a 再删 b，和先删 b 再删 a，到达的集合完全一样，博弈结论也一样，却被搜了两次。

图 1 展示了 3 个元素 $\{a_0, a_1, a_2\}$ 的状态转移关系。每个节点是一个子集（用二进制掩码标注），箭头表示”删除一个元素” 的转移。可以看到 $\{a_0\}$、$\{a_1\}$、$\{a_2\}$ 各自都被**两条不同的路径**到达。如果不记忆化，同一个状态会被重复计算多次；当 $n = 20$ 时，不同的删除排列数量可达 20!，远远无法承受。

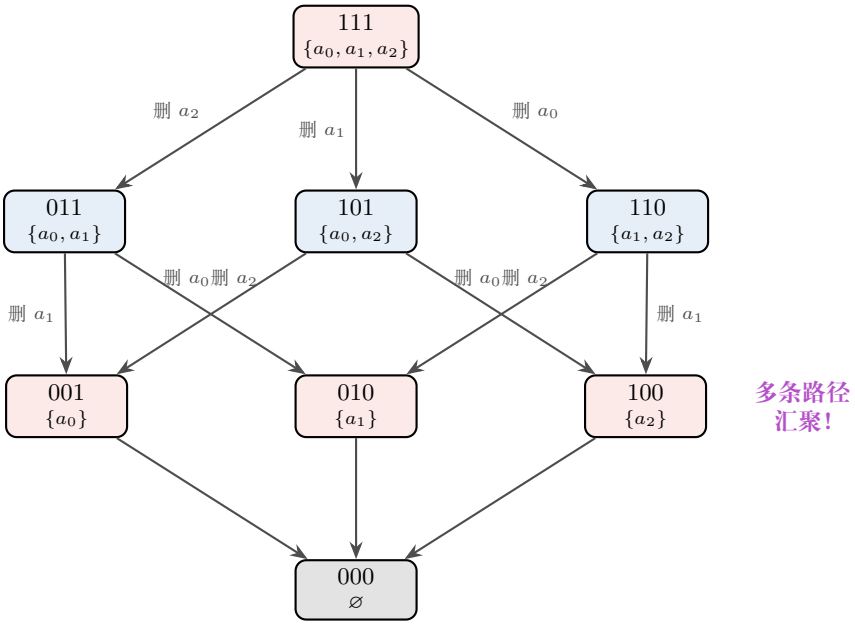

Figure 1: 3 个元素的子集状态转移 DAG（忽略操作合法性）。同一状态可从多条路径到达，如 $\{a_0\}$ 同时被 $\{a_0, a_1\}$ 和 $\{a_0, a_2\}$ 转移到。用掩码标识状态后，每个状态只需计算一次。

解决方案：$n \le 20$，用一个 n 位的二**进制掩码**（bitmask）表示” 哪些元素还在集合中”。共 2^n 种状态，对每个状态记忆化搜索结果即可。

注意到轮到谁操作也不需要额外记录——已经删除了 $n - \texttt{popcount}(mask)$ 个元素，其奇偶性直接决定当前是左脑还是右脑。

2.3 DFS 的设计：用一行表达极小极大

博弈搜索的核心在于 `dfs(sta)` 的返回值定义和递归逻辑。我们令 `dfs(sta)` 返回 1 表示” 当前操作者必胜”，0 表示” 当前操作者必败”。则关键的一行代码为：

```
res |= dfs(sta - (1ll << i)) ^ 1;
```

这一行同时完成了两件事：

- **视角翻转**（`^ 1`）：`dfs` 返回的是对手的胜负，而我们关心的是” 这步操作对**我**是否有利”。对手必败（返回 0）意味着我必胜（$0 \oplus 1 = 1$）；对手必胜（返回 1）意味着我必败（$1 \oplus 1 = 0$）。
- **存在性判定**（`|=`）：只要**存在**某个操作使对手进入必败态，`res` 就被置为 1，此后不论其它操作结果如何都不会再变回 0。这正是” 必胜态 $\iff$ 存在至少一步使对手进入必败态” 的直接编码。

若所有合法操作都找不到让对手必败的，`res` 保持初始值 0，即当前状态为必败态。这也自然涵盖了” 无合法操作” 的情况（循环体不执行，直接返回 0）。

图 2 展示了这一过程：当前状态 S 有三种合法操作，分别转移到 S_1, S_2, S_3。递归返回的是**对手视角**下的胜负，经 `^ 1` 翻转为我方视角后，再用 `|=` 汇总——只要有一个 1，当前状态就是必胜态。

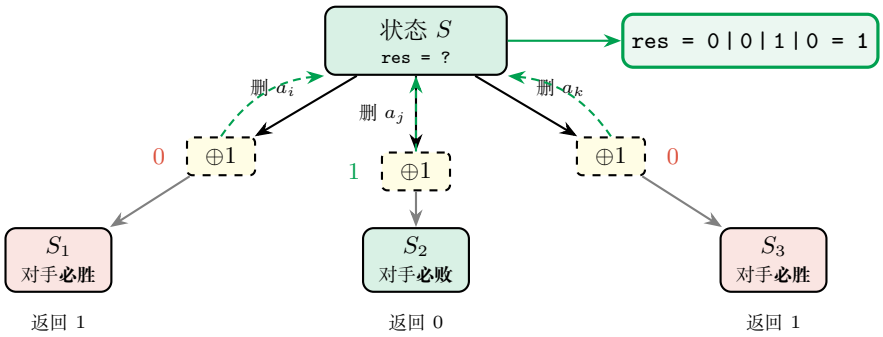

Figure 2: `res |= dfs(...) ^ 1` 的工作原理。递归返回对手视角的胜负，经 $\oplus 1$ 翻转为我方视角，再用 `|=` 汇总。只要有一个分支翻转后为 1，当前状态即为必胜。

2.4 完整算法流程

对状态 $mask$，执行如下递归：

1. 计算当前集合的异或和 $S = \bigoplus_{i:\, mask \text{ 的第} i \text{ 位为1}} a_i$。
2. 枚举 $mask$ 中每个为 1 的位 i，若 $a_i > S \oplus a_i$，则尝试删除 a_i，递归求解 $mask \setminus \{i\}$。
3. 若存在某次删除使对手进入必败态（`res` 被 `|=` 置 1），则当前状态为**必胜态**；否则为**必败态**。
4. 将结果存入 `memo`$[mask]$ 以避免重复计算。

3 样例推演

以样例 2（$\{2, 3, 6, 8\}$）为例，图 3 展示了完整的博弈过程。由于每一步恰好只有一种合法操作，博弈路径是唯一确定的。

样例 3（$\{2, 2\}$）则更简单：当前异或和为 $2 \oplus 2 = 0$，左脑若删去一个 2，剩余异或和为 2，不满足 $2 > 2$，因此左脑**无法进行任何操作**，直接判负。

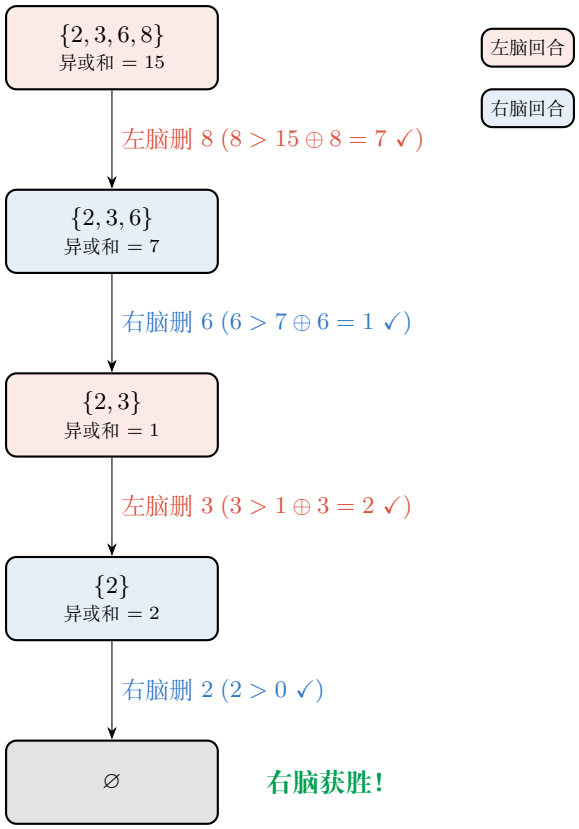

Figure 3: 样例 2（$\{2,3,6,8\}$）的博弈过程。每步只有唯一合法操作，共 4 步，最后一步由右脑完成，右脑获胜。

4 复杂度分析

- **时间复杂度**：共 2^n 个状态，每个状态枚举 $O(n)$ 个元素尝试转移，总计 $O(2^n \cdot n)$。当 $n = 20$ 时约为 2×10^7，可以通过。
- **空间复杂度**：$O(2^n)$ 用于存储记忆化数组。

5 AC 代码

```
// teamname: Gospel_rock
/**
 * Problem: 左右脑互搏
 * Contest:
 * Judge: HDOJ
 * URL: https://acm.hdu.edu.cn/contest/problem?cid=1200&pid=1004
 * Created: 2026-04-11 23:42:01
 * Author: Gospel_rock
 * My blog: https://znzryb.com/
 *
 * Powered by AutoCp https://github.com/Pushpavel/AutoCp
 */

#include <bits/stdc++.h>
#define all(vec) vec.begin(),vec.end()
#define lson(o) (o<<1)
#define rson(o) (o<<1|1)
#define SZ(a) ((long long) a.size())
#define debug(var) cerr << #var <<" = ["<<var<<"]"<<"\n";
#define debug1d(a)    \
cerr << #a << " = [";   \
```

```
for (int i = 0; i < (int)(a).size(); i++) \
cerr << (i ? ", " : "") << a[i]; \
cerr << "]\n";
#define debug2d(a)  \
cerr << #a << " = [\n";  \
for (int i = 0; i < (int)(a).size(); i++)  \
{   \
cerr << "  [";  \
for (int j = 0; j < (int)(a[i]).size(); j++) \
cerr << (j ? ", " : "") << a[i][j];   \
cerr << "]\n";   \
}  \
cerr << "]\n";
#define cend cerr<<"\n----------\n"
#define fsp(x) fixed<<setprecision(x)

using namespace std;

using ll = long long;
using ull = unsigned long long;
using DB = double;
using i128 = __int128;
using CD = complex<double>;

static constexpr ll MAXN = (ll) 1e6 + 10, INF = (1ll << 61) - 1;
static constexpr ll mod = 998244353; // (ll)1e9+7;
static constexpr double eps = 1e-8;
const double PI = acos(-1.0);

ll lT, testcase;

/*
 *
 */

struct Solve {
    ll N;
    vector<ll> A;
    vector<int> memo;

    int dfs(ll sta) {
        if (memo[sta] != -1) {
            return memo[sta];
        }
        int res = 0;
        ll xorSum = 0;
        for (int i = 0; i <= __lg(sta); ++i) {
            if (sta >> i & 1) {
                xorSum ^= A[i];
            }
        }
        for (int i = 0; i <= __lg(sta); ++i) {
            if (sta >> i & 1) {
                if (A[i] > (xorSum ^ A[i])) {
                    res |= dfs(sta - (1ll << i)) ^ 1;
                }
            }
        }
        memo[sta] = res;
        return memo[sta];
    }

    Solve() {
```

```
        cin >> N;
        A.resize(N);
        for (int i = 0; i < N; ++i) {
            cin >> A[i];
        }
        memo.resize((1ll << N) + 4, -1);
        int res=dfs((1ll << N) - 1);
        if (res) {
            cout<<"Left\n";
        }else {
            cout<<"Right\n";
        }
    }
};

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
#ifdef LOCAL
    cout.setf(ios::unitbuf); // 无缓冲流，方便我们调试
#endif

    cin >> lT;
    for (testcase = 1; testcase <= lT; ++testcase)
        Solve solve;
    return 0;
}

/*
AC
https://acm.hdu.edu.cn/contest/view-code?cid=1200&rid=13297

*/
```