백준 10749번 - Superbull

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1초	512MB	201	113	86	52.439%

문제

Bessie and her friends are playing hoofball in the annual Superbull championship, and Farmer John is in charge of making the tournament as exciting as possible. A total of N (1 <= N <= 2000) teams are playing in the Superbull. Each team is assigned a distinct integer team ID in the range 1...2^30-1 to distinguish it from the other teams. The Superbull is an elimination tournament -- after every game, Farmer John chooses which team to eliminate from the Superbull, and the eliminated team can no longer play in any more games. The Superbull ends when only one team remains.

Farmer John notices a very unusual property about the scores in matches! In any game, the combined score of the two teams always ends up being the bitwise exclusive OR (XOR) of the two team IDs. For example, if teams 12 and 20 were to play, then 24 points would be scored in that game, since 01100 XOR 10100 = 11000.

Farmer John believes that the more points are scored in a game, the more exciting the game is. Because of this, he wants to choose a series of games to be played such that the total number of points scored in the Superbull is maximized. Please help Farmer John organize the matches.

입력

The first line contains the single integer N. The following N lines contain the N team IDs.

출력

Output the maximum possible number of points that can be scored in the Superbull.

예제 입력1

```
4
3
6
9
10
```

예제 출력1

```
4
3
6
9
10
```

출처

출처

알고리즘 분류

- 그래프 이론
- 최소 스패닝 트리

접근 방법

각각의 팀들을 정점으로 보고, 두 팀이 붙게 될 경우의 Cost를 간선으로 표현할 수 있다.

N개의 팀이 붙어, 마지막 1팀이 남기 전까지 경기가 진행될 경우 총 N-1라운드의 매치가 진행된다는 것을 알 수 있다.

각각의 매치에서 서로 다른 2개의 팀이 붙고 해당 팀들의 ID를 XOR한 값이 하나의 라운드에서 얻을 수 있는 score라고 하였을 때, 전체 문제의 최적해는 최적 부분 구조를 이룬다.

(이를 증명하기 위해서는, 임의의 라운드에서 $cost(u, v) < cost(p, q)$ 인 u, v에 대하여 라운드를 진행하므로서 발생하는 모순을 보이면 된다, 귀류법)

즉, 매 라운드에서 Greedy하게 최대의 score를 이루는 팀을 선택하는 것으로 최적해를 구할 수 있다는 것이다.

단, 하나의 라운드에서 패배한 팀은 다음 라운드에 참가하지 못하므로 이에 따른 적절한 처리가 필요하다.

이 때 유용하게 쓰이는 테크닉으로는, 이미 라운드에서 만나 승/패가 결정된 팀들끼리는 하나의 **집합**으로 관리하는 것이다. 이 후에, 매 라운드에서 해당 매치 업이 성사될 수 있는지에 대한 여부를 서로 다른 집합에 속하는 원소끼리의 매치업인지를 확인하는 것으로 $O(1)$에 근사한 시간으로 확인할 수 있다. `_union함수 참고`

> 위 과정들을 정리해보았을 때, Maximum Spanning Tree를 구축하는 문제임을 알 수 있다.

소스코드

```cpp
#include <bits/stdc++.h>
using namespace std;

int parent[2001], rnk[2001];
int find(int u){
    if(u == parent[u]) return u;
    return parent[u] = find(parent[u]);
}
bool _union(int u, int v){
    u = find(u), v = find(v);
    if(u == v)
        return false;

    if(rnk[u] > rnk[v]){
        swap(u, v);
    }
    parent[u] = v;
    if(rnk[u] == rnk[v])
        rnk[v]++;

    return true;
}
typedef tuple<int, int, int> tp;
int main(void){
    int N;
    cin >> N;

    vector<int> id(N);
    for(int i=0; i<N; i++){
        cin >> id[i];
    }
    // 최적 부분 구조
    priority_queue<tp, vector<tp>> PQ;
    for(int i=0; i<N; i++){
        for(int j=i+1; j<N; j++){
            PQ.push({id[i] ^ id[j], i, j});
        }
    }

    for(int i=0; i<2001; i++)
        parent[i] = i;
    int c = N-1;
    long long ans = 0;
    while(c){
        int cost, u, v;
        tie(cost, u, v) = PQ.top(); PQ.pop();

```

```
48          if(!_union(u, v)) continue;
49          ans += cost;
50          c--;
51      }
52      cout << ans;
53      return 0;
54  }
```