Day 3: Tower Breakers, Again!

locked

by forthright48

Editorial by forthright48

This problem can be solved by calculating grundy values.

We have to calculate the grundy number for each tower. For each tower of height H, we can break them into D towers of height $\frac{H}{D}$ where D is a divisor of H and $D > H$. Then, for each possible break, we caculate the grundy number of $\frac{H}{D}$ and XOR them D times (we'll call the result X_i). For each possble D_i, we'll get one D_i. The grundy value is the Minimum Excluded number in X_i.

Next we XOR each grundy value of the towers and if it's non-negative, the first player wins; otherwise, the second player wins.

Complexity is $O(n \cdot log(n))$ because that's the sum of the divisors from 1 to n. When calculating the grundy value of a number, we iterate over its divisors. To iterate over the divisors of all numbers from 1 to n, we will need to iterate $n \cdot log(n)$ times.

Statistics

Difficulty: Medium
Time Complexity: $O(n \cdot log(n))$
Required Knowledge: Grundy Numbers
Publish Date: Mar 20 2016

Set by forthright48

Problem Setter's code :

C++

```
/***********Template Starts Here***********/
#include <bits/stdc++.h>

#define pb push_back
#define FOR(i,x,y) for(vlong i = (x) ; i <= (y) ; ++i)
#define ROF(i,x,y) for(vlong i = (y) ; i >= (x) ; --i)
#define CLR(x,y) memset(x,y,sizeof(x))
#define SZ(x) ((vlong)(x).size())

using namespace std;

typedef long long vlong;

/***********Template Ends Here***********/
#define LEN 101
#define SIZE 100001
#define HIGH 11 ///Highest Grundy Number Possible. Calculated using brute force

int arr[LEN+10];

vector<int> ddd[SIZE+10];

void precal( int n ) { ///Caculates divisors using Sieve
    FOR(i,1,n){
        for ( int j = i; j <= n; j += i ) {
            ddd[j].pb ( i );
        }
```

```
    };
}

int memo[SIZE+10];
int grundy ( int n ) {
    if ( n == 1 ) return 0;

    if ( memo[n] != -1 ) return memo[n];

    int arr[HIGH+2] = {0};

    FOR(i,0,SZ(ddd[n])-1) {
        int t = ddd[n][i]; ///Number of tower produced after breaking down

        if ( t == 1 ) continue; ///We will need to create more than one tower

        int x = n / t; ///Each tower will have height x

        int g = grundy ( x );
        if ( t & 1 ) g = g; ///If number of tower created is odd, then xor of all these t
ower will be g
        else g = 0; ///else xor will be 0

        if ( g <= HIGH ) arr[g] = 1;
    };

    FOR(i,0,HIGH){
        if ( arr[i] == 0 ) {
            return memo[n] = i;
        }
    }

    assert ( 0 );
    return 0;
}

void solution() {
    int kase;
    scanf ( "%d", &kase );

    assert ( kase >= 1 && kase <= 200 );

    while ( kase-- ) {
        int n;
        scanf ( "%d", &n );

        assert ( n >= 1 && n <= 100 );

        int res = 0;

        FOR(i,0,n-1) {
            scanf ( "%d", &arr[i] );

            assert ( arr[i] >= 1 && arr[i] <= 1000000 );

            res ^= grundy ( arr[i] ); ///Simply XOR grundy number of each tower
        };

        if ( res ) printf ( "1\n" );
        else printf ( "2\n" );
    }
}

int main () {
    precal( SIZE ); ///Calculate all divisor from 1 to N
    CLR(memo,-1); ///Memoization of grundy numbers

    solution();

    return 0;
}
```

 Tested by **allllekssssa**

Problem Tester's code :
C++

```
#include<iostream>
#include<stdio.h>
#include<algorithm>
#include<cmath>

using namespace std;

int gr[1000000];
int p,xs,x,t,n;

vector <int > v;

void grundy()
{
    gr[1]=0;

    for (int i=2;i<=100005;i++)
    {
        v.clear();
        v.push_back(-1);
        p=sqrt(i);

          for (int j=1;j<=p;j++)
            if (i%j==0)
            {
                if (j%2!=0) v.push_back(gr[i/j]);
                if ((i/j)%2!=0) v.push_back(gr[j]);
            }
      v.push_back(10000000);
      sort(v.begin(),v.end());

      for (int j=1;j<v.size();j++)
      if (abs(v[j]-v[j-1])>1)
          gr[i]=v[j-1]+1;
    }
}

int main()
{
    scanf("%d",&t);

    grundy();

    while (t--)
    {
        scanf("%d",&n);

         xs=0;
        for (int i=0;i<n;i++)
        {
            scanf("%d",&x);
            xs=xs^gr[x];
        }

        if (xs==0) printf("2\n"); else printf("1\n");
    }

    return 0;
}
```