Zadanie: NI3
Nim 3

Warsztaty ILO 2017-2018, grupa olimpijska, dzień 4. Dostępna pamięć: 256 MB.

Oznaczmy liczby z wejścia przez a_1 i a_2

Rozwiązanie brutalne $O(a_1 \cdot a_2)$

Nasze rozwiązanie będzie opierać się na programowaniu dynamicznym. Na razie nie przejmujmy się złożonością i spróbujmy po prostu rozwiązać zadanie.

Możemy za pomocą programowania dynamicznego dla każdego stanu obliczyć, czy jest stanem wygrywającym, sprawdzając czy istnieje jakieś przejście do stanu przegrywającego. Przejścia możemy łatwo rozważyć w czasie stałym.

Rozwiązanie wzorcowe $O(log(a_1)^3 \cdot log(a_2)^3)$

Obserwacja.1. *Ze stanu a_1, a_2 jest osiągalne co najwyżej $log(a_1)^3 \cdot log(a_2)^3$ innych stanów.*

Zauważmy, że nie potrzebujemy rozważać wszystkich stanów z mniejszymi wysokościami stosików, bo nie do wszystkich jesteśmy w stanie dojść podczas gry.

Dowód. Zauważmy, że zawsze, po wykonaniu pewnej liczy ruchów przejdziemy ze stanu a_1, a_2 przejdziemy do stanu $\frac{a_1}{2^{x_1} \cdot 3^{x_2} \cdot 5^{x_3}}, \frac{a_2}{2^{x_4} \cdot 3^{x_5} \cdot 5^{x_6}}$. Jednakże, żeby wysokość stosiku nie osiągnęła 0 wartość x_i musi być mniejsza od logarytmu z a_i. Dla początkowej wysokości a_i możemy dojść do co najwyżej $log(a_i)^3$ innych wysokości. $\square$

Daje to nam bardzo zgrabne rekurencyjne rozwiązanie, jedyne co pozostało to spamiętanie pośrednich wyników na haszmapie lub mapie.

```
map<pair<int, int>, int> M3;
int backtrack(pair<int, int> V)
    {
    if(M3.count(V))return M3[V];
    M3[V] = 0; // przegrywajacy

    for(auto v : {2, 3, 5})
        {
        if((V.first and !backtrack(MP(V.first / v, V.second))) or
           (V.second and !backtrack(MP(V.first, V.second / v))))
            {
            M3[V] = 1; // wygrywajacy
            return 1;
            }
        }
    return 0;
    }
```