

Gambit wieży

Limit pamięci: 512 MB



Związek Bitecki, lata 60-te. Trwa finałowy pojedynek między dwoma arcymistrzami gry w Wieżę – Algosi, młodej reprezentantki Bajtów Zjednoczonych, oraz Bajtowa, weterana gry reprezentującego gospodarzy. Zasady gry w Wieżę są następujące:

- Na początku układa się z $3n$ prostokątnych klocków ponumerowanych od 1 do $3n$ wieżę o n piętrach po trzy klocki na piętro. Klocki są ułożone w taki sposób, że są pod kątem 90 stopni względem niższego piętra (patrz obrazek).
- Gracze wykonują naprzemian ruchy polegające na wyjęciu klocka z dowolnego piętra oprócz najwyższego. Wyjęte klocki są odkładane na bok.
- Przegrywa gracz, który nie może wykonać ruchu lub spowoduje przewrócenie się wieży.

Właśnie wstrzymano partię. Twoim zadaniem jest przeanalizować pozycję, w której znalazła się Algosia.

Pierwszy etap analizy polega na określeniu **zwycięzcy**. Zarówno Algosia, jak i Bajtowa, mają za sobą lata gry i doskonałe zdolności manualne, dlatego pomijamy jakąkolwiek możliwość przypadkowego przewrócenia wieży. To oznacza, że wieża przewróci się **wtedy i tylko wtedy**, kiedy na którymś piętrze nie będzie żadnych klocków lub jest dokładnie jeden klocek i nie jest on środkowym klockiem piętra. Partię uznajemy za wygraną przez Algosię, jeśli może grać w taki sposób, że wygra niezależnie od ruchów Bajtowa (o takiej sytuacji mówimy, że Algosia ma **strategię wygrywającą**). W przeciwnym razie partię wygra Bajtowa.

Drugi etap polega na ustaleniu na ile różnych sposobów może się potoczyć reszta partii. Wykluczone jest popełnienie jakiegokolwiek pomyłki, dlatego należy policzyć tylko te partie, w których strategię wygrywającą utrzyma osoba, którą określiliśmy w pierwszym etapie jako zwycięzcę. Formalnie, jeśli gracz A został określony jako zwycięzca, to w trakcie gry nie będzie takiej sytuacji, że B będzie miał strategię wygrywającą. Naszym zadaniem jest policzenie, ile jest takich kontynuacji gry, że zwycięzca nigdy nie traci swojej wygrywającej pozycji. Zakładamy, że **jeśli gracz może wykonać ruch, który nie przewraca wieży, to go wykona**. Przebieg gry określa się za pomocą **notacji**, czyli listy, na której numer klocka wyjętego w i -tym ruchu jest na i -tej pozycji. **Nie zapisujemy w notacji ruchu, który spowodował przewrócenie wieży**. Dwie gry są uznawane za różne jeśli mają inne notacje.

Twój ruch.

Gambit wieży

Limit pamięci: 512 MB

Wejście

W pierwszym wierszu wejścia znajduje się liczba całkowita $1 \leq n \leq 5000$, oznaczająca liczbę pięter wieży.

W każdym z następnych n wierszy znajdują się trzy liczby reprezentujące ułożenie klocków na piętrach w kolejności od najwyższego do najniższego. W $i + 1$ -tym wierszu wyjścia jest opis $n + 1 - i$ -tego piętra. Liczba 0 oznacza, że w danym miejscu nie ma klocka, a 1, że jest. Gwarantowane jest, że na każdym piętrze jest co najmniej jeden klocek, a jeśli jest tylko jeden, to jest to środkowy klocek piętra.

Wyjście

W pierwszym wierszu wyjścia wypisz A jeśli zwycięzcą będzie Algosia, lub B jeśli wygra Bajtow. W drugim wierszu wyjścia wypisz liczbę możliwych notacji, jakie może mieć reszta partii. Ponieważ ta liczba może być bardzo duża, wypisz ją modulo $10^9 + 7$.

Jeżeli wypiszesz poprawnie jedynie pierwszy wiersz, otrzymasz 15% punktów możliwych do uzyskania za dany test.

Przykłady

Wejście dla testu r4d0a:

```
4
1 1 1
1 0 1
1 1 1
0 1 1
```

Wyjście dla testu r4d0a:

```
A
4
```

Wyjaśnienie: W pierwszym ruchu Algosia wyjmie lewy lub prawy klocek z drugiego piętra. Bajtow wyjmie wtedy klocek z pierwszego lub drugiego piętra. Wtedy zostaje Algosi jeden klocek, który może wyjąć, zatem wygrywa. Są 4 możliwe sposoby dokończenia tej gry.

Ocenianie

Podzadanie	Ograniczenia	Limit czasu	Punkty
1	$n = 2$	4 s (C++) / 70 s (Python)	5
2	Na każdym piętrze (poza najwyższym) jest dokładnie jeden klocek	4 s (C++) / 70 s (Python)	5
3	Na żadnym piętrze (poza najwyższym) nie ma trzech klocków	4 s (C++) / 70 s (Python)	5
4	$n \leq 10$	4 s (C++) / 70 s (Python)	10
5	Gra się dopiero zaczęła, na każdym piętrze wieży są 3 klocki	4 s (C++) / 70 s (Python)	20
6	$n \leq 100$	4 s (C++) / 70 s (Python)	20
7	Brak dodatkowych ograniczeń	4 s (C++) / 70 s (Python)	35