Game of Bugs

Limit pamięci: 64 MB

Conway wymyślając swoją "Grę w życie"[1], próbował wiele podobnych wariantów automatów komórkowych. Jednym z nich była mało znana "Game of Bugs"[2]. Jednym z powodów, z których "Game of Bugs" się nie przyjęła może być fakt, że jej pierwsza implementacja była była zbugowana.

"Game of Bugs" podobnie jak "Game of life" polega na tym, że na nieskończonej planszy złożonej z jednostkowych kwadratów, część z nich jest zamalowana, a część jest pusta. Na początku gracz ręcznie wybiera które pola zamalować, a których nie. Później zmienia się to automatycznie (w jednej iteracji wszystkie pola na raz) według następującej zasady:

Jeżeli w kwadracie 3x3 dookoła danego pola (łącznie ze środkiem) znajduje się parzysta liczba zamalowanych pól, to w następnej iteracji to pole będzie puste, w przeciwnym wypadku będzie zamalowane.

Niestety, tak jak wcześniej wspomniałem pierwszy program który symulował "Game of Bugs" był zbugowany, więc w ciągu każdej iteracji mógł, **ale nie musiał**, wpisać w **jednym** polu złą wartość.

Bajtek prowadził ostatnio badania naukowe korzystając z tej implementacji "Game of Bugs" nie wiedząc że jest zbugowana i teraz zastanawia się, jak bardzo ten błąd mógł wpłynąć na jego wyniki. Dla danego stanu końcowego powiedz Bajtkowi z ile różnych niepustych stanów początkowych mogło do niego doprowadzić, zakładając że były symulowane przez zbugowaną implementację opisaną wyżej.

2 stany są różne wtedy i tylko wtedy gdy istnieje takie pole, które w jednym z nich jest zamalowane, a w drugim nie. Stan jest pusty wtedy i tylko wtedy gdy żadne pole nie jest zamalowane.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby całkowite n i m $(1 \leq n, m \leq 300)$, oznaczające wymiary prostokąta opisującego stan końcowy "Game of Bugs". W następnych n wierszach znajdują się napisy złożone z m znaków "#" lub "̇" , jeżeli w i-tym wierszu j-ty znak to #, oznacza to że pole o współrzędnych i, j jest zamalowane. **Każde** pole które nie jest oznaczone znakiem # jest puste, łącznie z nieskończoną liczbą pól dookoła podanego prostokąta. Możesz założyć ża na wejściu będzie przynajmniej jeden znak #.

Wyjście

W jedynym wierszu wyjścia podaj jedną liczbę całkowitą - liczbę różnych niepustych stanów początkowych, które mogły doprowadzić do danego stanu końcowego. Ponieważ może ona być bardzo duża, wynik podaj modulo $1\,000\,000\,007$. Jeżeli istnieje nieskończenie wiele niepustych stanów początkowych które mogły doprowadzić do podanego stanu końcowego zamiast liczby wypisz "inf".

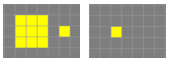
Przykłady

Wejście dla testu `r5e0`:

```
3 5
###..
###.#
###..
```

Wyjście dla testu `r5e0`:

```
2
```

Wyjaśnienie: Możliwe stany początkowe to:  W pierwszym dostajemy stan końcowy po zeru iteracjach, a w drugim po jednej, zakładając że maszyna się pomyliła i dodała jedno zamalowane pole po prawej.

[1] Game of life - `https://pl.wikipedia.org/wiki/Gra_w_życie`
[2] Jest to fikcyjna nazwa, wymyślona na rzecz zadania

Game of Bugs

Limit pamięci: 64 MB

Ocenianie

Podzadanie	Ograniczenia	Limit czasu	Punkty
1	Na wejściu jest dokładnie jeden znak #	1 s (C++) / 4 s (Python)	2
2	$n,mleq3$	1 s (C++) / 4 s (Python)	9
3	$n \leq 2$	1 s (C++) / 4 s (Python)	13
4	$n,m \leq 10$	1 s (C++) / 4 s (Python)	29
5	Brak dodatkowych ograniczeń	1 s (C++) / 4 s (Python)	47