Една типична слот машина има:

- ***Рейлове**: вертикални колони, които се въртят независимо (напр. 5 рейла).*
- ***Редове**: хоризонтални позиции във всеки рейл (напр. 3 или 4 реда).*
- ***Символи**: картинки/икони във всяка клетка.*

Линия е предварително зададено множество от позиции (по една от всеки рейл), което се проверява за съвпадащи символи.

Задача 1. (*playtech*) Дадена е слот машина с екран с размерност 5 x 3 (рейлове x редове). Дефинираме следното правило: две клетки от два съседни рейла участват в една линия, ако имат съседна стена или общ ръб. Да се изчисли колко различни линии има слот машината с така дефинираното правило.

Решение.

Това което искаме да постигнем е да пресметнем по колко начина може да достигнем до някоя от клетките на последния рейл започвайки от някоя от клетките на първия рейл и да ги съберем. Това ще е еквивалентно на нашия отговор.

Номерираме редовете: 1 (горен), 2 (среден), 3 (долен). Това са и нашите начални състояния при генерирането на линии.

Разглеждаме даденото правило. Възможните преходи са следните:

- От ред 1 → {1, 2}
- От ред 2 → {1, 2, 3}
- От ред 3 → {2, 3}

Конструираме матрица на прехода $M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$.

$$M[i,j] = \begin{cases} 1, & \text{ако може да преминем от ред към ред от един рейл към следващия.} \\ 0, & \text{ако преходът е невъзможен.} \end{cases}$$

Може лесно да подходим рекурсивно и да извлечем рекурсивна формула. Нека

$$a_n(r) = \text{брой начини по които може да достигнем ред } r \text{ на рейл } n.$$

Имаме, $\begin{cases} a_{n+1}(1) = a_n(1) + a_n(2) \\ a_{n+1}(2) = a_n(1) + a_n(2) + a_n(3), \\ a_{n+1}(3) = a_n(2) + a_n(3) \end{cases}$ с начални стойности: $a_1(1) = a_1(2) = a_1(3) = 1.$

Имайки тази рекурсилна формула, ще е достатъчно да пресметнем a_5 с всички 3 аргумента.

$$\begin{cases} \text{Рейл 2:} \\ a_2(1) = 1 + 1 = 2 \\ a_2(2) = 1 + 1 + 1 = 3; \\ a_2(3) = 1 + 1 = 2 \\ \rightarrow \mathbf{a_2} = [\mathbf{2}, \mathbf{3}, \mathbf{2}] \end{cases} \begin{cases} \text{Рейл 3:} \\ a_3(1) = 2 + 3 = 5 \\ a_3(2) = 2 + 3 + 2 = 7; \\ a_3(3) = 3 + 2 = 5 \\ \rightarrow \mathbf{a_3} = [\mathbf{5}, \mathbf{7}, \mathbf{5}] \end{cases} \begin{cases} \text{Рейл 4:} \\ a_4(1) = 5 + 7 = 12 \\ a_4(2) = 5 + 7 + 5 = 17; \\ a_4(3) = 7 + 5 = 12 \\ \rightarrow \mathbf{a_4} = [\mathbf{12}, \mathbf{17}, \mathbf{12}] \end{cases} \begin{cases} \text{Рейл 5:} \\ a_5(1) = 12 + 17 = 29 \\ a_5(2) = 12 + 17 + 12 = 41. \\ a_5(3) = 17 + 12 = 29 \\ \rightarrow \mathbf{a_5} = [\mathbf{29}, \mathbf{41}, \mathbf{29}] \end{cases}$$

Общо брой линии = 29 + 41 + 29 = **99**. □

Допълнително.

Матрицата на преходите M има следното свойство:

$$M[i,j] = \text{брой начини да стигнеш от ред } i \text{ в рил с с до ред } j \text{ в рейл } c+1$$

Може да комбинираме пътищата когато умножаваме матрици, тъй като:

$$(M^2)[i,j] = \sum_k M[i,k] \cdot M[k,j]$$

Това означава, че броят начини да стигнеш от i до j за 2 стъпки (т.е. през 1 междинен рейл) е сумата от начините: от i до k (1 стъпка) и от k до j (1 стъпка). Така M^2 дава броя пътища от рейл 1 до рейл 3.

Обобщение за N рейла.

Искаме пътища от рейл 1 до рейл N, което е $N-1$ стъпки (прехода).

$$M^{N-1}[i,j] = \text{брой пътища от ред } i \text{ в рейл 1 до ред } j \text{ в рейл } N$$

Начално разпределение. В рейл 1 имаме по 1 възможност за всеки ред: $v_1 = [\underbrace{1,\ldots,1}_{m}]$. Това може да се разгледа като „брой пътища да стигнеш до ред i в рил 1“ (в началото имаме 1 път за всеки ред или казано в контекста на нашата задача - по 1 линия за всяка клетка от m-те клетки в първия рейл).

Умножение с началния вектор.

$$v_N = v_1 \cdot M^{N-1}$$

е вектор, където

$$v_N[j] = \text{брой пътища (линии) от някой ред в рейл 1 до ред } j \text{ в рейл } N.$$

Сумата по j дава общия брой линии.

Използваме M^{N-1}, а не M^N, тъй като при N рейла имаме точно $N-1$ прехода от рейл 1 до рейл N.

Отговор: $v_1 M^{N-1} \overset{N=5}{=} [1,1,1] \times \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}^4 = [29,41,29]$, което са общо **99** линии.

Този метод ни дава логаритмична времева сложност за изчисление.

```python
def count_slot_lines_matrix_power(reels, rows):
    """
    Calculate number of lines using matrix exponentiation.
    Time Complexity: O(rows^3 * log(reels))
    """
    if reels < 1 or rows < 1:
        return 0
    if reels == 1:
        return rows
    M = build_transition_matrix(rows)
    power = reels - 1
    # Compute M^(reels-1)
    M_power = matrix_power(M, power)
    # Initial vector [1, 1, ..., 1]
    initial = [1] * rows
    # Multiply initial vector by result matrix
    final = [0] * rows
    for j in range(rows):
        for i in range(rows):
            final[j] += initial[i] * M_power[i][j]
    return sum(final)
```