У тебе є glass (склянки) і floors (поверхи). Треба знайти найменшу кількість кидків, яка гарантовано дозволяє дізнатися, з якого поверху склянка почне розбиватися.

**Як працює алгоритм**

dp[g][f] означає мінімальна кількість кидків, яка гарантує знайдення критичного поверху при g склянках і f поверхах.

Якщо тільки 1 склянка (g = 1), то треба перевіряти по одному поверху знизу вверх — бо ризик розбити останню склянку. Тобто:

dp[1][f] = f

Якщо 0 поверхів — нічого не треба робити:

dp[g][0] = 0

Якщо 1 поверх — один кидок максимум:

dp[g][1] = 1

**Основна ідея перебору**

У циклі ми перебираємо x — поверх, з якого кидаємо склянку якщо вона розбилась: значить, шукаємо відповідь нижче (поверхи 1 до x-1), і вже маємо g-1 склянок dp[g-1][x-1]

Якщо не розбилась шукаємо поверх вище (від x+1 до f), і склянок лишилось стільки ж dp[g][f - x]

Максимум з цих двох — це найгірший варіант

res = 1 + max(dp[g - 1][x - 1], dp[g][f - x])

Ми беремо мінімум з усіх x — тобто вибираємо найкращий з найгірших сценаріїв (так звана стратегія мінімаксу):

dp[g][f] = min(dp[g][f], res)

**Чому менше не може бути?**

Ми розглядаємо всі можливі варіанти x — тобто всі можливі поверхи, з яких можна кинути склянку. І для кожного з них обраховуємо гарантований максимум спроб у найгіршому випадку.

Алгоритм зберігає мінімум з цих максимумів. Тобто жодна стратегія не дасть кращого результату в найгіршому випадку — бо ми обрали найкращий із найгірших сценаріїв.

Якщо б існував кращий варіант — тобто менше кидків — він би був врахований у min(). Але ми перебрали всі x, отже жодного кращого варіанту не пропустили.

**Приклад**

Для 2 склянок і 10 поверхів

Кидаємо з 4-го поверху:

Якщо розбилась: 1 склянка, 3 поверхи → 3 спроби

Якщо не розбилась: 2 склянки, 6 поверхів → 3 спроби

Разом: 1 + max(3,3) = 4

Перебравши всі варіанти, знаходимо, що 4 — це мінімум, що гарантує результат.