Wafer Cutting

DiPS CodeJam 23

Prompt

When manufacturing chips, circular silicon wafers are cut into dies of required size. The goal of this challenge is to maximise the number of whole dies of dimensions l_x, l_y that can be cut from a wafer of a given diameter d. Your task is to find the maximum number of dies that can be cut from a wafer.
The machine measures a distance g_x from the left edge, and makes a vertical cut. Then it makes the needed number of additional vertical cuts, with spacing of the needed width. It makes the horizontal cuts in the same way: first one at a distance g_y, and then with spacing of height.
The material remains perfectly still during the cutting: it's not allowed to move some slices of material before making the perpendicular cuts.
The machine has a stepper motor that works at absolute accuracy but can only go integer-sized distances.

Input Format

- The first line of the input contains one integer d that denotes the diameter of the wafer.
- The second line contains 2 space-separated integers l_x and l_y that denote the width and height of one die respectively.

Output Format

Your output should contain a single integer w denoting the maximum number of dies that can be cut.

Constraints

$0 < d \leq 450$

Sample Input/Output

Input	Output
8 2 2	6

Solution

The challenge can be solved as follows:

- For each possible g_x value:
  - For each possible g_y value:
    - Assume the starting die count value is 0.
    - For every die position, check if the die fits on the board.
    - If the die fits, increment the die count.

  - If the current die count is greater than the greatest die count obtained previously, update the maximum value.

Sample Program

```
d = int(input())
l_x, l_y = list(map(int, input().strip().split(" ")))

max_die_count = 0
best_gx = 0
best_gy = 0
radius = d // 2

for g_x in range(l_x):
  for g_y in range(l_y):
    die_count = 0
    for x1 in range(g_x - radius, radius, l_x):
      x2 = x1 + l_x
      for y1 in range(g_y - radius, radius, l_y):
        y2 = y1 + l_y
        if (x1 * x1 + y1 * y1 <= radius**2 and
          x2 * x2 + y2 * y2 <= radius**2 and
          x1 * x1 + y2 * y2 <= radius**2 and
          x2 * x2 + y1 * y1 <= radius**2):
          die_count += 1
    if die_count > max_die_count:
      max_die_count = die_count
      best_gx = g_x
      best_gy = g_y

print(max_die_count)
```