Vamshi Jandhyala

Can You Fit the Stars on the Flag?

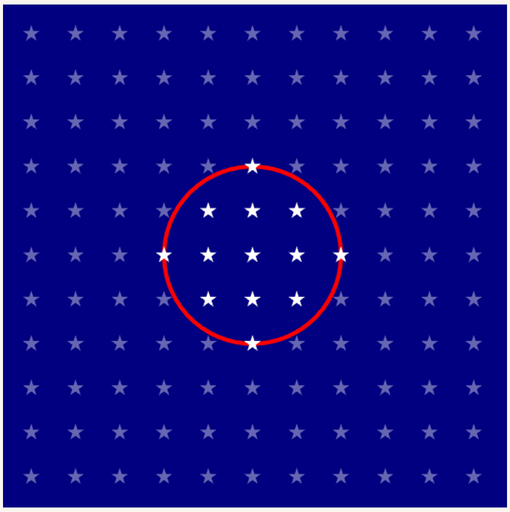
July 2026

A lattice-points-in-a-disk puzzle from The Fiddler.[1]

Problem

When Retsy Boss designed her new nation's flag, she wanted the stars packed in tightly. The stars sit at the points of a square grid, one unit apart, and she kept only those whose centres lay within 2 units of a single chosen point on the plane. Centred on a star, that rule admits 13 stars. Placed freely, what is the greatest number of stars the rule can admit?

For extra credit, 250 years on, the flag must carry one star for each of the nation's 58 states. As before, the chosen stars must all lie within some distance R of a single point. What is the smallest R for which the grid offers up 58 such stars?



Centred on a star, a disk of radius 2 *admits* 13 *grid points, four of them exactly on the rim.*

Solution

The rule keeps every grid point inside a disk of radius 2, and the puzzle lets the disk float wherever it likes. Centring it on a star is the obvious move and the wrong one.

[1] Zach Wissner-Gross, *The Fiddler*, 3 July 2026, thefiddler.substack.com.

It catches the four points at distance exactly 2, on the axes, but it wastes those four on the rim, where the slightest crowding would spill them out. A disk of radius 2 has area $4\pi \approx 12.57$, so 13 points is already a point above what the area alone would suggest, and the four rim points are the reason. The question is whether floating the disk between grid lines can do better than balancing four points on the edge.

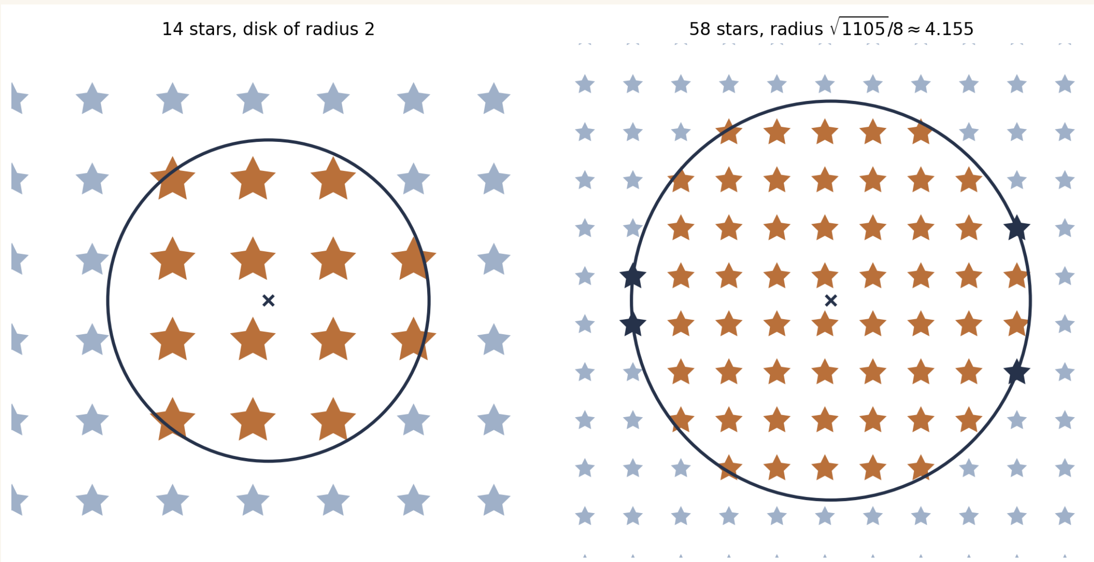
It can. Slide the disk so its centre sits near $\left(\frac{1}{5}, \frac{1}{2}\right)$, halfway up between two rows and just off a column, and it swallows a solid 3×4 block of 12 points, the columns $x \in \{-1, 0, 1\}$ across the rows $y \in \{-1, 0, 1, 2\}$, with room to spare on one side for two more, $(2, 0)$ and $(2, 1)$. That is 14.

To see that all 14 really fit, put the centre at $\left(c, \frac{1}{2}\right)$, level with the middle of the block. The two points reaching farthest to the right, $(2, 0)$ and $(2, 1)$, sit at distance $\sqrt{(2-c)^2 + \frac{1}{4}}$, and the two reaching farthest left, $(-1, -1)$ and $(-1, 2)$, at distance $\sqrt{(c+1)^2 + \frac{9}{4}}$. Both stay within 2 precisely when

$$2 - \frac{\sqrt{15}}{2} \le c \le \frac{\sqrt{7}}{2} - 1, \qquad \text{that is} \qquad 0.063 \le c \le 0.323,$$

a genuine interval, and the other ten points are nearer still. So a whole range of placements holds 14 at once. No placement holds 15: the count changes only as the centre crosses one of the circles of radius 2 drawn about the grid points, these circles cut the plane into finitely many cells with the count constant on each, and the pattern repeats with every unit square, so a sweep of one unit square settles the maximum. That sweep never reaches 15. The greatest number of stars is

$$\boxed{14}.$$



Left: an off-centre disk of radius 2 admits 14 grid points (copper), a 3×4 block plus two more. Right: the smallest disk holding 58 points has radius $\sqrt{1105}/8 \approx 4.155$ about $\left(\frac{1}{8}, \frac{1}{2}\right)$; four points (navy) lie exactly on its rim.

Extra credit

Now the target is a count, 58 stars, and the prize is the smallest disk that reaches it. For a fixed centre P, the tightest disk holding 58 points has radius equal to the distance from P to its 58th-nearest grid point, so the task is to choose P to bring that 58th point as close as possible.

Centre P on a star and the grid points arrive in rings by squared distance $x^2 + y^2 = 0, 1, 2, 4, 5, \ldots$. The counts accumulate to 57 points out to squared distance 17, and the next ring, the four points with $x^2 + y^2 = 18$ such as $(3, 3)$, brings the total past 58. So a star-centred disk needs radius $\sqrt{18} = 3\sqrt{2} \approx 4.243$, and it holds the 58th star only by stretching to that ring.

Floating the disk does better here too. The best centre is $P = \left(\frac{1}{8}, \frac{1}{2}\right)$, off a column by an eighth and midway between two rows. Its four farthest admitted points are $(-4, 0)$, $(-4, 1)$, $(4, -1)$ and $(4, 2)$, all at the same squared distance

$$\left(4 - \tfrac{1}{8}\right)^2 + \left(\tfrac{1}{2}\right)^2 = \frac{961}{64} + \frac{16}{64} = \frac{1105}{64},$$

so they sit together on one rim, and inside that rim lie exactly 58 points. The minimum distance is therefore

$$R = \sqrt{\frac{1105}{64}} = \frac{\sqrt{1105}}{8} = \boxed{\frac{\sqrt{1105}}{8} \approx 4.155.}$$

Trading the star-centred disk for this one shaves the radius from $3\sqrt{2} \approx 4.243$ down to 4.155, just enough to matter. The official solution had not yet been published at the time of writing, so both headline values are my own, confirmed by the exhaustive sweep below.

Python code

Both parts are a search over where to put the centre, and the count is a step function of the centre that repeats with the unit square, so a fine sweep of one unit square is exhaustive. For the main puzzle the sweep records the most grid points a radius-2 disk can hold; for the extra credit it records, at each centre, the distance to the 58th-nearest grid point, and keeps the smallest.

```
import math

# MAIN: maximise the number of grid points in a CLOSED disk of radius 2.
# The count repeats with the unit square, so sweeping [0,1)^2 is exhaustive.
def count_in_disk(cx, cy, R=2.0):
    r2, c = R*R + 1e-9, 0
    for i in range(math.floor(cx-R)-1, math.ceil(cx+R)+2):
        for j in range(math.floor(cy-R)-1, math.ceil(cy+R)+2):
            if (i-cx)**2 + (j-cy)**2 <= r2:
                c += 1
    return c

best = max(count_in_disk(i/400, j/400) for i in range(401) for j in range(401))
print("MAIN: most stars in a radius-2 disk =", best)
```

```
# EC: minimise the radius that admits 58 points = min over centres of the
# 58th-nearest grid distance.
def kth_distance(cx, cy, k=58):
    d = sorted((i-cx)**2 + (j-cy)**2 for i in range(-6, 7) for j in range(-6, 7))
    return d[k-1]                # squared distance to the k-th nearest point

bestR2 = min(kth_distance(i/400, j/400) for i in range(401) for j in range(201))
print(f"EC: min radius = sqrt({bestR2:.6f}) = {math.sqrt(bestR2):.6f}")
print("   R^2 = 1105/64 =", 1105/64)
print("   R   = sqrt(1105)/8 =", math.sqrt(1105)/8)
# MAIN: most stars in a radius-2 disk = 14
# EC: min radius = sqrt(17.265625) = 4.155193
#     R^2 = 1105/64 = 17.265625
#     R   = sqrt(1105)/8 = 4.155192534648665
```