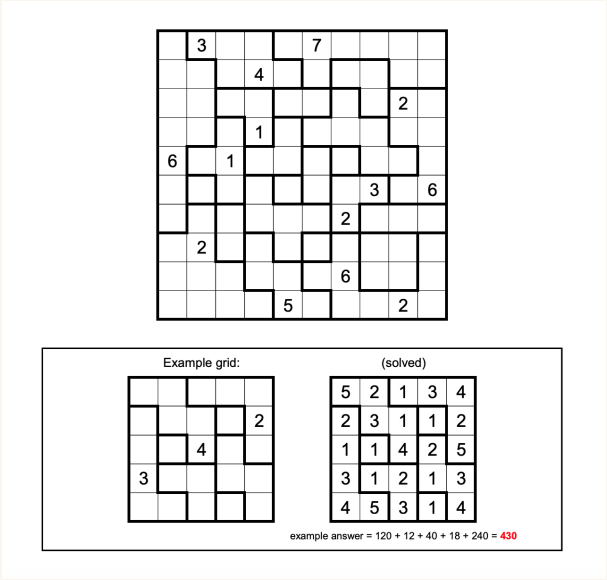
Vamshi Jandhyala

Block Party 4

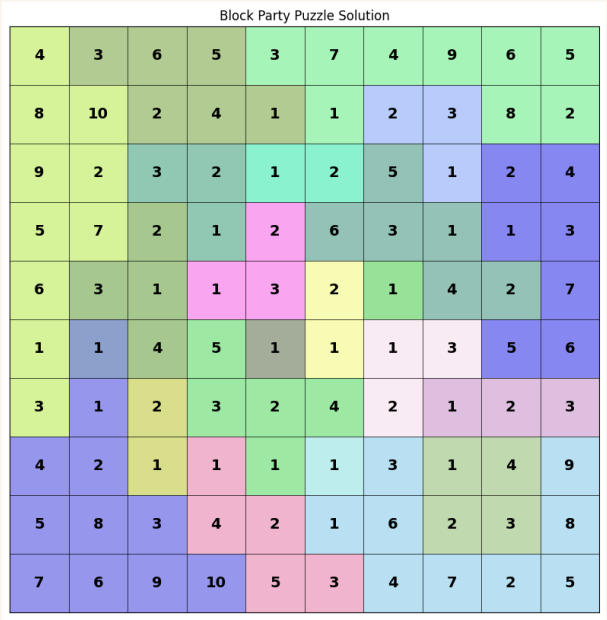
✠

2024



Fill each region with the numbers 1 through N, where N is the number of cells in the region. For each number K in the grid, the nearest K via taxicab distance must be exactly K cells away.

Solution



Python code

```
from z3 import *
import matplotlib.pyplot as plt
import random

def solve_block_party(grid):
    solver = Solver()
    n = 10
    cells = [[Int(f"cell_{i}_{j}") for j in range(n)] for i in range(n)]

    for item in grid:
        if isinstance(item, tuple):
            i, j, value = item
            solver.add(cells[i][j] == value)
        elif isinstance(item, list):
            size = len(item)
            solver.add(Distinct([cells[x][y] for x, y in item]))
            for x, y in item:
                solver.add(And(1 <= cells[x][y], cells[x][y] <= size))

    for i in range(n):
        for j in range(n):
            k = cells[i][j]

            solver.add(And([
                Implies(cells[x][y] == k,
                        abs(x - i) + abs(y - j) >= k)
                for x in range(n) for y in range(n)
                if (x != i or y != j)
            ]))

            solver.add(Or([
```

```
                And(cells[x][y] == k, abs(x - i) + abs(y - j) == k)
                for x in range(n) for y in range(n)
                if (x != i or y != j)
            ]))

    for i in range(n):
        for j in range(n):
            solver.add(And(1 <= cells[i][j], cells[i][j] <= 10))

    if solver.check() == sat:
        model = solver.model()
        return [[model.evaluate(cells[i][j]).as_long() for j in range(n)]
                for i in range(n)]
    return None

grid = [
    (0, 1, 3), (0, 5, 7), (2, 8, 2),
    (1, 3, 4), (3, 3, 1), (4, 0, 6), (4, 2, 1),
    (5, 7, 3), (5, 9, 6), (6, 6, 2), (7, 1, 2),
    (8, 6, 6), (9, 4, 5), (9, 8, 2),

    [[0,0],[1,0],[1,1],[2,0],[2,1],[3,0],[3,1],[4,0],[5,0],[6,0]],
    [[0,1],[0,2],[0,3],[1,2],[1,3],[1,4]],
    [[0,4],[0,5],[0,6],[0,7],[0,8],[0,9],[1,5],[1,8],[1,9]],
    [[1,6],[1,7],[2,7]],
    [[2,2],[2,3],[3,3]],
    [[2,4],[2,5]],
    [[2,6],[3,5],[3,6],[3,7],[4,7],[4,8]],
    [[3,2],[4,1],[4,2],[5,2]],
    [[5,1]],
    [[3,4],[4,3],[4,4]],
    [[2,8],[2,9],[3,8],[3,9],[4,9],[5,8],[5,9]],
    [[4,6]],
    [[5,4]],
    [[4,5],[5,5]],
    [[5,3],[6,3],[6,4],[6,5],[7,4]],
    [[5,6],[5,7],[6,6]],
    [[6,7],[6,8],[6,9]],
    [[6,2],[7,2]],
    [[6,1],[7,0],[8,0],[9,0],[7,1],[8,1],[9,1],[8,2],[9,2],[9,3]],
    [[7,3],[8,3],[8,4],[9,4],[9,5]],
    [[7,5]],
    [[7,6],[8,5],[8,6],[9,6],[9,7],[9,8],[9,9],[8,9],[7,9]],
    [[7,7],[7,8],[8,7],[8,8]],
]

solution = solve_block_party(grid)
```