Pizza Slice Instant Insanity

Behrouz Barati B

1 Problem Parameters

- Name: Behrouz Barati B
- First letter of last name: B (position 2)
- $N = 100 + 2 = 102$
- Generator: $1 + \lfloor e \cdot n \rfloor \mod N$ for $n = 1, \ldots, N$
- Skip rule: Skip value once it has appeared 3 times

2 Pseudocode

Algorithm 1 Generate Sequences

```
function GenerateRaw(N)
    raw ← []
    for n = 1 to N do
        val ← 1 + ⌊e · n⌋ mod N
        Append val to raw
    end for
    return raw
end function
```

Algorithm 2 Reduce Sequence

```
function Reduce(raw)
    counts ← {}
    reduced ← []
    for each val in raw do
        if counts[val] < 3 then
            Append val to reduced
            counts[val] ← counts[val] + 1
        end if
    end for
    return reduced
end function
```

Algorithm 3 Solve Pizza Slice Instant Insanity

```
function ROTATIONS((a, b, c))
    return [(a, b, c), (b, c, a), (c, a, b)]
end function

function SOLVE(slices)
    cols ← [∅, ∅, ∅]                                  ▷ Track colors in each column
    return BACKTRACK(0, cols, slices)
end function

function BACKTRACK(idx, cols, slices)
    if idx = |slices| then
        return []                                      ▷ Solution found
    end if
    for each rotation r in ROTATIONS(slices[idx]) do
        if r[j] ∉ cols[j] for all j ∈ {0, 1, 2} then
            for j = 0 to 2 do
                cols[j] ← cols[j] ∪ {r[j]}
            end for
            result ← BACKTRACK(idx+1, cols, slices)
            if result ≠ None then
                return [(idx, r)] + result
            end if
            for j = 0 to 2 do
                cols[j] ← cols[j] \ {r[j]}
            end for
        end if
    end for
    return None
end function
```

Algorithm 4 Find Minimal Obstacle

```
function MINIMALOBSTACLE(slices)
    for size = 2 to |slices| do
        for each combination C of size slices do
            if not SOLVE(C) then
                return C                               ▷ Minimal unsolvable subset
            end if
        end for
    end for
    return None
end function
```

3 Python Code

```python
import math
```

```
from itertools import combinations

e = math.e
N = 102

raw_seq = [1 + int(e * n) % N for n in range(1, N + 1)]

print("RAW SEQUENCE:")
print(raw_seq)

counts = {}
reduced_seq = []
for val in raw_seq:
    if counts.get(val, 0) < 3:
        reduced_seq.append(val)
        counts[val] = counts.get(val, 0) + 1

print("\nREDUCED SEQUENCE:")
print(reduced_seq)

slices = [tuple(reduced_seq[i*3:(i+1)*3]) for i in range(len(reduced_seq)
   //3)]
print(f"\nPIZZA SLICES ({len(slices)}):")
for i, s in enumerate(slices):
    print(f"  {i}: {s}")

def rotations(t):
    return [(t[0],t[1],t[2]), (t[1],t[2],t[0]), (t[2],t[0],t[1])]

def solve(slices):
    n = len(slices)
    def bt(idx, cols):
        if idx == n:
            return []
        for r in rotations(slices[idx]):
            if all(r[j] not in cols[j] for j in range(3)):
                for j in range(3):
                    cols[j].add(r[j])
                rest = bt(idx+1, cols)
                if rest is not None:
                    return [(idx, r)] + rest
                for j in range(3):
                    cols[j].remove(r[j])
        return None
    return bt(0, [set(), set(), set()])

def minimal_obstacle(slices):
    for sz in range(2, len(slices)+1):
        for combo in combinations(range(len(slices)), sz):
            sub = [slices[i] for i in combo]
            if not solve(sub):
                return combo, sub
    return None, None

```

```
sol = solve(slices)
if sol:
    print("\nSOLUTION:")
    for idx, triple in sol:
        print(f"  {triple}")
else:
    print("\nNO SOLUTION")
    obs_idx, obs = minimal_obstacle(slices)
    if obs_idx:
        print(f"MINIMAL OBSTACLE: {obs_idx}")
        for i, s in zip(obs_idx, obs):
            print(f"  Slice {i}: {s}")
```

4 Results

4.1 Raw Sequence (Length 102)

```
[3, 6, 9, 11, 14, 17, 20, 22, 25, 28, 30, 33, 36, 39, 41, 44, 47, 49,
 52, 55, 58, 60, 63, 66, 68, 71, 74, 77, 79, 82, 85, 87, 90, 93, 96,
 98, 101, 2, 5, 7, 10, 13, 15, 18, 21, 24, 26, 29, 32, 34, 37, 40, 43,
 45, 48, 51, 53, 56, 59, 62, 64, 67, 70, 72, 75, 78, 81, 83, 86, 89,
 91, 94, 97, 100, 102, 3, 6, 9, 11, 14, 17, 19, 22, 25, 28, 30, 33,
 36, 38, 41, 44, 47, 49, 52, 55, 57, 60, 63, 66, 68, 71, 74]
```

4.2 Reduced Sequence (Length 102)

Same as raw sequence (no value exceeded 3 occurrences before end).

4.3 Pizza Slices (34 slices)

```
 0: (3, 6, 9)       1: (11, 14, 17)    2: (20, 22, 25)    3: (28, 30, 33)
 4: (36, 39, 41)    5: (44, 47, 49)    6: (52, 55, 58)    7: (60, 63, 66)
 8: (68, 71, 74)    9: (77, 79, 82)   10: (85, 87, 90)   11: (93, 96, 98)
12: (101, 2, 5)    13: (7, 10, 13)    14: (15, 18, 21)   15: (24, 26, 29)
16: (32, 34, 37)   17: (40, 43, 45)   18: (48, 51, 53)   19: (56, 59, 62)
20: (64, 67, 70)   21: (72, 75, 78)   22: (81, 83, 86)   23: (89, 91, 94)
24: (97, 100, 102) 25: (3, 6, 9)      26: (11, 14, 17)   27: (19, 22, 25)
28: (28, 30, 33)   29: (36, 38, 41)   30: (44, 47, 49)   31: (52, 55, 57)
32: (60, 63, 66)   33: (68, 71, 74)
```

4.4 Solution (Column of Triples)

A solution exists. Each triple is rotated so no color repeats in any column position.

```
  (3, 6, 9)
  (11, 14, 17)
  (20, 22, 25)
  (28, 30, 33)
  (36, 39, 41)
  (44, 47, 49)
```

```
(52, 55, 58)
(60, 63, 66)
(68, 71, 74)
(77, 79, 82)
(85, 87, 90)
(93, 96, 98)
(101, 2, 5)
(7, 10, 13)
(15, 18, 21)
(24, 26, 29)
(32, 34, 37)
(40, 43, 45)
(48, 51, 53)
(56, 59, 62)
(64, 67, 70)
(72, 75, 78)
(81, 83, 86)
(89, 91, 94)
(97, 100, 102)
(6, 9, 3)
(14, 17, 11)
(22, 25, 19)
(30, 33, 28)
(38, 41, 36)
(47, 49, 44)
(55, 57, 52)
(63, 66, 60)
(71, 74, 68)
```