29. Write a python program for SHA-3 option with a block size of 1024 bits and assume that each of the lanes in the first message block (P0) has at least one nonzero bit. To start, all of the lanes in the internal state matrix that correspond to the capacity portion of the initial state are all zeros. Show how long it will take before all of these lanes have at least one nonzero bit. Note: Ignore the permutation. That is, keep track of the original zero lanes even after they have changed position in the matrix.

Code:

```
#!/usr/bin/env python3
"""Simulate Keccak lane permutation (π) to see when capacity lanes
(indices 16..24) become nonzero starting from nonzero lanes 0..15.
Lanes are indexed as idx = x + 5*y for x,y in {0..4}.
π maps (x,y) -> (y, (2*x + 3*y) mod 5).
"""
from math import isqrt
def idx_to_xy(idx):
    x = idx % 5
    y = idx // 5
    return x, y
def xy_to_idx(x, y):
    return x + 5 * y
def pi_map(idx):
    x, y = idx_to_xy(idx)
    x_new = y
    y_new = (2 * x + 3 * y) % 5
    return xy_to_idx(x_new, y_new)
def simulate_activation(initial_nonzero, capacity_lanes):
    # activation_step[i] = step when lane i first became nonzero (None if not yet)
    activation_step = {i: None for i in range(25)}
    for i in initial_nonzero:
```

```
        activation_step[i] = 0
    current = set(initial_nonzero)
    visited = set(initial_nonzero)
    step = 0
    cap_activation = {}
# keep going until every capacity lane has been seen nonzero at least once
    while not capacity_lanes.issubset(visited):
        step += 1
        # apply π: content from lane i moves to lane pi_map(i)
        current = {pi_map(i) for i in current}
        for lane in current:
            if activation_step[lane] is None:
                activation_step[lane] = step
        visited |= current
        for lane in capacity_lanes.intersection(current):
            if lane not in cap_activation:
                cap_activation[lane] = step
    # safety: if loop grows too long (shouldn't for 25-lane permutation), break
        if step > 100:
        raise RuntimeError("Exceeded 100 steps - something is wrong.")
    return activation_step, cap_activation, step
def permutation_cycles():
    seen = set()
    cycles = []
    for i in range(25):
        if i in seen:
            continue
        cyc = []
```

```
        cur = i
        while cur not in cyc:
            cyc.append(cur)
            cur = pi_map(cur)
        cycles.append(cyc)
        seen |= set(cyc)
    return cycles
def main():
    # initial condition: lanes 0..15 are nonzero at step 0
    initial_nonzero = set(range(16))
    capacity_lanes = set(range(16, 25))  # lanes 16..24
activation_step, cap_activation, total_steps = simulate_activation(initial_nonzero,
capacity_lanes)
print(f"\nAll capacity lanes (indices 16..24) became nonzero by step {total_steps}.\n")
    print("Activation steps for capacity lanes (index: step):")
    for lane in sorted(capacity_lanes):
        print(f"  lane {lane}: step {activation_step[lane]}")
print("\nFull activation table (lane: first_step_nonzero):")
    for i in range(25):
        print(f"  lane {i:2d}: {activation_step[i]}")
print("\nπ permutation cycles (lane indices):")
    cycles = permutation_cycles()
    for cyc in cycles:
        print(cyc)
if __name__ == "__main__":
    main()
```

IDLE Shell 3.14.0

File Edit Shell Debug Options Window Help

```
>>> ============== RESTART: C:/Users/Maria/OneDrive/Documents/ex29.py ==============

All capacity lanes (indices 16..24) became nonzero by step 2.

Activation steps for capacity lanes (index: step):
  lane 16: step 1
  lane 17: step 1
  lane 18: step 2
  lane 19: step 2
  lane 20: step 1
  lane 21: step 1
  lane 22: step 1
  lane 23: step 1
  lane 24: step 2

Full activation table (lane: first_step_nonzero):
  lane  0: 0
  lane  1: 0
  lane  2: 0
  lane  3: 0
  lane  4: 0
  lane  5: 0
  lane  6: 0
  lane  7: 0
  lane  8: 0
  lane  9: 0
  lane 10: 0
  lane 11: 0
  lane 12: 0
  lane 13: 0
  lane 14: 0
  lane 15: 0
  lane 16: 1
  lane 17: 1
  lane 18: 2
  lane 19: 2
  lane 20: 1
  lane 21: 1
  lane 22: 1
  lane 23: 1
  lane 24: 2

π permutation cycles (lane indices):
[0]
[1, 10, 7, 11, 17, 18, 3, 5, 16, 8, 21, 24, 4, 15, 23, 19, 13, 12, 2, 20, 14, 22, 9, 6]
>>>
```