0.1 code

```python
import numpy as np
from itertools import combinations

# -------------------------
# 1) Low function (EH-style): bottom-most 1
# -------------------------i

def low(col: np.ndarray):
    idx = np.where(col == 1)[0]
    return None if idx.size == 0 else int(idx.max())

def lows(M: np.ndarray):
    return [low(M[:, j]) for j in range(M.shape[1])]

def is_reduced(M: np.ndarray):
    L = [x for x in lows(M) if x is not None]
    return len(L) == len(set(L))

# -------------------------
# 2) Standard EH reduction (left-to-right, eliminate current column using earlier columns)
# -------------------------
def reduce_standard_left_to_right(M: np.ndarray):
    R = (M.copy() % 2).astype(np.uint8)
    m = R.shape[1]

    for j in range(m):
        while True:
            lj = low(R[:, j])
            if lj is None:
                break
            # find j0 < j with same low
            j0 = None
            for k in range(j):
                if low(R[:, k]) == lj:
                    j0 = k
                    break
            if j0 is None:
                break
            R[:, j] ^= R[:, j0]    # col_j <- col_j + col_j0
    return R

# -------------------------
# 3) Variant "one-sweep" (your Q2 pseudocode): push col j into right columns sharing the same low
# -------------------------
def reduce_variant_one_sweep(M: np.ndarray):
    R = (M.copy() % 2).astype(np.uint8)
    m = R.shape[1]

    for j in range(m):
        while True:
            lj = low(R[:, j])
            if lj is None:
                break
            # find a right column j0 > j with same low
            j0 = None
            for k in range(j + 1, m):
                if low(R[:, k]) == lj:
                    j0 = k
                    break
            if j0 is None:
                break
            R[:, j0] ^= R[:, j]   # col_j0 <- col_j0 + col_j
    return R

# -------------------------
# 4) Variant "sweep until stable": repeat the one-sweep pass until no changes
# -------------------------
def reduce_variant_until_stable(M: np.ndarray, max_rounds: int = 200):
    R = (M.copy() % 2).astype(np.uint8)
    m = R.shape[1]

    for _ in range(max_rounds):
        changed = False
        for j in range(m):
            while True:
                lj = low(R[:, j])
                if lj is None:
                    break
                j0 = None
                for k in range(j + 1, m):
                    if low(R[:, k]) == lj:
                        j0 = k
                        break
                if j0 is None:
                    break
                R[:, j0] ^= R[:, j]
                changed = True
        if not changed:
            return R
    raise RuntimeError("max_rounds reached; increase max_rounds if needed.")

# in 4-simplice, take the submatrix as the follow:10 rows are 1-simplices,10 columns are 2-simplices
A = np.array([
    [1,1,1,0,0,0,0,0,0,0],  # 12
    [1,0,0,1,1,0,0,0,0,0],  # 13
    [0,1,0,1,0,1,0,0,0,0],  # 14
    [0,0,1,0,1,1,0,0,0,0],  # 15
    [1,0,0,0,0,0,1,1,0,0],  # 23
    [0,1,0,0,0,0,1,0,1,0],  # 24
    [0,0,1,0,0,0,0,1,1,0],  # 25
    [0,0,0,1,0,0,1,0,0,1],  # 34
    [0,0,0,0,1,0,0,1,0,1],  # 35
    [0,0,0,0,0,1,0,0,1,1], ]) # 45
R_X = reduce_standard_left_to_right(A)
R_Y = reduce_variant_one_sweep(A)
R_Z = reduce_variant_until_stable(A)
print("4-simplice")
print("lows(R_X):", lows(R_X))
print("standard:",R_X)
print("lows(R_Y):", lows(R_Y))
print("one-sweep:",R_Y)
print("lows(R_Z):", lows(R_Z))
print("until stable:",R_Z)
```

0.2 matrices

$$A = \begin{array}{r|cccccccccc} & 123 & 124 & 125 & 134 & 135 & 145 & 234 & 235 & 245 & 345 \\ \hline 12 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 23 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 24 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 34 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 35 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 45 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

$$standardreducedmatrix = \begin{array}{r|cccccccccc} & 123 & 124 & 125 & 134 & 135 & 145 & 234 & 235 & 245 & 345 \\ \hline 12 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 23 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 24 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 34 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 35 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 45 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

$$onesweepmatrix = \begin{array}{r|cccccccccc} & 123 & 124 & 125 & 134 & 135 & 145 & 234 & 235 & 245 & 345 \\ \hline 12 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 14 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 15 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 23 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 24 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 34 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 35 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 45 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

$$untilreducedmatrix = \begin{array}{r|cccccccccc} & 123 & 124 & 125 & 134 & 135 & 145 & 234 & 235 & 245 & 345 \\ \hline 12 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 14 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 15 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 23 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 24 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 34 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 35 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 45 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$