```
import numpy as np
adj = [[2], [1,2], [0,2,3], [3,4], [6], [5,6], [3,4,6]]
W = np.zeros((7,7))
for i, js in enumerate(adj):
    W[i, js] = 1 / len(js)
pi = np.linalg.solve(np.eye(7) - .86*W.T, np.ones(7)*.02)
assert np.allclose(pi.sum(), 1)
assert np.allclose(pi, .86*W.T@pi + .02)
```

Structural or analytic derivation

The resolvent system gives the exact target without taking a long stochastic trajectory. Sparse fixed-point iteration is an alternative; an approximate vector with residual $r = \widehat{\pi} - .86W^{\top}\widehat{\pi} - .02\mathbf{1}$ satisfies $\|\widehat{\pi} - \pi\|_1 \le \|r\|_1/.14$.

Comparative cost and preferred method

For seven nodes, direct elimination is inexpensive and the strongest simple baseline. On a large sparse graph, factorization fill-in can make iterative sparse products preferable. The exact rational calculation and residual bound certify the reproduced digits; no improvement is inferred from using the word resolvent.