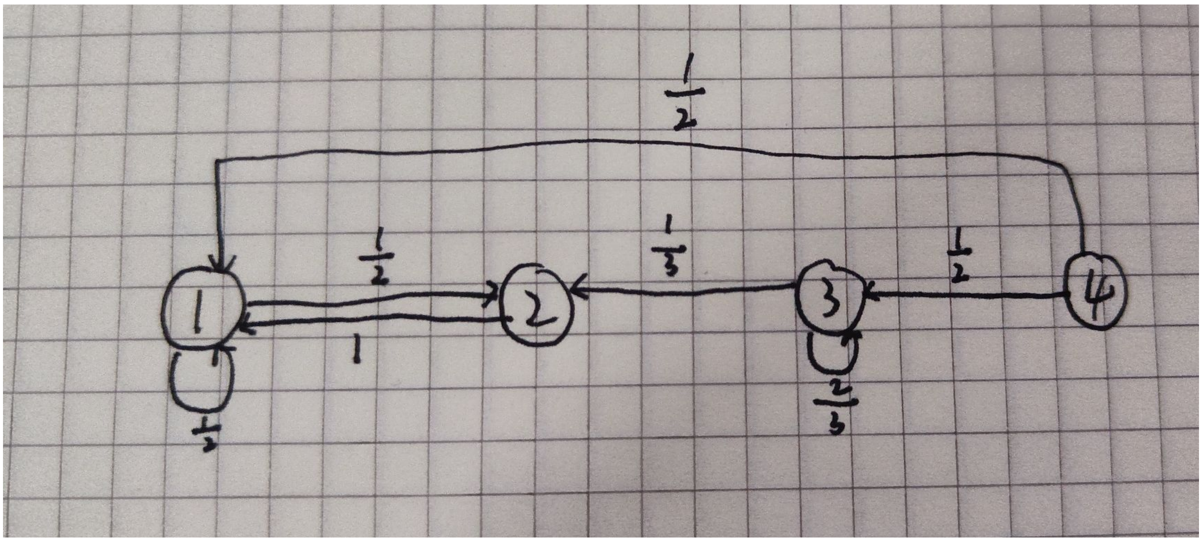
离散马尔科夫链的性质

一步转移矩阵：

$$P=\begin{pmatrix}\frac{1}{2} & 1 & 0 & \frac{1}{2}\\ \frac{1}{2} & 0 & \frac{1}{3} & 0\\ 0 & 0 & \frac{2}{3} & \frac{1}{2}\\ 0 & 0 & 0 & 0\end{pmatrix}$$

状态转移图：



互通性：

很明显，1不能到达3、4

所以没有互通性

周期性：

1和3带有自环，所以一定是非周期的

常返性:

状态1:

1步返回的概率

$$f_1 = \frac{1}{2}$$

2步返回的概率

$$f_2 = \frac{1}{2} \times 1 = \frac{1}{2}$$

m1:

$$m_1 = \frac{1}{2} \times 1 + \frac{1}{2} \times 2 = 1.5$$

所以状态1是正常返的

状态2:

2步返回的概率$f_2 = 1 \times \frac{1}{2} = \frac{1}{2}$

3步返回的概率$f_3 = 1 \times \frac{1}{2} \times \frac{1}{2} = \left(\frac{1}{2}\right)^2$

以此类推，n步返回的概率$f_n = \left(\frac{1}{2}\right)^{n-1}$

所以m_2的表达式是无穷级数$\sum_{i=2}^{\infty} n \left(\frac{1}{2}\right)^{n-1}$

算出来是$\frac{1}{2} m_2 = 2 \times \frac{1}{2} + \left[\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 + \cdots + \left(\frac{1}{2}\right)^{n-1}\right] - n\left(\frac{1}{2}\right)^n$

结果是$m_2 = 3$
所以状态2也是正常返的

状态3、4：

很明显是**非常返**，3、4状态一旦转移到1、2状态，就回不来了

遍历性：

思路1：

从图上看，3、4是可约的，主循环只有1、2

现在单看1、2及他俩之间的边

- 1、2都是正常返的
- 1带自环，因此这个是非周期的

非周期的正常返是遍历的，故这个马尔科夫链存在稳定结果

思路2：根据充分条件："存在正常数n，使得P的n次方没有零元"

拿python跑一下

```
import numpy as np
P = [[0.5, 1, 0, 0.5],
    [0.5, 0, 0.333, 0],
    [0, 0, 0.667, 0.5],
    [0, 0, 0, 0]]

for i in range(1, 1000):
    print('第', i, '次迭代')
    P = P.dot(P)
    print(P)
```

结果：

```
第 1 次迭代
[[0.75     0.5      0.333    0.25     ]
 [0.25     0.5      0.222111 0.4165   ]
 [0.       0.       0.444889 0.3335   ]
 [0.       0.       0.       0.       ]]
第 2 次迭代
[[0.6875     0.625      0.50895354 0.5068055 ]
 [0.3125     0.375      0.29312024 0.34482402]
 [0.         0.         0.19792622 0.14837048]
 [0.         0.         0.         0.        ]]
第 3 次迭代
[[0.66796875 0.6640625  0.63384096 0.63945747]
 [0.33203125 0.3359375  0.32698425 0.33117612]
 [0.         0.         0.03917479 0.02936641]
 [0.         0.         0.         0.        ]]
```

```
第 11 次迭代
[[0.66666667 0.66666667 0.66666667 0.66666667]
 [0.33333333 0.33333333 0.33333333 0.33333333]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
第 12 次迭代
[[0.66666667 0.66666667 0.66666667 0.66666667]
 [0.33333333 0.33333333 0.33333333 0.33333333]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
第 13 次迭代
[[0.66666667 0.66666667 0.66666667 0.66666667]
 [0.33333333 0.33333333 0.33333333 0.33333333]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
```

从1次方算到1000次方，没有哪个是没有非零元的，看来这个例子里，这种方法是无效的?

稳定结果

理论计算:

$$\begin{pmatrix} \frac{1}{2} & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{2}{3} & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

并且有 $a+b+c+d=1$

算出来

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \\ 0 \\ 0 \end{pmatrix}$$

也就是说，结果是稳定的，状态1的概率是$\frac{2}{3}$，状态2的概率是$\frac{1}{3}$

代码验证一下

```
import numpy as np
P = [[0.5, 1, 0, 0.5],
    [0.5, 0, 0.333, 0],
    [0, 0, 0.667, 0.5],
    [0, 0, 0, 0]]
```

```
P = np.array(P)

vector = [0.25, 0.25, 0.25, 0.25]#随便给一个初始的概率向量
vector = np.array(vector)
vector = vector.T

for i in range(0, 100000):#迭代100000次
    print('第', i, '次迭代')
    vector = P.dot(vector)
    print(vector)
```

运行结果

```
第 99993 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99994 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99995 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99996 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99997 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99998 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
第 99999 次迭代
[6.66666667e-001 3.33333333e-001 4.94065646e-324 0.00000000e+000]
```

vector[2]的值是 4.94×10^{-324}

实际已经是0了，可能是python计算精度太高的缘故

与理论计算一致