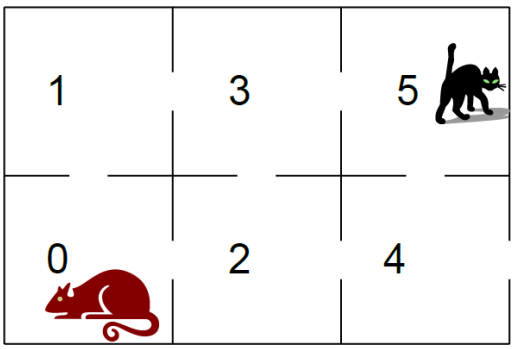
Lista 3

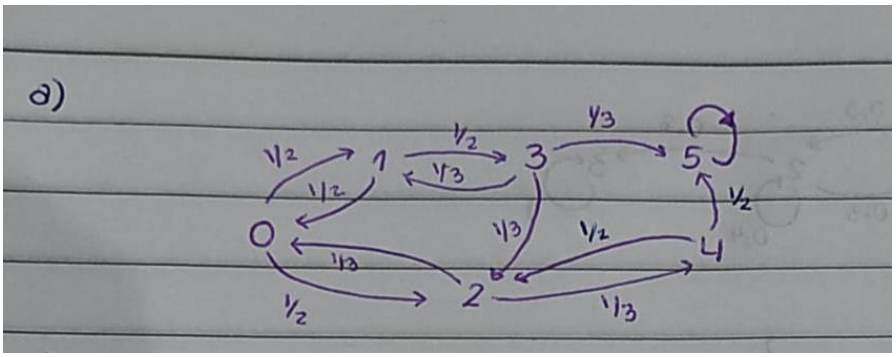
1) Um labirinto é composto de 6 salas numeradas como mostrado na figura abaixo. Um gato é colocado na sala 5 e lá permanece. Um rato é colocado na sala 0 no instante $t = 0$. A cada hora o rato se cansa de permanecer na mesma sala e vai para uma das salas vizinhas com igual probabilidade. A decisão do rato independe do caminho que ele percorreu até então (note queo rato pode voltar para uma sala em que já esteve). Infelizmente (ou felizmente, depende do seu ponto de vista), se o rato vai para a sala 5 ele não sai mais de lá.



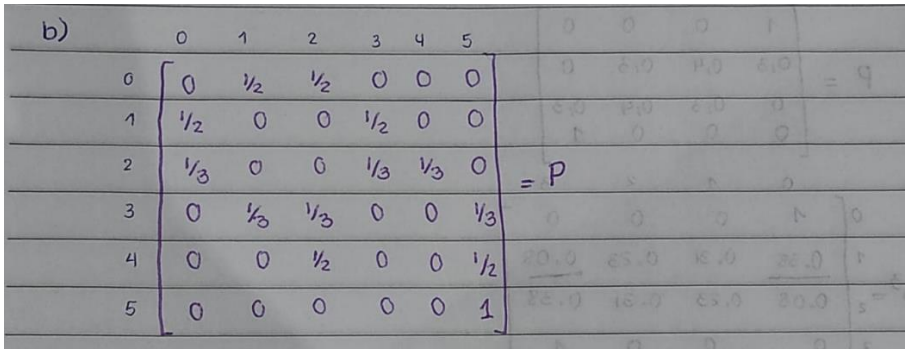
Pede-se:

a) O diagrama de transição de estados.

No diagrama a seguir, supomos que, se o rato sair do labirinto, ele morrerá. É por isso que assumimos um valor de 0,5 para a sala 5, pois ele ainda acabará morrendo.



b) A matriz de transição de 1 passo.



c)A probabilidade do rato morrer após 3 horas.

```
xt = [[0,0.5,0.5,0,0,0], [0.5, 0, 0, 0.5, 0,0], [1/3, 0, 0, 1/3, 1/3, 0], [0, 1/3, 1/3, 0,0, 1/3], [0, 0, 0.5, 0,0, 0.5], [0,0,0,0,0,1]]
xt3 = matrix_power(xt,3)
print(xt3)
```

```
[[0.         0.34722222 0.43055556 0.         0.         0.22222222]
 [0.34722222 0.         0.         0.34722222 0.13888889 0.16666667]
 [0.28703704 0.         0.         0.28703704 0.14814815 0.27777778]
 [0.         0.23148148 0.28703704 0.         0.         0.48148148]
 [0.         0.13888889 0.22222222 0.         0.         0.63888889]
 [0.         0.         0.         0.         0.         1.        ]]
```

Por simulação

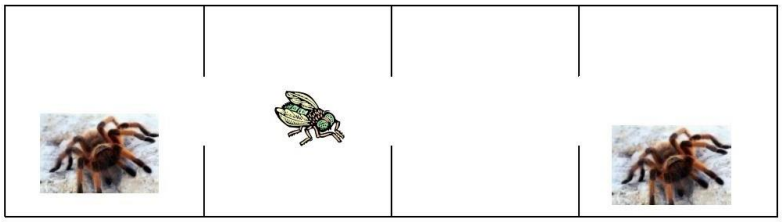
```
n = 100000
chain2 = np.zeros(n, int)
for j in np.arange(0, n, 1):
    probabilidades_transicao =[[0,0.5,0.5,0,0,0], [0.5, 0, 0, 0.5, 0,0], [1/3, 0, 0, 1/3, 1/3, 0], [0, 1/3, 1/3, 0,0, 1/3], [0, 0, 0.5, 0,0, 0.5], [0

    valor_inicial = 0#0 pesq 1 tq 2 tc
    chain_length = 10
    chain = np.zeros(chain_length, int)
    chain[0] = valor_inicial

    for i in np.arange(1, chain_length, 1):
        Linha_atual = probabilidades_transicao[chain[i - 1]]
        acumulada = np.cumsum(Linha_atual)
        r = np.random.uniform(0, 1)
        chain[i] = np.argmax(acumulada > r)

    chain2[j] = chain[3]
```

```
prob1 = np.sum(chain2 == 0) / n
prob2 = np.sum(chain2 == 1) / n
prob3 = np.sum(chain2 == 2) / n
prob4 = np.sum(chain2 == 3) / n
prob5 = np.sum(chain2 == 4) / n
prob6 = np.sum(chain2 == 5) / n

print(prob1)
print(prob2)
print(prob3)
print(prob4)
print(prob5)
print(prob6)
```

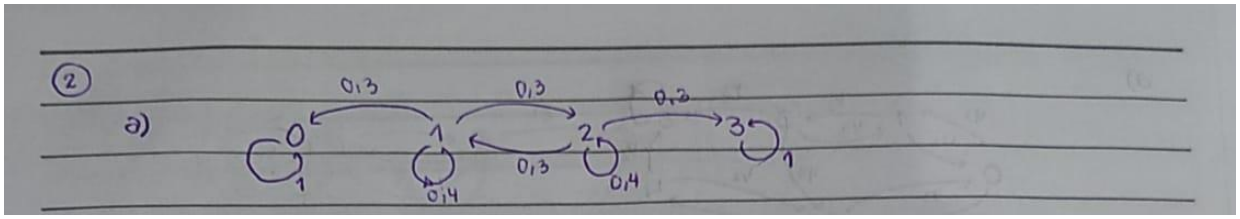
```
0.0
0.34861
0.42916
0.0
0.0
0.22223
```

A probabilidade do rato morrer após 3 horas = 0.222

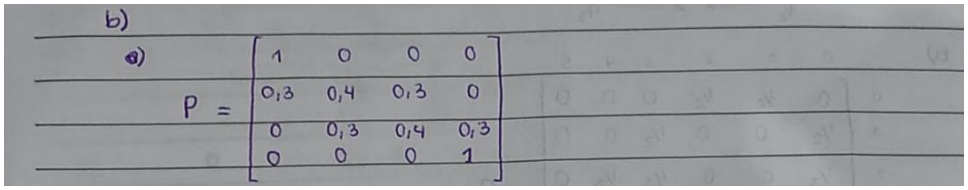
1) Uma caixa possui 4 compartimentos, como mostrado na figura abaixo. No compartimento 0 há uma aranha, assim como no compartimento 3. Uma mosca pousaem um dos compartimentos. A cada minuto (se ela ainda não foi comida) a mosca decide se continua no mesmo compartimento ou se vai para um dos compartimentos vizinhos. A probabilidade de ficar no mesmo compartimento é 0.4 e a probabilidade deir para um compartimento vizinho é 0.6 (0.3 para cada vizinho). Se a mosca vai para onde há uma aranha, ela não sai mais (fica presa na teia).



Pede-se:
a)O diagrama de transição de estados.



b)A matriz de transição.



c)Dado que a mosca pousou no compartimento 1, a probabilidade dela cair em umateia exatamente no terceiro minuto.

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import matrix_power

xt = [[1,0,0,0], [0.3, 0.4, 0.3, 0], [0, 0.3, 0.4, 0.3], [0, 0, 0, 1]]
xt3 = matrix_power(xt,3)
print(xt3)
```

```
[[1.    0.    0.    0.   ]
 [0.495 0.172 0.171 0.162]
 [0.162 0.171 0.172 0.495]
 [0.    0.    0.    1.   ]]
```

P = 0.495 + 0.162 = 0.657

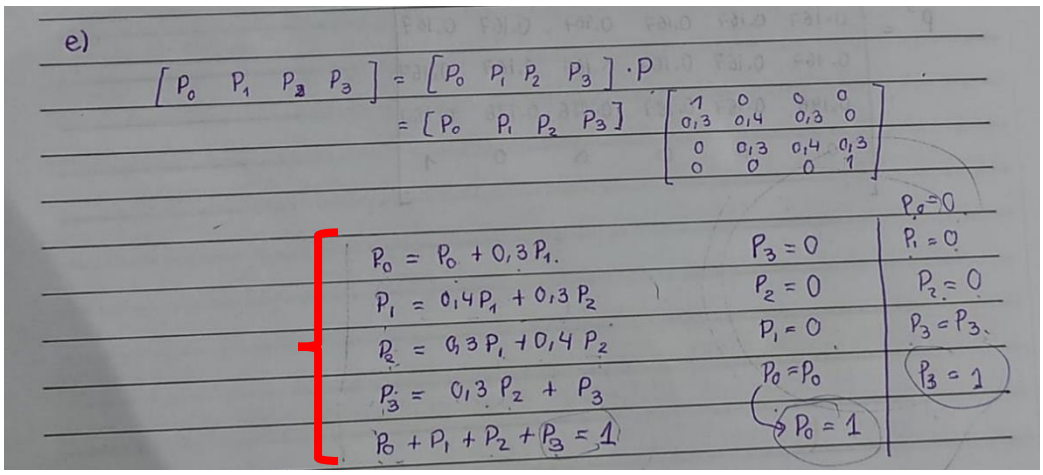
Por simulação

```
n = 100000
chain2 = np.zeros(n, int)
for j in np.arange(0, n, 1):
    probabilidades_transicao =[[1,0,0,0], [0.3, 0.4, 0.3, 0], [0, 0.3, 0.4, 0.3], [0, 0, 0, 1]]

    valor_inicial = 1#posicion mosca
    chain_length = 10
    chain = np.zeros(chain_length, int)
    chain[0] = valor_inicial

    for i in np.arange(1, chain_length, 1):
        Linha_atual = probabilidades_transicao[chain[i - 1]]
        acumulada = np.cumsum(Linha_atual)
        r = np.random.uniform(0, 1)
        chain[i] = np.argmax(acumulada > r)

    chain2[j] = chain[3]
```

```
prob1 = np.sum(chain2 == 0) / n
prob2 = np.sum(chain2 == 1) / n
prob3 = np.sum(chain2 == 2) / n
prob4 = np.sum(chain2 == 3) / n


print(prob1)
print(prob2)
print(prob3)
print(prob4)
```

```
0.49287
0.17151
0.17308
0.16254
```

d)A probabilidade de ser absorvido associada a cada estado.



Temos dois casos para a probabilidade de permanecer em cada estado, de acordo com o sistema de equações encontrado.

Caso 1:

$$Se: P_0 = P_1 = P_2 = 0$$

$$Então: P_3 = P_3$$

$$P_0 + P_1 + P_2 + P_3 = 1$$

$$P_3 = 1$$

A probabilidade de a mosca permanecer no compartimento 3 é de 100%.

Caso 2:

$$Se: P_1 = P_2 = P_3 = 0$$

$$Então: P_0 = P_0$$

$$P_0 + P_1 + P_2 + P_3 = 1$$

$$P_0 = 1$$

A probabilidade de a mosca permanecer no compartimento 0 é de 100%.