Filas MMm

A clase FilaMMm calcula probabilidades e valores esperados para as variáveis aleatórias de uma fila MMm. Deve ser chamada com 3 argumentos:

- taxa de chegada das tarefas
- taxa de serviço de cada servidor
- quantidade de servidores

Um objeto da classe FilaMMm oferece os seguintes atibutos:

- lb: taxa de chegada
- mu: taxa de serviço de cada servidor
- m: quantidade de servidores
- ro: lb / (m*mu)
- p0: probabilidade de nenhuma tarefa no sistema
- epsilon: probabilidade de m ou mais tarefas no sistema
- E_Ns: valor esperado de Ns (quantidade media de tarefas servidas)
- E_Nq: valor esperado de Nq (tamanho medio da fila)
- E_N: valor esperado de N (quantidade media de tarefas no sistema)
- E_S: valor esperado de S (tempo de serviço médio)
- E_W: valor esperado de W (tempo de espera medio na fila)
- E_R: valor esperado de R (tempo de resposta medio)

Um objeto da classe FilaMMm oferece os seguintes métodos:

- pmf_N(x): P[N = x]
- cdf_W(x): P[W < x]
- cdf_R(x): P[R < x]

```
import numpy as np
from math import factorial
from math import exp

class FilaMMm:
    def calculaP0(self):
        soma = 0
        for n in range(1, self.m):
            soma = soma + (((self.m * self.ro) ** n) / factorial(n))
        return 1 / (1 + ((self.m * self.ro) ** self.m) /
(factorial(self.m) * (1 - self.ro)) + soma)

    def __init__(self, lb, mu, m):
        if (lb >= m*mu):
            raise ValueError('Lambda deve ser menor do que m*mu')
        self.lb = float(lb)
```

```
        self.mu = float(mu)
        self.m = m
        self.ro = lb / (m*mu)
        self.p0 = self.calculaP0()
        self.epsilon = ((self.m*self.ro)**self.m)/((1-
self.ro)*factorial(self.m))*self.p0
        # valor esperado de Ns (quantidade media de tarefas servidas)
        self.E_Ns = self.m * self.ro
        # valor esperado de Nq (tamanho medio da fila)
        self.E_Nq = (self.epsilon * self.ro) / (1 - self.ro)
        # valor esperado de N 9quentidade media de tarefas no sistema)
        self.E_N = self.E_Nq + self.E_Ns
        # valor esperado de S (tempo de serviço médio)
        self.E_S = 1 / self.mu
        # valor esperado de W (tempo de espera medio na fila)
        self.E_W = self.epsilon / (self.m * self.mu * (1 - self.ro))
        # valor esperado de R (tempo de resposta medio)
        self.E_R = self.E_S + self.E_W


    def pmf_N(self, x):
        if (x<0 or not(isinstance(False, int))):
            raise ValueError('x deve ser inteiro positivo')
        if (x < self.m):
            return (((self.m * self.ro) ** x) / factorial(x)) *
self.p0
        else:
            return (((self.ro ** x) * (self.m ** self.m)) /
factorial(self.m)) * self.p0

    def pmf_Ns(self, x):
        if (x<0 or not(isinstance(False, int))):
            raise ValueError('x deve ser inteiro positivo')
        if (x < self.m):
            return self.pmf_N(x)
        if (x == self.m):
            return self.epsilon
        else:
            return 0.0

    def pmf_Nq(self, x):
        if (x<0 or not(isinstance(False, int))):
            raise ValueError('x deve ser inteiro positivo')
        if (x == 0):
            pNq = 0
            for i in range(0, self.m +1):
                pNq = pNq + self.pmf_N(i)
            return pNq
        else:
            return self.pmf_N(x+self.m)
```

```
    def cdf_W(self, x):
        if (x<0):
            raise ValueError('x deve ser positivo')
        return 1 - (self.epsilon*exp(-self.m*self.mu*(1-self.ro)*x))

    def cdf_R(self, x):
        if (x<0):
            raise ValueError('x deve ser positivo')
        if (self.ro != (self.m-1)/self.m):
            MULT = (self.epsilon / (1 - self.m + self.m * self.ro)) #
calculado aqui para legibilidade
            p = 1 - exp(-self.mu * x) - MULT * (exp(-self.m * self.mu
* (1 - self.ro) * x) - exp(-self.mu * x))
        else:
            p = 1 - exp(-self.mu * x) - (
                    self.epsilon * self.mu * x * exp(-self.mu * x))
        return p
```

Exercício 1

Um porto recebe um navio a cada 8 horas. Os tempos entre as chegadas são distribuídos exponencialmente. Um navio leva em média 12 horas para ser atendido em um terminal. Suponha que o porto tenha 2 terminais para atender os navios.

- Taxa de chegada: $lb = 1/8 = 0,125$
- Tempo médio de serviço: $E[S] = 12$
- Quantidade de servidores: $m = 2$

a) Quantos navios podem ser atendidos por hora em cada terminal (qual é a taxa de serviço em cada terminal)?

$\mu = \frac{1}{E[S]}$ mu = ?

Calcule o valor de mu e crie a fila com os parâmetros lb, mu e m.

```
mu = 1 / 12

lb = 1/8

m = 2

fila = FilaMMm(lb, mu, m)

print(mu)
print(f"Fila criada com lb: {lb}, mu: {mu}, m: {m}")
```

```
0.08333333333333333
Fila criada com lb: 0.125, mu: 0.08333333333333333, m: 2
```

b) Qual é o número médio de navios na fila do porto?

```
print(fila.E_Nq)
```

```
1.9285714285714284
```

c) Qual a média do tempo que um navio demora para ser atendido no porto, considerando o tempo que está aguardando para atracar (tempo na fila) mais o tempo que está sendo atendido no cais (tempo de serviçoserviço)? (Dica: tempo de resposta)

```
fila.E_R
```

```
27.428571428571427
```

d) Quantos terminais são necessários para que o tempo de resposta médio seja menor do que 15 horas? (Dica: variar o valor de m e testar o tempo médio na fila)

```
for m in range(1, 10):
  try:
    fila = FilaMMm(lb, mu, m)
    if fila.E_R < 15:
      print(m)
      break
  except ValueError:
    pass
```

```
3
```

e) Qual a média do tempo que o navio espera para atracar (tempo na fila) se o porto tiver a quantidade de terminais calculadas no item anterior?

```
fila = FilaMMm(lb, mu, 3)
fila.E_W
```

```
1.894736842105263
```

Exercício 2

Uma central de atendimento 10 pessoas a cada hora. Cada pesoas leva 20 minutos em média para ser atendida. O central tem atualmente 5 pontos de atendimento. Alguns pessoas reclamaram que o tempo de espera é muito grande. Analise o atendimento usando o modelo de filas: Dica - converter todos os dados para minutos:

- Taxa de chegada: 10/60
- Tempo médio de serviço: 20
- Quantidade de servidores: 5

a) Qual é a probabilidade de todos os pontos de atendimento estarem ocupados? (dica: $epsilon = P[N \geq m)$

b) Qual é o número médio de pessoas na central?

c) Qual é o número médio de pessoas na fila?

d) Qual é o número médio de pessoas sendo atendidas?

e) Qual é a média do tempo que uma pessoa fica na central? (Dica: tempo de resposta médio)

f) Qual é a média do tempo que uma pessoa fica na fila? (Dica: tempo médio na fila)

g) O diretor da central quer limitar o seu tempo de espera na fila para menos do que 1,5 minutos. Quantos pontos de atendimento seriam necessários? (Dica: variar o valor de m e testar o tempo médio na fila)

```
lb = 10 / 60
mu = 1 / 20
m = 5

fila = FilaMMm(lb, mu, m)

print(fila.epsilon)

print(fila.E_N)

print(fila.E_Nq)

print(fila.E_Ns)

print(fila.E_R)

print(fila.E_W)

for m_novo in range(5, 20):
  try:
    fila_nova = FilaMMm(lb, mu, m_novo)
    if fila_nova.E_W < 1.5:
      print( m_novo)
      break
  except ValueError:
    pass
```

```
0.3266692800209068
3.9866718933751466
0.6533385600418135
3.333333333333333
23.92003136025088
3.920031360250881
6
```