Trabalho 1

Diana Laura Fernández Duarte

José Gabriel Claure Sánchez

1. Considere um Gerador Linear Congruente (GLC) misto com os seguintes parâmetros: $a = 5$, $c = 3$, $m = 16$ e $x_0 = 7$.

a) Calcule os cinco primeiros números gerados pelo GLC misto.

$$x_{n+1} = (a \times x_n + c) \bmod m$$

$$x_1 = (5 \times 7 + 3) \bmod 16 = 38 \bmod 16 = 6$$

$$x_2 = (5 \times 6 + 3) \bmod 16 = 33 \bmod 16 = 1$$

$$x_3 = (5 \times 1 + 3) \bmod 16 = 8 \bmod 16 = 8$$

$$x_4 = (5 \times 8 + 3) \bmod 16 = 43 \bmod 16 = 11$$

$$x_5 = (5 \times 11 + 3) \bmod 16 = 58 \bmod 16 = 10$$

b) Determine o período desse gerador.

O gerador tem o máximo período possível, ou seja $T = m = 16$ pois:

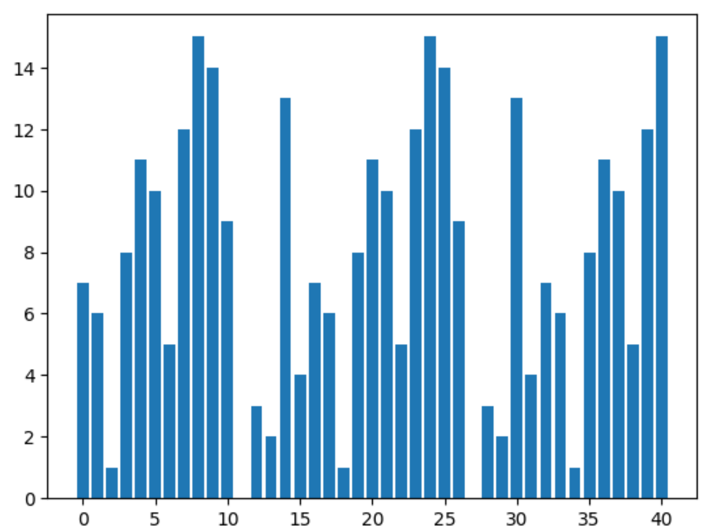
- $c \neq 0$
- $c = 3$ e $m = 16$ são primos entre si, pois não têm divisores em comum, exceto 1.
- O número 2 é o único número primo divisor de m. O número 2 também é um divisor de $a - 1 = 4$.
- Como $m = 16$ é múltiplo de 4, é necessário que $a - 1$ também seja múltiplo de 4. Nesse caso $a - 1 = 4$, portanto, é um múltiplo de 4.

c) Explique se este GLC misto é adequado para aplicações criptográficas. Justifique sua resposta.

A criptografia é uma ferramenta da cibersegurança utilizada para proteger informações confidenciais, que só podem ser decifradas pelo destinatário autorizado. Para esse fim, os geradores lineares congruentes não são adequados, pois a sequência que produzem é facilmente previsível. Além disso, seu período máximo é limitado pelo parâmetro m, o que restringe a geração de grandes quantidades de números aleatórios e pode comprometer a segurança do sistema.

```python
# Exercicio 1: Gerador Congruente Linear Misto
import matplotlib.pyplot as plt
a = 5                          # Constante multiplicadora
c = 3                          # Incremento
m = 16                         # Módulo
xo = 7                         # Semente
amostras = 40                  # Quantidade de números gerados
numeros_gerados = [xo]         # Lista para armazenar as amostras geradas.
cont = 0

 # Executa um loop tantas vezes quanto o número de amostras
 # para calcular os números gerados.
for i in range(amostras):
  if i == 0:
    x = xo
  x = (a*x + c)%(m)            # Fórmula do Gerador Congruente Linear Misto
  numeros_gerados.append(x)

# a) O período desse gerador.
# Conta quantas iterações são necessárias até que o número gerado se repita
# (volte à semente xo). Esse valor corresponde ao período do gerador.

for n in numeros_gerados[1:]:
  cont += 1
  if n == xo:
    break
T = cont                    # Período do gerador
print(numeros_gerados)
plt.bar(range(len(numeros_gerados)),numeros_gerados)
print('Período:',T)
```

Período: 16



2. Em uma central telefônica, o número médio de chamadas recebidas por minuto é igual a 3. Suponha que o número de chamadas recebidas por minuto siga uma distribuição Poisson.

a) Qual é a probabilidade de que exatamente 5 chamadas sejam recebidas em um minuto específico?

$$Pr(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$Pr(X = 5) = \frac{3^5 e^{-3}}{5!}$$

$$Pr(X = 5) = 0.1008$$

b) Qual é a probabilidade de que no máximo 2 chamadas sejam recebidas em um minuto específico?

$$Pr(X \leq 2) = Pr(X = 0) + Pr(X = 1) + Pr(X = 2)$$

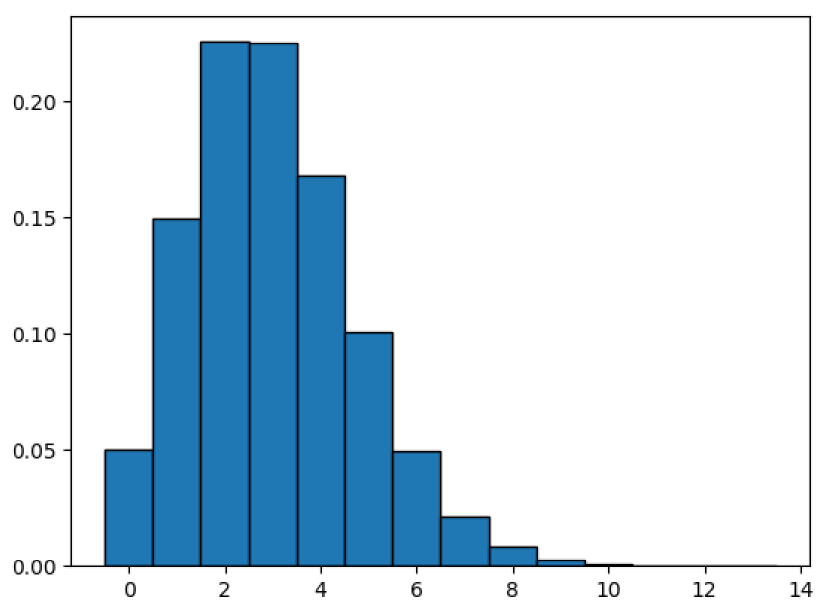
$$Pr(X = 0) = \frac{3^0 e^{-3}}{0!} = 0.0498$$

$$Pr(X = 1) = \frac{3^1 e^{-3}}{1!} = 0.1494$$

$$Pr(X = 2) = \frac{3^2 e^{-3}}{2!} = 0.2240$$

$$P\,r(X \leq 2) = 0.0498 + 0.1494 + 0.2240 = 0.4232$$

```python
import math
import numpy as np
import matplotlib.pyplot as plt

media = 3                       # Numero medio de chegadas num intervalo de tempo
N = 100000                      # Número de amostras a serem geradas
u = np.random.uniform(0, 1, N) # Gera N amostras aleatórias uniformes u ~ U(0,1)
poisson = []

# Converte cada valor uniforme u em uma variável Poisson
# ao encontrar o menor x tal que CDF(x) > u

for element in u:
  i = 0
  pr = math.exp(-media)
  CDF = pr
  while element >= CDF:
    pr = media/(i+1)*pr         # Probabilidade de forma recursiva
    CDF = CDF + pr
    i += 1
  poisson.append(i)
print(poisson)
plt.hist(poisson, bins = np.arange(0,max(poisson)+2)-0.5, edgecolor ='black', density = True)
plt.show()

# a) A probabilidade de que exatamente 5 chamadas sejam recebidas em um minuto específico
x = 5
poisson = np.array(poisson)   # Converte a lista em um array Numpy.
mask = poisson == x           # Máscara booleana: True onde os elementos do array poisson sao iguais ao x.
quantidade = np.sum(mask)     # Soma os valores True da máscara booleana
# Divide o número de amostras igual a x pelo número total de amostras N para calcular Pr(X = x).
Pr = quantidade/N
print('A probabilidade de que exatamente 5 chamadas sejam recebidas em um minuto específico:', Pr)

# b) A probabilidade de que no máximo 2 chamadas sejam recebidas em um minuto específico
y = 2
mask1 = poisson <= y          # Máscara booleana: True onde os elementos do array poisson sao menores ou iguais ao y.
quantidade1 = np.sum(mask1)   # Soma os valores True da máscara booleana
# Divide o número de amostras menores ou iguais ao y pelo número total de amostras N para calcular Pr(Y <= y).
Pr1 = quantidade1/N
print('A probabilidade de que no máximo 2 chamadas sejam recebidas em um minuto específico:', Pr1)
```




```
A probabilidade de que exatamente 5 chamadas sejam recebidas em um minuto específico: 0.10064
A probabilidade de que no máximo 2 chamadas sejam recebidas em um minuto específico: 0.42458
```

3. Uma prova objetiva possui 10 questões, e cada questão apresenta 4 alternativas, das quais apenas uma é correta. Um aluno despreparado responde aleatoriamente todas as questões, assinalando uma alternativa por questão. Considere que X seja a variável aleatória que representa o número de questões acertadas pelo aluno.

a) Qual é a probabilidade de o aluno acertar exatamente 3 questões?

$$Pr(X = 3) = \binom{n}{x} q^x (1-q)^{n-1}$$

$$Pr(X = 3) = \binom{10}{3} (0.25)^3 (1-0.25)^7$$

$$Pr(X = 3) = 0.2503$$

b) Qual é a probabilidade de ele acertar no máximo 2 questões?

$$Pr(X \leq 2) = Pr(X = 0) + Pr(X = 1) + Pr(X = 2)$$

$$Pr(X = 0) = \binom{10}{0} (0.25)^0 (1-0.25)^{10} = 0.5631$$

$$Pr(X = 1) = \binom{10}{1} (0.25)^1 (1-0.25)^9 = 0.1877$$

$$Pr(X = 2) = \binom{10}{2} (0.25)^2 (1-0.25)^8 = 0.2816$$

$$Pr(X \leq 2) = 0.5631 + 0.1877 + 0.2816 = 0.5256$$

c) Determine a média e o desvio padrão da variável aleatória X.

$$\mu = nq = 10 \times 0.25 = 2.5$$

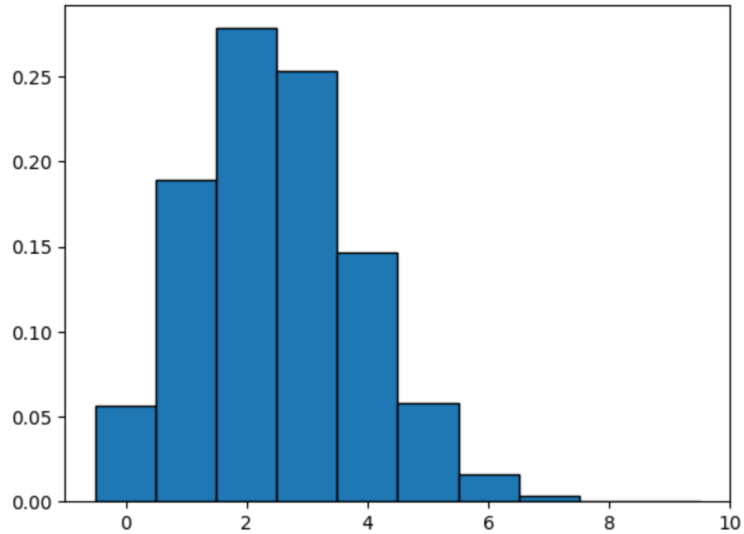
$$\sigma^2 = nq(1-q)$$

$$\sigma^2 = 10 \times 0.25 \times 0.75$$

$$\sigma^2 = 1.875$$

$$\sigma = 1.3693$$

```python
# Exercicio 3: Gerador de Variaveis Aleatoria com Distribuicao Binomial
import numpy as np
import math
import matplotlib.pyplot as plt
q = 0.25                            # Probabilidade de sucesso
n = 10                              # Número de experimentos
N = 100000                          # Número de amostras a serem geradas
u = np.random.uniform(0, 1, N)      # Gera N amostras aleatórias uniformes u ~ U(0,1)
binomial = []

# Converte cada valor uniforme u em uma variável Binomial ao encontrar o menor x
# tal que CDF(x) > u

for element in u:
  i = 0
  pr = (1-q)**(n)
  CDF = pr
  while element >= CDF and i < n:
      pr = (n-i)/(i+1)*(q/(1-q))*pr  # Probabilidade de forma recursiva
      CDF = CDF + pr
      i += 1
  binomial.append(i)

print(binomial)
plt.hist(binomial, bins = np.arange(0,max(binomial)+2)-0.5, edgecolor ='black', density = True)
plt.show()

# a) A probabilidade de o aluno acertar exatamente 3 questões
x = 3
# Converte a lista em um array Numpy.
binomial = np.array(binomial)
# Máscara booleana: True onde os elementos do array binomial sao iguais ao x.
mask = binomial == x
# Soma os valores True da máscara booleana
quantidade = np.sum(mask)
# Divide o número de amostras igual a x pelo número total de amostras N para
# calcular Pr(X = x).
Pr = quantidade/N
print('A  probabilidade de o aluno acertar exatamente 3 questões:', Pr)

# b) A probabilidade de ele acertar no máximo 2 questões
y = 2
# Máscara booleana: True onde os elementos do array binomial sao menores ou iguais ao y.
mask1 = binomial <= y
# Soma os valores True da máscara booleana
quantidade1 = np.sum(mask1)
# Divide o número de amostras menores ou iguais ao y pelo número total de
# amostras N para calcular Pr(Y <= y).
Pr1 = quantidade1/N
print('A probabilidade de ele acertar no máximo 2 questões:', Pr1)

# c) A média e o desvio padrão
# Média da variable aleatoria X
media = np.sum(binomial)/N
print('A Média:', media)
media_analitica = n*q
print('A Média analítica:', media_analitica)
# Variância
variance = np.sum((binomial - media)**2)/(N-1);
variance_analitica = n*q*(1-q)
# Desvio padrão
desvio_padrao = math.sqrt(variance)
desvio_padrao_analitico = math.sqrt(variance_analitica)
print('O desvio padrão da amostra:',desvio_padrao)
print('O desvio padrão analítico:',desvio_padrao_analitico)
```




```
A  probabilidade de o aluno acertar exatamente 3 questões: 0.25308
A probabilidade de ele acertar no máximo 2 questões: 0.52368
A Média: 2.49863
A Média analítica: 2.5
O desvio padrão da amostra: 1.3667760620774856
O desvio padrão analítico: 1.3693063937629153
```

4. Se ocorrerem falhas de energia elétrica de acordo com uma distribuição de Poisson com uma média de 6 falhas a cada duas semanas, calcule a probabilidade de que haverá ao menos 3 falhas durante uma semana específica. Traçar o histograma da variável analisada.

$$Pr(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$$\lambda = \frac{6}{2} = 3\, falhas/semana$$

$$Pr(X \geq 3) \ = \ 1 - \ Pr(X \leq 2)$$

$$Pr(X \leq 2) = Pr(X = 0) + Pr(X = 1) + Pr(X = 2)$$

$$Pr(X = 0) = \frac{3^0 e^{-3}}{0!} = 0.0498$$

$$Pr(X = 1) = \frac{3^1 e^{-3}}{1!} = 0.1494$$

$$Pr(X = 2) = \frac{3^2 e^{-3}}{2!} = 0.2240$$

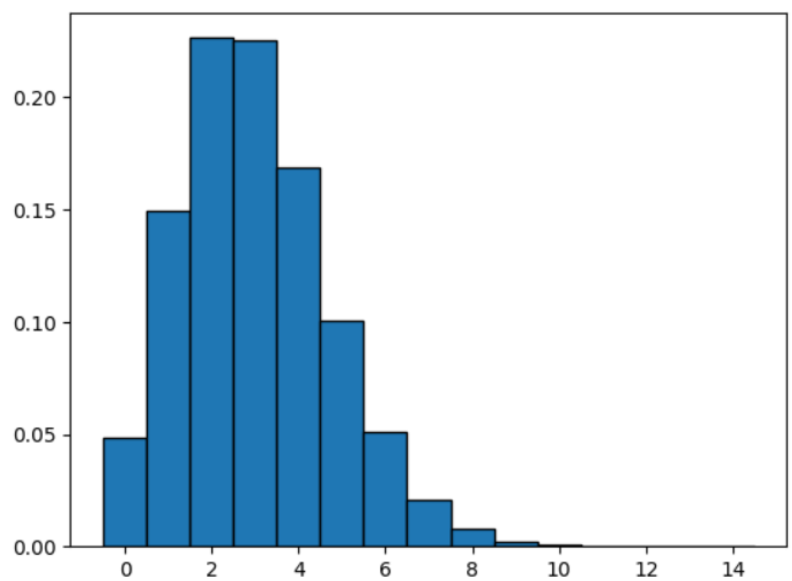
$$P\,r(X \leq 2) = 0.0498 + 0.1494 + 0.2240 = 0.4232$$

$$Pr(X \geq 3) \ = \ 1 - 0.4232$$

$$Pr(X \geq 3) \ = \ 0.5768$$

```python
# Exercicio 4: Gerador de Variaveis Aleatoria com Distribuicao de Poisson
import math
import numpy as np
import matplotlib.pyplot as plt

media = 3                      # Numero medio de chegadas num intervalo de tempo
N = 100000                     # Número de amostras a serem geradas
u = np.random.uniform(0, 1, N) # Gera N amostras aleatórias uniformemente distribuídos u ~ U(0,1)
poisson = []

# Converte cada valor uniforme u em uma variável Poisson ao encontrar
# o menor x tal que CDF(x) > u

for element in u:
  i = 0
  pr = math.exp(-media)
  CDF = pr
  while element >= CDF:
    pr = media/(i+1)*pr               # Probabilidade de forma recursiva
    CDF = CDF + pr
    i += 1
  poisson.append(i)
print(poisson)

# A probabilidade de que haverá ao menos 3 falhas durante uma semana específica
x = 2
# Converte a lista em um array Numpy.
poisson = np.array(poisson)
# Máscara booleana: True onde os elementos do array poisson sao menores ou iguais ao x.
mask = poisson <= x
# Soma os valores True da máscara booleana
quantidade = np.sum(mask)
# Divide o número de amostras menores ou iguais a x pelo número total de amostras N para calcular Pr(X <= x).
p = quantidade/N
 # Pr(X > x) = 1 - P(X <= x-1)
q = 1 - p
print('A probabilidade de que haverá ao menos 3 falhas durante uma semana específica:', q)

# Histograma
plt.hist(poisson, bins=np.arange(0, max(poisson)+2)-0.5, edgecolor='black', density=True)
plt.show()
```

```
A probabilidade de que haverá ao menos 3 falhas durante uma semana específica: 0.57631
```



5. O tempo (em minutos) entre chegadas sucessivas de clientes a um caixa eletrônico pode ser descrito por uma variável aleatória com distribuição exponencial, cuja média é de 2 minutos.

a) Qual é o parâmetro (λ) dessa distribuição exponencial?

$$\beta = \frac{1}{\lambda} = 2$$

$$\lambda = \frac{1}{\beta} = 0.5\ chegadas/minuto$$

b) Qual é a probabilidade de que o tempo de espera até a chegada do próximo cliente seja inferior a 1 minuto?

$$f(x) = \lambda e^{-\lambda x}$$

$$F(x) = Pr(X \leq x) = 1 - e^{-\lambda x}$$

$$Pr(X \leq 1) = 1 - e^{-0.5}$$

$$Pr(X \leq 1) = 0.3935$$

c) Qual é a probabilidade de que o tempo de espera até a chegada do próximo cliente seja superior a 4 minutos?

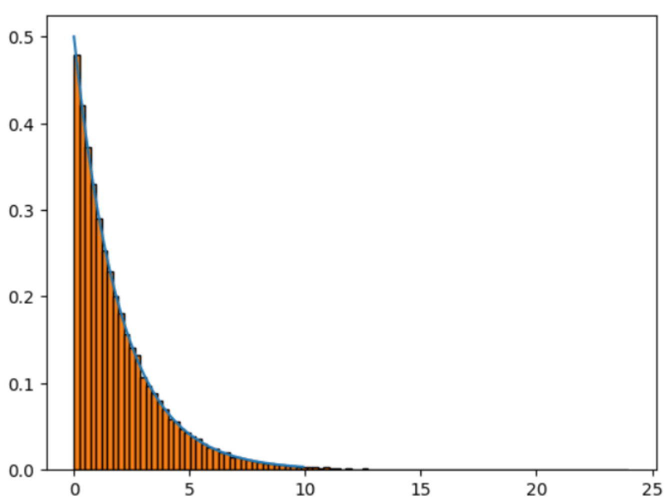
$$Pr(X \geq 4) = 1 - Pr(X \leq 4)$$

$$Pr(X \geq 4) = 1 - (1 - e^{-0.5\times 4})$$

$$Pr(X \geq 4) = e^{-0.5\times 4}$$

$$Pr(X \geq 4) = 0.1353$$

```python
# Exercicio 5: : Gerador de Variaveis Aleatoria com Distribuicao Exponencial
import numpy as np
import matplotlib.pyplot as plt
import math

beta = 2                      # Tempo médio entre eventos (média da distribuição)
Lambda = 1/beta               # Numero medio de chegadas num intervalo de tempo
N = 100000                    # Número de amostras a serem geradas
u = np.random.uniform(0, 1, N)  # Gera N amostras aleatórias uniformemente distribuídas u~U(0,1)
exponencial = np.array([])

# Método da transformada inversa: x = F^(-1)(u)

for element in u:
  x = -math.log(element)/Lambda
  exponencial = np.append(exponencial, x)

# Curva teórica da distribuição exponencial

# Gera uma matriz com elementos igualmente espaçados de 0 até 10 (passo 0.1)
y = np.arange(0, 10, 0.1)
# Função de densidade de probabilidade (pdf) de uma distribuição exponencial
pdf = Lambda*(np.exp(-Lambda*y))

# Gráfico
plt.plot(y, pdf)
plt.hist(exponencial, bins=100, edgecolor='black', density=True)
plt.show()

# b) A probabilidade de que o tempo de espera até a chegada do próximo
#    cliente seja inferior a 1 minuto
valor = 1
# Máscara booleana: True onde os elementos do array exponencial sao menores ou iguais ao valor.
mask = exponencial <= valor
# Soma os valores True da máscara booleana
quantidade = np.sum(mask)
# Divide o número de amostras menores ou iguales a valor pelo número total de
# amostras N para calcular P(X <= valor).
Pr = quantidade/N
print('A probabilidade de que o tempo de espera até a chegada do próximo cliente seja inferior a 1 minuto:', Pr)

# c) A probabilidade de que o tempo de espera até a chegada do próximo cliente seja superior a 4 minutos
valor1 = 4
# Máscara booleana: True onde os elementos do array exponencial sao maiores ou iguais ao valor1.
mask1 = exponencial >= valor1
# Soma os valores True da máscara booleana
quantidade1 = np.sum(mask1)
# Divide o número de amostras maiores ou iguales a valor1 pelo número total de amostras N para calcular P(X >= valor1).
Pr1 = quantidade1/N
print('A probabilidade de que o tempo de espera até a chegada do próximo cliente seja superior a 4 minutos:', Pr1)
```




```
A probabilidade de que o tempo de espera até a chegada do próximo cliente seja inferior a 1 minuto: 0.39689
A probabilidade de que o tempo de espera até a chegada do próximo cliente seja superior a 4 minutos: 0.13427
```

6. A distribuição discreta geométrica conta o número de tentativas até o primeiro sucesso. A pmf é dada por $f(x) = p(1-p)^{x-1}$, onde p representa a probabilidade de sucesso e x o número de tentativas. Fazer um algoritmo para a geração das variáveis aleatórias geométricas. Com o algoritmo proposto calcular:

   Um jogador participa de um jogo no qual ele lança um dado justo (equilibrado, com 6 faces numeradas de 1 a 6). Ele ganha o jogo assim que o número "5" aparecer pela primeira vez. Considere que os lançamentos são independentes.

   Seja X a variável aleatória que representa o número do lançamento no qual o jogador obtém pela primeira vez o número "5".

a) Qual é a probabilidade de que o jogador ganhe o jogo exatamente no terceiro lançamento?

$$Pr(X = 3) = \frac{1}{6}\left(1 - \frac{1}{6}\right)^{3-1}$$

$$Pr(X = 3) = \frac{25}{216} = 0.1157$$

b) Qual é a probabilidade de que ele precise lançar o dado pelo menos 4 vezes para ganhar o jogo?

$$Pr(X \geq 4) = 1 - Pr(X \leq 3)$$

$$Pr(X \leq 3) = Pr(X = 1) + Pr(X = 2) + Pr(X = 3)$$

$$Pr(X = 1) = \frac{1}{6}\left(1 - \frac{1}{6}\right)^{1-1} = \frac{1}{6}$$

$$Pr(X = 2) = \frac{1}{6}\left(1 - \frac{1}{6}\right)^{2-1} = \frac{5}{36}$$

$$Pr(X \leq 3) = \frac{1}{6} + \frac{5}{36} + \frac{25}{216} = 0.4213$$

$$Pr(X \geq 4) = 1 - 0.4213$$

$$Pr(X \geq 4) = 0.5787$$

c) Calcule a média e o desvio padrão de X.

$$\mu = \frac{1}{p} = 6$$

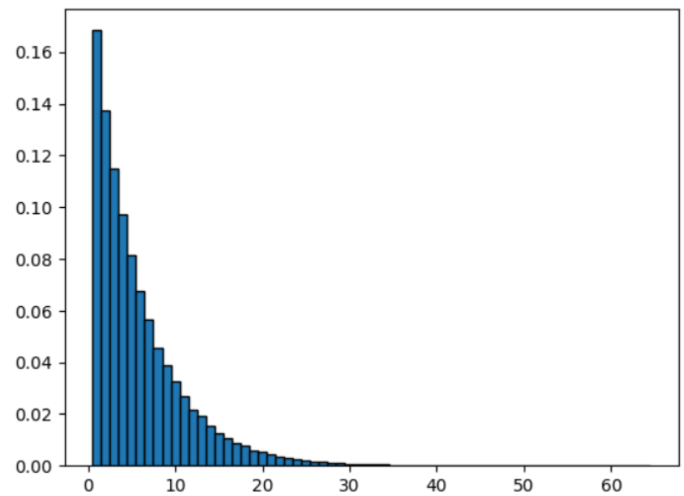
$$\sigma^2 = \frac{1-p}{p^2}$$

$$\sigma^2 = \frac{1 - \frac{1}{6}}{\frac{1}{6}^2}$$

$$\sigma^2 = 30$$

$$\sigma = 5.47$$

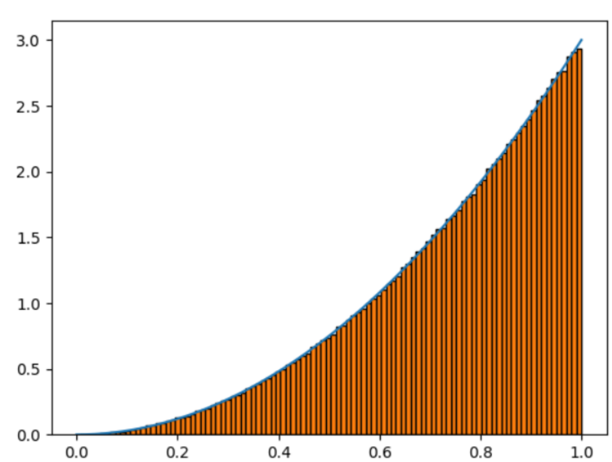
```
# Exercicio 6:  Gerador de Variaveis Aleatoria com Distribuicao Geométrica
import numpy as np
import math
import matplotlib.pyplot as plt

N = 100000                                 # Número de amostras a serem geradas
p = 1/6                                    # Probabilidade de succeso
u = np.random.uniform(0, 1, N)             # Gera N amostras aleatórias uniformes u ~ U(0,1)
dist_geometrica = []

# Converte cada valor uniforme u em uma variável com Distribuicao Geométrica
# ao encontrar o menor x tal que CDF(x) > u

for element in u:
  i = 1
  Pr = p
  CDF = Pr
  while CDF <= element:
    Pr = (1-p)*Pr                          # Probabilidade de forma recursiva
    CDF = CDF + Pr
    i += 1
  dist_geometrica.append(i)

print(dist_geometrica)
plt.hist(dist_geometrica, bins = np.arange(1, max(dist_geometrica) + 2)-0.5, edgecolor ='black', density=True)
plt.show()

# a) A probabilidade de que o jogador ganhe o jogo exatamente no terceiro lançamento
cont = 0
for element in dist_geometrica:
  if element == 3:
    cont += 1
pr = cont/N
print('A probabilidade de que o jogador ganhe o jogo exatamente no terceiro lançamento:',pr)

# b) A probabilidade de que ele precise lançar o dado pelo menos 4 vezes para ganhar o jogo
cont1 = 0
for element in dist_geometrica:
  if element >= 4:
    cont1 += 1
pr1 = cont1/N
print('A probabilidade de que ele precise lançar o dado pelo menos 4 vezes para ganhar o jogo:',pr1)

# c) A média e o desvio padrão de X
media = sum(dist_geometrica)/N
print('A Média:', media)
media_analitica = 1/p
print('A Média analítica:', media_analitica)

numerador = 0
for element in dist_geometrica:
  numerador = numerador + (element - media)**2
variance = numerador/(N-1)
variance_analitica = (1-p)/(p**2)
desvio_padrao = math.sqrt(variance)
desvio_padrao_analitico = math.sqrt(variance_analitica)
print('O desvio padrão da amostra:', desvio_padrao)
print('O desvio padrão analítico:', desvio_padrao_analitico)
```



```
A probabilidade de que o jogador ganhe o jogo exatamente no terceiro lançamento: 0.11467
A probabilidade de que ele precise lançar o dado pelo menos 4 vezes para ganhar o jogo: 0.57975
A Média: 5.98488
A Média analítica: 6.0
O desvio padrão da amostra: 5.451987585025731
O desvio padrão analítico: 5.477225575051661
```

7. Utilizando o método da inversa gerar amostras para a distribuição. Plotar a pdf analítica e o histograma normalizado $f(x) = 3x^2 \quad 0 \leq x \leq 1$

```
# Exercicio 7: Método da inversa para gerar amostras da distribuição f(x) = 3x^2
import numpy as np
import matplotlib.pyplot as plt

N = 1000000                           # Número de amostras a serem geradas
u = np.random.uniform(0, 1, N)        # Gera N amostras aleatórias uniformemente distribuídos entre 0 e 1
x = np.pow(u, 1/3)                    # Inversa

# Curva teórica
y = np.arange(0, 1.1, 0.1)
PDF = 3*(y**2)

plt.plot(y, PDF)
plt.hist(x, bins = 100,edgecolor='black', density=True)
plt.show()
```



8. Considere uma variável aleatória contínua X cuja função densidade de probabilidade (pdf) é dada por:

$$f(x) = 3x^2 \qquad 0 \leq x \leq 1$$

e $f(x) = 0$, caso contrário.

Suponha que você queira gerar valores dessa variável usando o método da aceitação-rejeição.

a) Verifique que f(x) é uma densidade válida.

$$f(x) \geq 0, \forall x$$

$$\int_0^1 3x^2 dx = \left[\frac{3x^3}{3}\right]_0^1 = 1$$

b) Encontre uma constante c adequada para a aplicação do método da aceitação-rejeição, considerando a distribuição candidata escolhida.

$$g(x) = 1$$

$$c = \max_{x \in [0,1]} \frac{f(x)}{g(x)}$$

$$c = \max_{x \in [0,1]} \frac{3x^2}{1}$$

$$c = 3$$

c) Explique o procedimento passo a passo para gerar uma observação de X usando esse método.

1. Gerar um valor aleatório Y a partir de uma distribuição conhecida *g(x).*
2. Gerar um número *u ~ U (0, 1),* independente de Y
3. Se $U \leq \frac{f(Y)}{cg(Y)}$ então aceita-se a amostra *(X = Y)*. Caso contrário, rejeita-se e retorna ao passo 1.

```
# Exercicio 8: Método da aceitação-rejeição para gerar amostras da distribuição f(x) = 3x^2
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate

N = 1000000  # Número de amostras a serem geradas
def f(x):
    return 3 * (x**2)  # PDF objetivo

# a) Verificar se a função de densidade de probabilidade f(x) = 3x^2 é válida:
#    f(x) >= 0 e a área sob a curva == 1

# Integral de f(x)
# Gerar u números aleatórios uniformes u ~ U(0,1)
# Para cada número gerado, calcular f(Ui)
# A integral e igual a média de todos os valores f(Ui)

u = np.random.uniform(0, 1, N)
area = np.sum(f(u))/N

# Para que uma função de densidade seja válida, a função deve ser não negativa
# e a área sob a curva dessa função deve ser 1.

x = np.linspace(0, 1, 100)

if np.all(f(x) >= 0) and np.isclose(area,1,rtol=1e-2):
    mensagem = 'Função de densidade válida'
else:
    mensagem = 'Função de densidade nao válida'
print(mensagem)

# b) Encontrar uma constante c adequada para aceitação-rejeição

def g(x):
    return 1  # PDF proposta g(x) uniforme

c = max(f(x) / g(x))  # max(f(x)/g(x))
print('Constante c adequada:', c)

# c) Geração das amostras
Y = np.random.uniform(0, 1, N)  # Amostras Y ~ g(x)
U = np.random.uniform(0, 1, N)  # U ~ Uniforme(0,1)

amostras_aceitas = []
for i in range(N):
    if U[i] <= f(Y[i]) / (c * g(Y[i])):
        amostras_aceitas.append(Y[i])

# Gráficos
plt.plot(x, f(x))                                                      # PDF analítica
plt.hist(amostras_aceitas, bins = 100, edgecolor = 'black', density=True)   # Histograma normalizado
plt.show()
```

```
Função de densidade válida
Constante c adequada: 3.0
```