Estatística e Probabilidade - Projeto 01

Rodrigo Pita - 118187443

O código completo utilizado para responder todas as questões desse projeto pode ser encontrado no meu GitHub(https://github.com/RodrigoPita/Probest/blob/main/projeto_1.py)

1-a) Abaixo segue o algoritmo (em Python 3) usado para fazer os cálculos da questão 1:

```python
# bibliotecas complementares
from random import randint

# constante para simular infinitas iteracoes
TAM = 5000000

# funcoes complementares
def shiftR( L:list ) -> list:
    '''Recebe uma lista L e faz um shift de seus elementos para a direita
    ex. shiftR( [1, 2, 3] ) -> [3, 1, 2]'''

    if ( len( L ) <= 1 ): return L
    return [L[-1]] + L[:-1]

def shiftL( L:list ) -> list:
    '''Recebe uma lista L e faz um shift de seus elementos para a esquerda
    ex. shiftL( [1, 2, 3] ) -> [2, 3, 1]'''

    if ( len( L ) <= 1 ): return L
    return L[1:] + [L[0]]

# Questao 1
def posicao( inf:bool = False, t:int = TAM, L:list = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ) -> list:
    '''Calcula a posicao de uma particula em relacao aos vertices do poligono p
    depois de passados t segundos, retornando uma nova lista com a posicao atual
    da particula ou uma lista com essa mesma nova lista e o tempo, caso todos
    os vertices tenham sido visitados'''

    # lista auxiliar para registrar quais vertices ja foram percorridos
    auxL = [] + L

    for i in range( t ):
        # variavel para indicar em qual sentido a particula vai se mover
        sinal = randint( 0, 1 )

        # caso o sinal seja 1, a particula anda no sentido horario
        if ( sinal ): L = shiftR( L )
        # caso o sinal seja 0, a a particula anda no sentido anti-horario
        else: L = shiftL( L )

        # posicao atual da particula
        pos = L.index( 1 )

        # se a particula estiver num vertice novo pela primeira vez, o vertice fica registrado na lista auxiliar
        if ( auxL[pos] < 1 ): auxL[pos] = 1
        else: auxL[pos] += 1 # caso a particula esteja revisitando o vertice, incrementamos o numero de visitas

        # se todos os vertices tiverem sido visitados, retorna o tempo em que isso ocorreu
        if ( 0 not in auxL and not inf ): return [ L, i ]
    return [ auxL, t ]
```

1-b) Para responder os itens não teóricos, tanto da questão 1, quanto da questão 2, criei uma função main, o resultado no terminal para o item b) foi:

```
O valor de E[Y] = 64.396
```

1-c) O procedimento do item b) funciona para aproximar o valor esperado de Y por causa da Lei Fraca dos Grandes Números, a qual afirma que para qualquer margem diferente de 0, com uma amostra suficientemente grande haverá probabilidade muito alta que a média das observações se aproxima do valor esperado, dentro da margem.

1-d) Seguindo na função main, abaixo podemos ver os comandos para imprimir os resultados da função massa de probabilidade da variável aleatória Z e os resultados no terminal:

```
A massa de probabilidade da variável Z se dá por:
 -Vertice 0: 8.31 %
 -Vertice 1: 8.3 %
 -Vertice 2: 8.31 %
 -Vertice 3: 8.33 %
 -Vertice 4: 8.33 %
 -Vertice 5: 8.36 %
 -Vertice 6: 8.38 %
 -Vertice 7: 8.35 %
 -Vertice 8: 8.35 %
 -Vertice 9: 8.34 %
 -Vertice 10: 8.33 %
 -Vertice 11: 8.32 %
```

2-a) A situação do inseto no tabuleiro se adequa bem ao modelo de uma cadeia de Markov, pois ela cumpre com as duas principais propriedades da cadeia de Markov. A primeira é que o estado futuro da cadeia depende apenas do estado atual (assim como o inseto só precisa da posição atual dele para saber entre quais posições ele poderá escolher), e a segunda é que a soma das probabilidades de mudança de estados a partir de um dado estado é sempre 1. (Na representação da matriz de probabilidades de transição do inseto ao lado, podemos ver isso com mais clareza).

	[00]	[01]	[02]	[10]	[11]	[12]	[20]	[21]	[22]	Tot
[00]	1	0	0	0	0	0	0	0	0	1
[01]	0.33	0	0.33	0	0.33	0	0	0	0	1
[02]	0	0.5	0	0	0	0.5	0	0	0	1
[10]	0.33	0	0	0	0.33	0	0.33	0	0	1
[11]	0	0.25	0	0.25	0	0.25	0	0.25	0	1
[12]	0	0	0.33	0	0.33	0	0	0	0.33	1
[20]	0	0	0	0.5	0	0	0	0.5	0	1
[21]	0	0	0	0	0.33	0	0.33	0	0.33	1
[22]	0	0	0	0	0	0	0	0	1	1

2-b) Abaixo podemos ver a primeira função auxiliar para o algoritmo de simulação da posição do inseto:

```
52  # Questao 2
53  def caso( s:str ) -> int:
54      '''Calcula com igual probabilidade qual movimento o inseto deve fazer
55      0 -> esquerda
56      1 -> cima
57      2 -> direita
58      3 -> baixo
59      '''
60
61      # caso em que o inseto se encontra na posicao superior central do tabuleiro
62      if ( s == '01' ):
63          aux = [0, 2, 3]
64          return aux[randint( 0, 2 )]
65      # caso em que o inseto se encontra na posicao superior direita do tabuleiro
66      elif ( s == '02' ):
67          aux = [0, 3]
68          return aux[randint( 0, 1 )]
69      # caso em que o inseto se encontra na posicao central esquerda do tabuleiro
70      elif ( s == '10' ):
71          aux = [1, 2, 3]
72          return aux[randint( 0, 2 )]
73      # caso em que o inseto se encontra na posicao central do tabuleiro
74      elif ( s == '11' ):
75          aux = [0, 1, 2, 3]
76          return aux[randint( 0, 3 )]
77      # caso em que o inseto se encontra na posicao central direita do tabuleiro
78      elif ( s == '12' ):
79          aux = [0, 1, 3]
80          return aux[randint( 0, 2 )]
81      # caso em que o inseto se encontra na posicao inferior esquerda do tabuleiro
82      elif ( s == '20' ):
83          aux = [1, 2]
84          return aux[randint( 0, 1 )]
85      # caso em que o inseto se encontra na posicao inferior central do tabuleiro
86      elif ( s == '21' ):
87          aux = [0, 1, 2]
88          return aux[randint( 0, 2 )]
89      # caso de erro
90      return -1
```

Aqui temos uma função auxiliar para visualizar o tabuleiro e a posição do inseto:

```
92  def imprimeTabuleiro( M:list ) -> None:
93      '''Imprime o tabuleiro M em forma de matriz'''
94
95      # imprime a borda superior do tabuleiro
96      print( '\n' + '-'*13 )
97      for i in M:
98          # imprime as linhas do tabuleiro
99          print( f'| {i[0]} | {i[1]} | {i[2]} |')
100     # imprime a borta inferior do tabuleiro
101     print( '-'*13 + '\n' )
```

E abaixo, mais 3 funções auxiliares para resolver os cálculos dos itens c), d) e e):

```
103 def iteraSalto( M:list, trajeto:list = [] ) -> list:
104     '''Entra num loop de iteracoes da funcao salto ate que o inseto caia na armadilha'''
105
106     # inicializando variaveis auxiliares para nao alterar os valores originais
107     auxM, auxTrajeto = [] + M, [] + trajeto
108     while ( True ):
109         auxM, auxTrajeto = salto( auxM, auxTrajeto )
110         # quebra o loop caso o inseto seja capturado
111         if ( 'armadilha' in auxTrajeto[-1] ): break
112     return [ auxM, auxTrajeto ]
```

```
114 def iteraSalto2( M:list, trajeto:list = [], count:int = 0 ) -> list:
115     '''Faz o mesmo que a funcao iteraSalto, mas tambem registra o numero count de vezes
116     que o inseto visita a casa central'''
117
118     # inicializando variaveis auxiliares para nao alterar os valores originais
119     auxM, auxTrajeto, auxCount = [] + M, [] + trajeto, count
120     while ( True ):
121         auxM, auxTrajeto = salto( auxM, auxTrajeto )
122         # incrementa a conta caso haja visita a casa central
123         if ( auxM[1][1] == 1 ): auxCount += 1
124         # quebra o loop caso o inseto seja capturado
125         if ( 'armadilha' in auxTrajeto[-1] ): break
126     return [ auxM, auxTrajeto, auxCount ]
```

```
128 def probabilidadeArmadilhas( casos:list ) -> list:
129     '''A partir de uma lista casos, confere quantas vezes o inseto
130     caiu na armadilha1 e na armadilha2, depois calcula a probabilidade
131     de o inseto cair em cada uma, de acordo com os testes feitos'''
132     # inicializando a lista com a contagem de capturas por armadilha e uma variavel para o total de casos
133     contagemArmadilhas, total = [ 0, 0 ], len( casos )
134     for i in casos:
135         # incrementando o numero de capturas da armadilha1
136         if ( 'armadilha1' in i ): contagemArmadilhas[0] += 1
137         # incrementando o numero de capturas da armadilha2
138         elif ( 'armadilha2' in i ): contagemArmadilhas[1] += 1
139     # retornando uma lista com a probabilidade arredondada da captura de cada armadilha
140     return [ round( contagemArmadilhas[0] * 100 / total, 2 ), round( contagemArmadilhas[1] * 100 / total, 2 ) ]
```

Por fim, temos o algoritmo principal da questão 2:

```python
def salto( M:list, trajeto:list = []) -> list:
    '''Calcula a posicao de um inseto num tabuleiro M após um salto'''

    # dicionario para legenda das direcoes
    legenda = { 0: 'esquerda',
                1: 'cima',
                2: 'direita',
                3: 'baixo',
                4: 'armadilha1',
                5: 'armadilha2' }

    # o inseto ja se encontra na armadilha superior
    if ( M[0][0] == 1 ):
        # fecha a lista do trajeto
        if ( legenda[4] not in trajeto ): trajeto.append( legenda[4] )
        return [ M, trajeto ]
    # o inseto ja se encontra na armadilha inferior
    elif ( M[2][2] == 1 ):
        # fecha a lista do trajeto
        if ( legenda[5] not in trajeto ): trajeto.append( legenda[5] )
        return [ M, trajeto ]

    for i in range( 3 ):
        if ( 1 not in M[i] ): continue # pula a iteracao, caso o inseto nao esteja na linha i do tabuleiro
        for j in range( 3 ):
            # se o inseto estiver na posicao Mij do tabuleiro
            if ( M[i][j] == 1 ):
                # chama a funcao auxiliar caso para escolher a direcao do movimento do inseto
                dir = caso( str( i ) + str( j ) )
                trajeto.append( legenda[dir] ) # atualiza a lista do trajeto
                # esquerda
                if ( dir == 0 ):
                    M[i] = shiftL( M[i] )
                # cima
                elif (dir == 1 ):
                    M = shiftL( M )
                # direita
                elif (dir == 2 ):
                    M[i] = shiftR( M[i] )
                # baixo
                elif ( dir == 3 ):
                    M = shiftR( M )
                # caso de erro
                else:
                    print( f'--ERRO {dir}, caso invalido--')
                    return -1
                return [ M, trajeto ]
```

2-c) Seguem os resultados no terminal:

```
Probabilidades do inseto ser capturado por cada armadilha:
 -Caso 01:
  -Armadilha 1: 66.2 %
  -Armadilha 2: 33.8 %
 -Caso 02:
  -Armadilha 1: 48.96 %
  -Armadilha 2: 51.04 %
 -Caso 10:
  -Armadilha 1: 66.28 %
  -Armadilha 2: 33.72 %
 -Caso 11:
  -Armadilha 1: 50.4 %
  -Armadilha 2: 49.6 %
 -Caso 12:
  -Armadilha 1: 33.05 %
  -Armadilha 2: 66.95 %
 -Caso 20:
  -Armadilha 1: 50.93 %
  -Armadilha 2: 49.07 %
 -Caso 21:
  -Armadilha 1: 33.87 %
  -Armadilha 2: 66.13 %
```

Para os resultados acima, criei 7 listas (para simular os casos em que o inseto não começa em uma armadilha) de iterações do segundo elemento do retorno (lista com o trajeto do inseto até chegar em uma armadilha) da função auxiliar **iteraSalto( )**, depois usei essas listas como parâmetro de entrada para a função auxiliar **probabilidadeArmadilhas( )**.

2-d) Para os resultados ao lado, criei 7 listas (para simular os casos em que o inseto não começa em uma armadilha) de iterações com a quantidade de elementos do segundo elemento do retorno (lista com o trajeto do inseto até chegar em uma armadilha) da função auxiliar **iteraSalto( )**.

```
Médias de saltos são:
 -Caso 01: 6.0176
 -Caso 02: 6.9542
 -Caso 10: 5.9608
 -Caso 11: 6.963
 -Caso 12: 5.92
 -Caso 20: 6.9426
 -Caso 21: 5.9802
```

2-e) Segue o resultado no terminal:

```
O inseto, em média, visita o centro 1.0058 vezes antes de ser capturado.
```

Para calcular o resultado evidenciado acima, criei uma lista de iterações do terceiro elemento do retorno (contador de vezes que o inseto visita a casa central) da função auxiliar **iteraSalto2( )**, dando como parâmetro de entrada o caso em que o inseto começa na posição inferior esquerda, em seguida, fiz a média dos resultados da lista.