Caminhadas aleatórias (Random Walks)

O primeiro projeto de baseia na implementação do cálculo da distribuição estacionária da cadeia de Markov a partir da matriz de transição de estados do jogo “Snakes and Ladders” (imagem1).

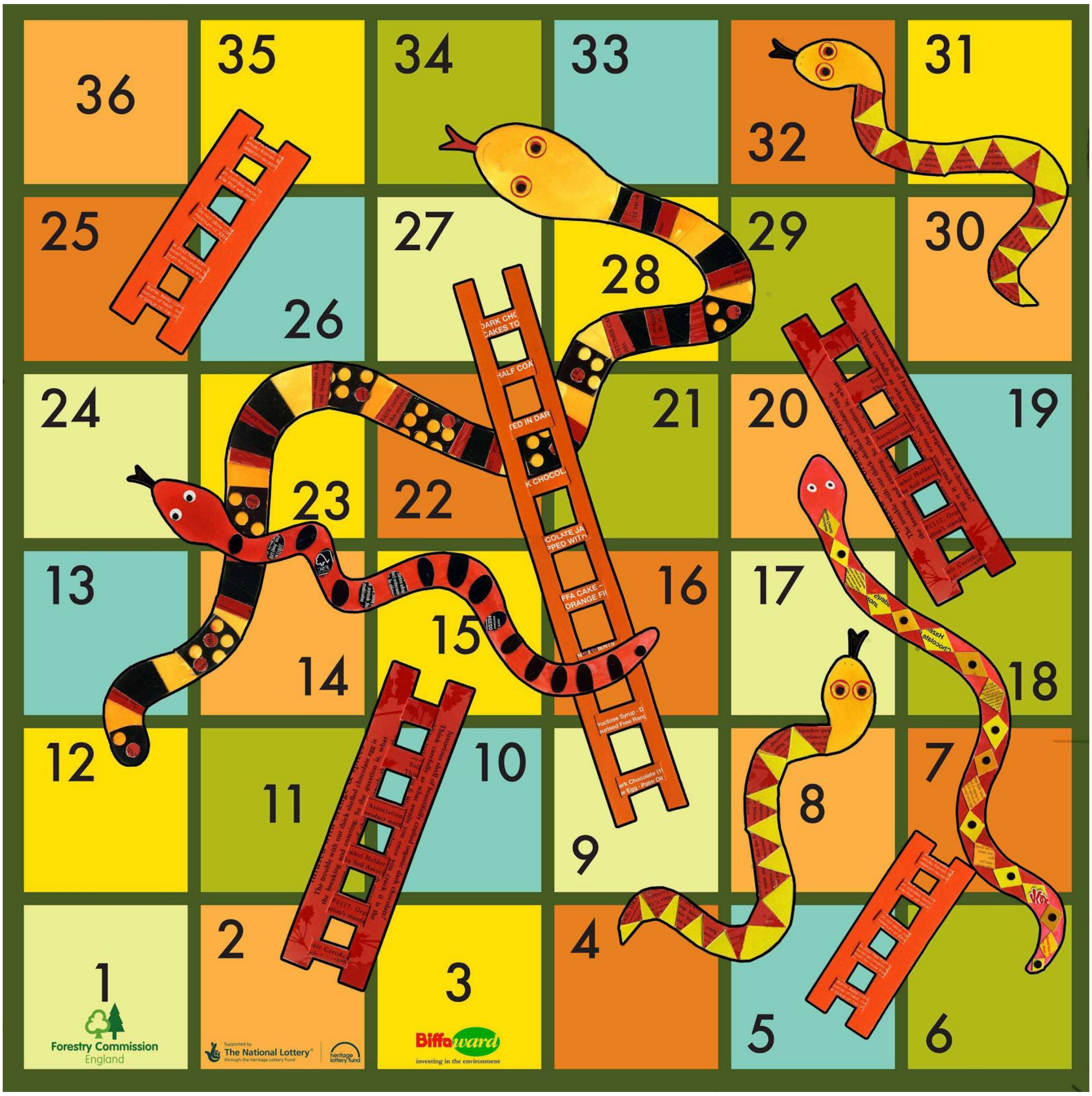

Imagem: Tabuleiro do jogo snakes and ladders.

Primeiramente foi elaborada a matriz de adjacências do tabuleiro representando o grafo do diagrama de estados presentes no jogo, a partir disso foi elaborada a matriz (delta)^-1, matriz essa que receberá em sua diagonal o valor 1/d(vi), sendo d(vi) o grau dos vértices pertencentes ao grafo.

```
matrizADJ = [[0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0], #1
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#2
             [0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#3
             [0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#4
             [0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#5
             [0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#6
             [0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#7
             [0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#8
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0],#9
             [0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#10
             [0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#11
             [0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#12
             [0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#13
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#14
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#15
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#16
             [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#17
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],#18
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#19
             [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#20
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],#21
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0],#22
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0],#23
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#24
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],#25
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0],#26
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],#27
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0],#28
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0],#29
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0],#30
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0],#31
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0],#32
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0],#33
             [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],#34
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1],#35
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]]#36
```

Imagem: Matriz de adjacência.

A matriz P, ou matriz de probabilidade é computada como sendo: P=(delta)^-1*matrizAdj, a matriz de probabilidades define o processo conhecido como Cadeia de Morkov.

```
matrizD = []
max_linhas = 36
max_colunas = 36
soma = 0
k = 0

#Computando matriz delta:

for i in range(max_linhas):
   linha = []
   for j in range(max_colunas):
      linha.append(0)
   matrizD.append(linha)
```

```
for i in range(max_linhas):
    for j in range(max_colunas):
        soma = soma + matrizADJ[i][j]
    matrizD[k][k] = 1/soma
    k = k+1
    soma = 0


#computando matriz P = matrizD*matrizADJ

matrizP = []
mult = 0

for i in range(max_linhas):
    linha = []
    for j in range(max_colunas):
        for k in range(max_colunas):
            mult = mult + matrizD[i][k]*matrizADJ[k][j]
        linha.append(mult)
        mult = 0
    matrizP.append(linha)
```

Para computar o vetor de probabilidades de cada estado num tempo k, nesse exercício k =100, ‘w’ da cadeia de Markov Homogênea, foi utilizado o Power Method, que consiste de multiplicar a matriz de probabilidades por ela mesmo k vezes, e depois obter a partir do resultado um vetor contendo a probabilidade de se estar em cada estado em um tempo k.

```
# tempo k
for k in range(100):
    aux = []
    #multiplicação
    for i in range(36):
        aux.append(0)
        for j in range(36):
            aux[i] += v[j]*matrizP[j][i]
    v = aux
```

Resultado Obtidos

[0.0,
0.0,
 0.0,

0.0019876139002673876,
0.0010245929380067287,
0.0010245929380067287,
0.0015844982582033495,
0.001344957181767505,
0.001510101754845458,
0.0006933106591457362,
0.00035739421106157796,
0.0024530652647714985,
0.0014487609716806375,
0.002011348331737679,
0.0017836480208753363,
0.0019562787784050436,
0.001927891325030704,
0.0010084403567855687,
0.0,
0.0,
0.0,
0.0,
0.0,
0.0,
0.0,
0.0,
0.0015568823432814804,
0.0008025561497204872,
0.0022559449592002463,
0.011278882947334912,
0.006977054299492151,
0.009410734123282851,
0.003596594392709062,
0.0018540048143700673,
0.0018540048143700673,
0.9382968462656476]

A matriz P Barra referente ao Page Rank foi obtida através da expressão:

P=(1−α) P+α 1/n U (Google matrix) , onde
α = 0.1
P = matriz de Probabilidades
U = matriz composta unicamente por 1's

Para facilitar a o cálculo da google Matrix, utilizamos recursos presentes na biblioteca networkX, outro grafo sendo equivalente ao primeiro foi gerado, dessa vez com a criação do grafo utilizando da biblioteca e inserção dos vértices e arestas:

```
vertices = range(1, 37)
arestas = [(1, 2), (1, 3), (2, 15), (3, 4), (3, 5), (4, 5), (4, 6), (5, 7), (6, 7), (6, 8), (7, 8), (7, 9),(8, 9), (8, 10), (9, 27),
        (10, 11), (10, 12), (11, 12),(11, 13), (12, 13), (12, 14), (13, 14), (13, 15), (14, 15),(14, 16), (15, 16), (15, 17), (16, 17),
        (16, 18), (17, 4), (18, 29), (19, 20), (19, 21), (20, 6), (21, 22), (21, 23), (22, 23), (22, 24),(23, 24), (23, 25), (24, 16),
        (25, 35), (26, 27), (26, 28), (27, 28), (27, 29), (28, 29), (28, 30), (29, 30), (29, 31),(30, 31), (30, 32), (31, 32),
        (31, 33), (32, 30), (33, 34), (33, 35), (34, 12), (35, 36), (36, 36)]

grafo = nx.DiGraph(name = 'Dígrafo snakes and ladders')

grafo.add_nodes_from(vertices)
grafo.add_edges_from(arestas)
```

Foi criada a matriz U a calculada o Page rank através das seguintes instruções, em seguida utilizando também do Power Method, foram computados os dois resultados obtidos.

```
#matriz P_
mP_ = matrizP[i][j]*(1-alfa) + u * (alfa * (1/grafo.number_of_nodes()))

#POWER METHOD COM MATRIZ P BARRA
mP_aux = mP_
#P elevado a k
for _ in range(0, k-1):
    mP_aux = np.dot(mP_aux, mP_)

#w(100) = w(0) * P^k
wk_ = np.dot(w, mP_aux)
```

Como observado nos resultados obtidos através do Power Method utilizando-se Pagerank e Cadeia de Markov, obtivemos resultados diferentes sendo o primeiro mais preciso pois com o Pagerank não são obtidos resultados nulos na distribuição estacionária.