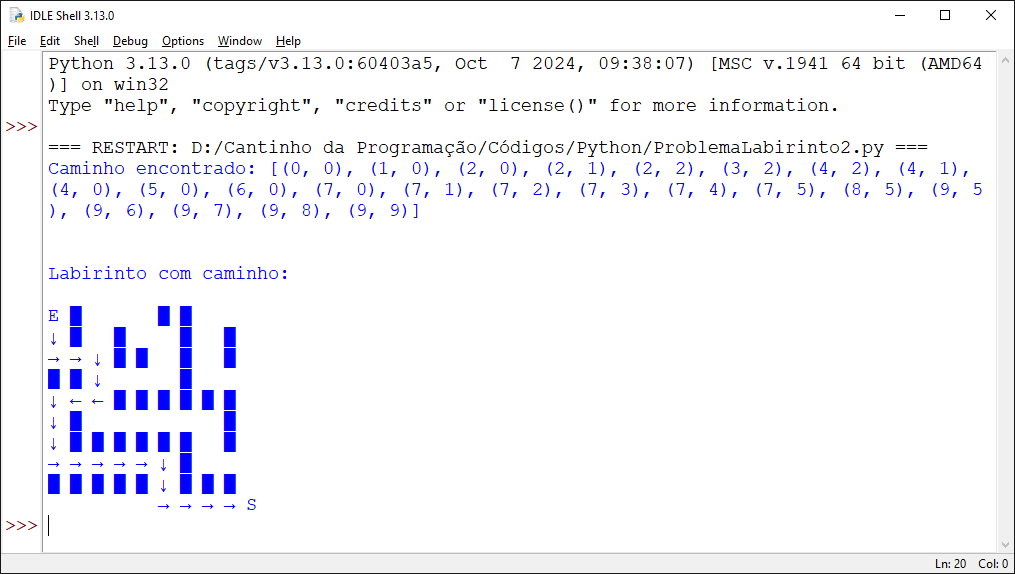
**O Problema do Labirinto**

***Mário Leite***

O “Problema do Labirinto” é um dos muitos tipos de “problemas-desafio” de lógica, muito importante para os programadores iniciantes. Basicamente, o problema se resume a encontrar a saída de um labirinto representado por uma matriz em que cada elemento representa um *vazio* (espaço livre) ou uma *parede* (obstáculo); então, o objetivo é encontrar a saída partir de uma entrada inicial. Assim, como acontece em alguns filmes, o labirinto aparece na forma de um jardim estilizado, com corredores separados por paredes de arbustos. Por exemplo, no clássico filme “*O Iluminado”* (*The Shining*, 1980), em que *Jack* (personagem interpretado pelo ator *Jack Nicholson*) persegue seu filho *Danny*. Desse modo, esse labirinto funciona como um elemento simbólico; servindo de cenário para a perseguição final.

O programa **“ProblemaLabirindo”** é um programa baseado no algoritmo **BFS** (**B**readth-**F**irst **S**earch) - “Busca em Largura” em Português - que mostra um exemplo de código que simula a solução para encontrar a saída de um labirinto, representado por uma matriz 10x10 em que, graficamente, o *output* mostra o elemento de entrada (**E**), os vazios (caminhos livres), os obstáculos (paredes: representados por uma pequena barra vertical azul) e a saída (**S**). A **figura 1** exibe a saída gráfica do programa, indicando os movimentos desde a entrada (**E)** até a saída do labirinto (**S**).

------------------------------------------------------------------------------------------------------------------



**Figura 1 - Saída do programa “ProblemaLabirinto”**

'''

**ProblemaLabirinto.py**

----------------------------------------------------------------------

Simula o "Problema do Labirinto", usando uma matriz 10x10 de modo gráfico. As barras verticais representam paredes (obstáculos) e as setas o sentido do caminho através dos espaços, para achar a saída.

----------------------------------------------------------------------

'''

**from** collections **import** deque

**def** **EncontrarSaida**(LstLabirinto, inicio, saida):

LstDirecoes = [(-1, 0), (1, 0), (0, -1), (0, 1)] *#lista das direções*

linhas, colunas = **len**(LstLabirinto), **len**(LstLabirinto[0])

fila = **deque**([(inicio, [inicio])]) *#cria o deque para a fila de execuções*

visitados = **set**() *#para verificar os locais já pesquisados*

visitados.**add**(inicio)

**while**(fila):

(x, y), caminho = fila.popleft()

**if**(x, y) == saida:

**return** caminho

**for** dx, dy **in** LstDirecoes:

novoX, novoY = x + dx, y + dy

**if**(0 <= novoX < linhas **and** 0 <= novoY < colunas):

**if**(LstLabirinto[novoX][novoY] == 0 **and** (novoX, novoY)

**not** **in** visitados):

fila.**append**(((novoX,novoY),caminho+[(novoX, novoY)]))

visitados.add((novoX, novoY))

**return** **None**

*#-----------------------------------------------------------------------------*

**def** **MostrarLabirinto**(LstLabirinto, caminho):

LstDirecoes = {

(-1, 0): '↑',

(1, 0): '↓',

(0, -1): '←',

(0, 1): '→'

}

*#Cria o mapa (lista-matriz) do labirinto: parede/espaço*

LstMapa = [['█' **if** cel== 1 **else** ' ' **for** cel **in** linha] **for** linha **in**

LstLabirinto]

**for** i **in** **range**(**len**(caminho)-1):

x1, y1 = caminho[i]

x2, y2 = caminho[i+1]

dx, dy = x2 - x1, y2 - y1

LstMapa[x1][y1] = LstDirecoes.**get**((dx, dy), '?')

entradaX, entradaY = caminho[0]

saidaX, saidaY = caminho[-1]

LstMapa[entradaX][entradaY] = 'E'

LstMapa[saidaX][saidaY] = 'S'

**print**()

**print**("\nLabirinto com caminho:\n")

**for** linha **in** LstMapa:

**print**(' '.**join**(linha)) *#cria os vazios do labirinto*

*#=============================================================================*

*#Programa principal*

**if**(\_\_name\_\_ == "\_\_main\_\_"):

LstLabirinto = [

[0, 1, 0, 0, 0, 1, 1, 0, 0, 0],

[0, 1, 0, 1, 0, 0, 1, 0, 1, 0],

[0, 0, 0, 1, 1, 0, 1, 0, 1, 0],

[1, 1, 0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 1, 1, 1, 1, 1, 1, 0],

[0, 1, 0, 0, 0, 0, 0, 0, 1, 0],

[0, 1, 1, 1, 1, 1, 1, 0, 1, 0],

[0, 0, 0, 0, 0, 0, 1,'S', 0, 0, 0],

[1, 1, 1, 1, 1, 0, 1, 1, 1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

]

inicio = (0, 0)

saida = (9, 9)

caminho = **EncontrarSaida**(LstLabirinto, inicio, saida)

**if**(caminho):

**print**("Caminho encontrado:", caminho)

**MostrarLabirinto**(LstLabirinto, caminho)

**else**:

**print**("Não há caminho possível.")

**#Fim do programa "ProblemaLabirinto" ---------------------------------**