BCC | Inteligência Artificial 2022.2
Exercício 5 – Resolução de problemas por meio de busca
Thiago Cavalcanti Silva

UFAPE

O código completo está em anexo.

O algoritmo de busca foi implementado dessa forma:

```python
def busca_largura(mapa, inicio, objetivo):
    fila = deque([(inicio, [])]) # Fila com início e caminho até tal ponto
    visitados = set()

    while fila:
        (x, y), caminho = fila.popleft() # Desempacota na posição e no caminho até ela

        if (x, y) == objetivo:
            return caminho + [(x, y)]

        if (x, y) not in visitados:
            visitados.add((x, y))

            # Explora as bordas
            vizinhos = [(x+1, y), (x-1, y), (x, y+1), (x, y-1)]
            for vx, vy in vizinhos:
                if (0 <= vx < TAMANHO) and (0 <= vy < TAMANHO) and (mapa[vy][vx] in ['_', '$']):
                    fila.append(((vx, vy), caminho + [(x, y)])) # Adiciona o vizinho na fila e o caminho até ele

    return None
```

As outras funções são auxiliares, para criar o mapa e imprimir.

A função de criar o mapa garante a aleatoriedade da posição inicial, final e também dos obstáculos.

```python
def criar_mapa(tamanho, taxa_obstaculos):
    global posicao_inicial, posicao_objetivo

    mapa = [['_'] * tamanho for _ in range(tamanho)]

    num_obstaculos = int(tamanho * tamanho * taxa_obstaculos)
    obstaculos_inseridos = 0

    while obstaculos_inseridos < num_obstaculos:
        x = random.randint(0, tamanho - 1)
        y = random.randint(0, tamanho - 1)

        if mapa[y][x] == '_':
            mapa[y][x] = '|'
            obstaculos_inseridos += 1

    # Insere a posição inicial '@'
    x_inicial = random.randint(0, tamanho - 1)
    y_inicial = random.randint(0, tamanho - 1)
    mapa[y_inicial][x_inicial] = '@'
    posicao_inicial = (x_inicial, y_inicial)


    # Insere a posição do objetivo '$'
    x_objetivo = random.randint(0, tamanho - 1)
    y_objetivo = random.randint(0, tamanho - 1)

    # Certifica-se de que o objetivo não esteja na mesma posição da inicial
    while (x_objetivo, y_objetivo) == (x_inicial, y_inicial):
        x_objetivo = random.randint(0, tamanho - 1)
        y_objetivo = random.randint(0, tamanho - 1)
    mapa[y_objetivo][x_objetivo] = '$'
    posicao_objetivo = (x_objetivo, y_objetivo)

    return mapa
```

```python
def imprimir_mapa(mapa):
    for linha in mapa:
        print(''.join(linha))
    print()
```

Além disso, o tamanho do mapa e também a taxa de obstáculos é facilmente modificável, pois estão em constantes (variáveis globais) para serem passadas como argumentos.

```python
TAMANHO = 10 # Matriz de TAMANHO x TAMANHO
TAXA_OBSTACULOS = 0.1  # Taxa de obstáculos (10%)
```

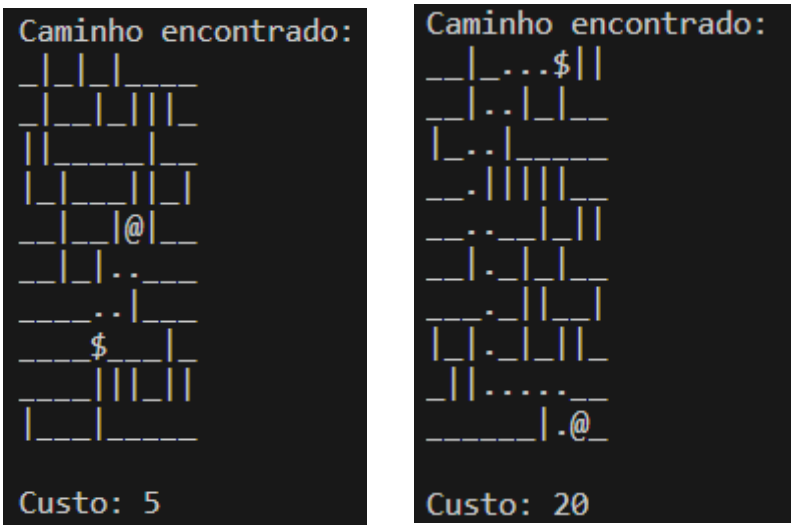
E por fim, a chamada das funções para executar o algoritmo, exibindo o resultado e o custo caso exista um caminho possível.

```python
# Execução do algoritmo
mapa = criar_mapa(TAMANHO, TAXA_OBSTACULOS)
caminho_encontrado = busca_largura(mapa, posicao_inicial, posicao_objetivo)

if caminho_encontrado:
    custo = 1 # Adiciona o passo da última posição para o objetivo
    for x, y in caminho_encontrado:
        if(mapa[y][x] == '_'):
            mapa[y][x] = '.'
            custo += 1

    print("Caminho encontrado:")
    imprimir_mapa(mapa)
    print(f'Custo: {custo}')
else:
    imprimir_mapa(mapa)
    print("Nenhum caminho encontrado.")
```

Em seguida, alguns resultados de execução com 10%, 30% e 40% em uma matriz de tamanho 10x10.

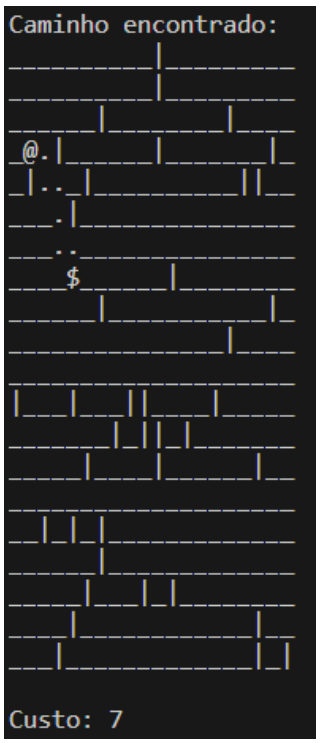
Após, as mesmas taxas de obstáculo em uma matriz 20x20.

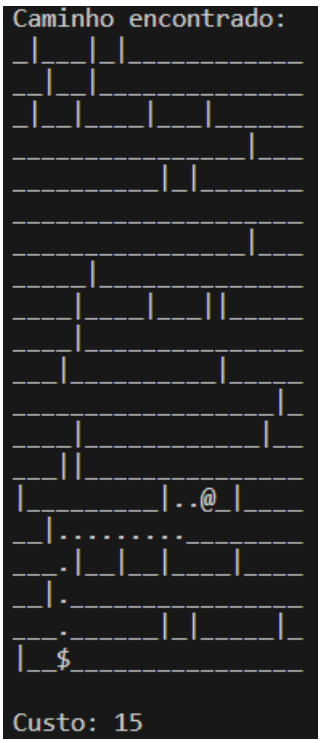
Exemplos com 10% de obstáculos e matriz 10x10



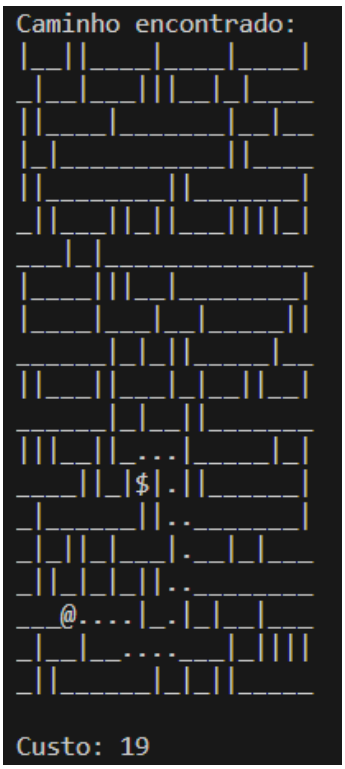
Exemplos com 30% de obstáculos e matriz 10x10

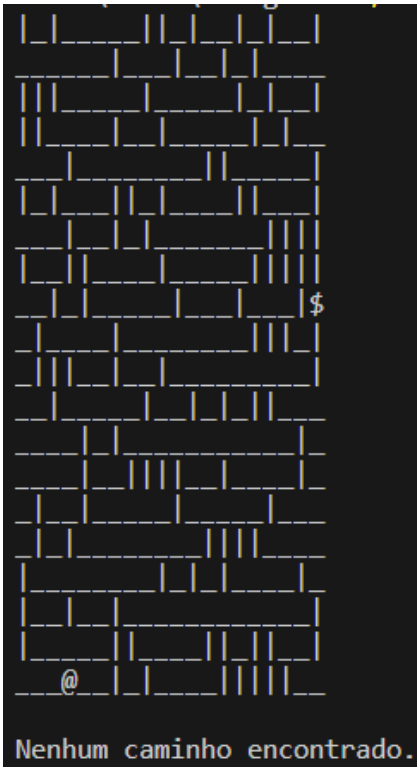
Exemplos com 40% de obstáculos e matriz 10x10



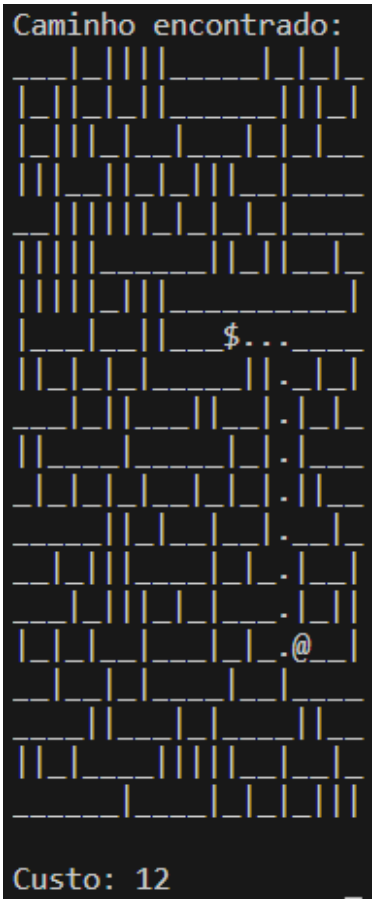


Exemplos com 10% de obstáculos e matriz 20x20





Exemplos com 30% de obstáculos e matriz 20x20



Exemplos com 40% de obstáculos e matriz 20x20