2026-1

Tarea 8
LABERINTO CON BACKTRACKING

Diseño y Análisis de Algoritmos
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

REALIZADO POR:

Enrique Emiliano Cano García

08/11/2025

```python
import sys
from typing import List, Tuple, Optional

class LaberintoSolver:  1 usage  kno4 *
    """
    Resuelve el laberinto cargado desde un archivo .txt con un formato tipo CSV
    usando backtracking y una pila para almacenar la ruta de la solución
    """
    def __init__(self, filename : str):  kno4 *
        self.maze: List[List[str]] = []
        self.rows: int = 0
        self.cols: int = 0
        self.start: Optional[Tuple[int, int]] = None
        self.end: Optional[Tuple[int, int]] = None

        self.path_stack: List[Tuple[int, int]] = []

        try:
            self._cargar_laberinto(filename)
            self.visited = [[False for _ in range(self.cols)] for _ in range(self.rows)]
        except FileNotFoundError:
            print(f"Error en el archivo {filename}, no fue encontrado.", file=sys.stderr)
            sys.exit(1)
        except Exception as e:
            print(f"Error al cargar el laberinto {e}", file=sys.stderr)
            sys.exit(1)
```

```python
    def _cargar_laberinto(self, filename : str):  1 usage  kno4
        """
        Cargar el laberinto desde el archivo .txt con un formato tipo CSV
        """
        with open(filename, 'r') as f:
            self.rows = int(f.readline().strip())
            self.cols = int(f.readline().strip())

            for r in range(self.rows):
                linea = f.readline().strip().split(',')
                if len(linea) != self.cols:
                    raise ValueError(f"Error en la fila {r}: se esperaban {self.cols} columnas, pero se encont

                if 'E' in linea:self.start = (r, linea.index('E'))
                if 'S' in linea: self.end = (r, linea.index('S'))
                self.maze.append(linea)

        if self.start is None:
            raise ValueError("No se encontró un punto de entrada 'E' en el Laberinto")
        if self.end is None:
            raise ValueError("No se encontró un punto de salida 'S' en el laberinto")
```

```python
def resolver(self):  1 usage  kno4
    """
    Punto de entrada público para iniciar la solución del laberinto
    """
    if not self.start:
        print("El laberinto no se cargó correctamente (sin inicio)")
        return

    start_row, start_col = self.start
    if self._backtrack_solve(start_row, start_col):
        print("¡Solución Encontrada!")
        print("La ruta de solución (fila, columna) es:")
        self._imprimir_ruta()
    else:
        print("No se encontró una ruta de solución para este laberinto")
```

```python
def _backtrack_solve(self, r: int, c: int) -> bool:  5 usages  kno4
    """
    Función recursiva con backtracking
    """
    if r < 0 or c < 0 or r >= self.rows or c >= self.cols:
        return False #Se salió del mapa
    if self.maze[r][c] == '1':
        return False #Es una pared
    if self.visited[r][c]:
        return False #Ya se había visitado esa celda

    self.visited[r][c] = True #Marca la celda en la memoria para que sepamos que ya la visitamos
    self.path_stack.append((r, c)) #Se añade a la pila porque resulta ser una solución potencial

    if self.maze[r][c] == 'S': #Se verifica que la celda sea o no sea la Salida con ''S''
        return True
    #ya que no es ''s'', entonces buscamos de nuevo en las direcciones con llamadas recursivas una y otra
    if self._backtrack_solve(r + 1, c): return True
    if self._backtrack_solve(r - 1, c): return True
    if self._backtrack_solve(r, c + 1): return True
    if self._backtrack_solve(r, c - 1): return True
```

```python
def _imprimir_ruta(self):  1 usage  kno4
    """
    Imprime la solución en forma de pila
    """

    maze_con_ruta = [row[:] for row in self.maze]

    for (r, c) in self.path_stack:
        if self.maze[r][c] != 'E' and self.maze[r][c] != 'S':
            maze_con_ruta[r][c] = '*'

    print("\n--- Vista de Laberinto resuelto ---")
    for row in maze_con_ruta:
        print(" ".join(row)) #añade espacios a la salida para que sea visualmente más bonito

    print(f"\n--- Coordenadas de la pila ({len(self.path_stack)} PASOS) ---")
```

Ejecución

```
--- Vista de Laberinto resuelto ---
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 * * * * * * * * * * * * * * * 1 1 1 1
1 1 * 1 1 1 1 1 1 1 1 1 1 1 1 * 1 1 1 1
1 * * 1 1 1 1 1 1 1 1 1 1 1 1 * 1 1 1 1
1 * 1 1 1 1 1 1 1 1 1 1 1 1 1 * 1 1 1 1
1 * * * * * * * * * * 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 1 1 1 1 * 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 1 1 1 1 * 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 1 1 1 1 * 1 1 1 1 * 1 1 1 1
S * * * * * * * * * * 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 * 1 1 1 1
1 1 1 1 1 1 * * * * * * * * * * 1 1 1 1
1 1 1 1 1 1 * 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 * * * * * * * * * * * * * 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 * 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 * * * * 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 E 1 1 1 1
```

```
--- Coordenadas de la pila (80 PASOS) ---
PASO 0: (19, 15)
PASO 1: (18, 15)
PASO 2: (18, 16)
PASO 3: (18, 17)
PASO 4: (18, 18)
PASO 5: (17, 18)
PASO 6: (16, 18)
PASO 7: (16, 17)
PASO 8: (16, 16)
PASO 9: (16, 15)
PASO 10: (16, 14)
```

```
PASO 70: (11, 9)
PASO 71: (11, 8)
PASO 72: (11, 7)
PASO 73: (11, 6)
PASO 74: (11, 5)
PASO 75: (11, 4)
PASO 76: (11, 3)
PASO 77: (11, 2)
PASO 78: (11, 1)
PASO 79: (11, 0)
```