MANUAL TÉCNICO

Manual de Backend

PROGRAMACIÓN III

21-3898 - EDWIN ADONY MONTEJO MARTINEZ

01/06/2024

Contenido

MANUAL TÉCNICO

Introducción

Este manual describe la implementación de un juego de "Totito" (también conocido como "Tic-Tac-Toe") en Python utilizando un árbol de búsqueda y decisión. El programa emplea dos clases principales: `nodoArbol` y `ARBOL`, donde `nodoArbol` representa los nodos del árbol y `ARBOL` maneja la estructura del árbol y sus operaciones.

Clases

Clase nodoArbol

Esta clase representa un nodo en el árbol de decisión.

Atributos:

- `anterior`: Nodo anterior en el árbol.
- `siguiente`: Lista de nodos hijos.
- `nombre`: Nombre del nodo.
- `valor`: Valor del nodo (1 para victoria, 0 para derrota, 0.5 para empate).
- `poscision`: Posición del nodo en el tablero de Totito.
- `profundidad`: Nivel de profundidad del nodo en el árbol.

Métodos:

- `__init__(self, valor, poscision, nombre, profundidad, anterior=None)`: Inicializa el nodo con los valores proporcionados.

Clase ARBOL

Esta clase maneja el árbol de búsqueda y decisión.

Atributos:

- `raiz`: Nodo raíz del árbol.
- `ubicacion`: Nodo actual en el árbol.
- `busqueda`: Nodo de búsqueda en el árbol.

Métodos:

- `__init__(self)`: Inicializa el árbol con un nodo raíz.
- `ubicaciones(self)`: Muestra la ubicación actual en el árbol y sus nodos hijos.
- `asignarValores(self, ganador=None)`: Asigna valores a los nodos en función de si es una victoria, derrota o empate.
- `sumValores(self)`: Suma los valores de los nodos hijos del nodo actual.
- `insert(self, valor, poscision)`: Inserta un nuevo nodo en el árbol.
- `generarNombre(self, poscision)`: Genera un nombre único para un nodo en función de su posición.
- `volver(self)`: Regresa al nodo anterior.
- `avanzar(self, poscision)`: Avanza al nodo hijo especificado por la posición.
- `eliminar(self, poscision)`: Elimina un nodo hijo especificado por la posición.
- `existe(self, poscision)`: Verifica si un nodo hijo especificado por la posición existe.
- `buscaOpc(self)`: Busca la mejor opción de movimiento.
- `generar_arbol_grafico(self)`: Genera un gráfico del árbol utilizando `graphviz`.
- `_generar_arbol_grafico(self, nodo, dot)`: Método recursivo para generar el gráfico del árbol.
- `generarJson(self, nodo)`: Genera una representación JSON del árbol.
- `generarArbolDeJson(self)`: Reconstruye el árbol desde un archivo JSON.
- `_generarArbolDeJson(self, array, nodoAnt=False)`: Método recursivo para reconstruir el árbol desde JSON.
- `_asignarValores(self, nodo)`: Asigna valores a los nodos de forma recursiva.
- `evaluarArbol(self)`: Evalúa el valor del árbol.
- `_evaluarArbol(self, nodo)`: Método recursivo para evaluar el árbol.
- `_evaluarHojas(self, nodo)`: Método recursivo para contar las hojas del árbol.

Las Funciones Clave

`insert(self, valor, poscision)`

Inserta un nuevo nodo en el árbol en la posición especificada.

`asignarValores(self, ganador=None)`

Asigna valores a los nodos en función del resultado del juego (victoria, derrota o empate).

`generar_arbol_grafico(self)`

Genera un gráfico visual del árbol usando la biblioteca `graphviz`.

`generarJson(self, nodo)`

Genera una representación JSON del árbol para su almacenamiento.

`generarArbolDeJson(self)`

Reconstruye el árbol desde un archivo JSON.