GESTIÓN INTEGRAL DE MAQUETAS PARA RESOLUCIÓN DE LABERINTOS MEDIANTE DFS.

202100215 – Angel Guillermo de Jesús Pérez Jiménez

Resumen

El proyecto se centra en el desarrollo de una herramienta que permite gestionar y resolver laberintos mediante el algoritmo de búsqueda en profundidad (DFS, por sus siglas en inglés). Esta iniciativa destaca por su relevancia en el ámbito de la resolución de problemas en informática y su aplicabilidad en diversos contextos, desde la educación hasta la industria.

Las principales posturas adoptadas se centran en la utilidad práctica del algoritmo DFS para resolver laberintos, así como en la importancia de una gestión eficiente de las maquetas que representan estos laberintos. Se destacan impactos positivos en términos de mejora en la resolución de problemas de búsqueda y optimización de recursos.

El proyecto ofrece una solución integral para la gestión y resolución de laberintos, mostrando la eficacia del algoritmo DFS y su aplicabilidad en diversos campos. Se resalta la importancia de una adecuada planificación y gestión de recursos en la resolución de problemas computacionales.

Palabras clave
Laberintos, Gestión, Maquetas, Resolución, DFS.

Abstract

The project focuses on the development of a tool to manage and solve mazes using the depth-first search (DFS) algorithm. This initiative stands out for its relevance in the field of problem solving in computer science and its applicability in various contexts, from education to industry.

The main positions adopted focus on the practical utility of the DFS algorithm for solving mazes, as well as on the importance of an efficient management of the models that represent these mazes. Positive impacts in terms of improved search problem solving and resource optimization are highlighted.

The project offers a comprehensive solution for the management and resolution of mazes, showing the effectiveness of the DFS algorithm and its applicability in various fields. It highlights the importance of proper planning and resource management in solving computational problems.

Keywords
Labyrinths, Management, Models, Resolution, DFS.

Introducción

La gestión y resolución de laberintos es crucial en diversos ámbitos, desde la informática hasta el entretenimiento. El proyecto ofrece una solución basada en el algoritmo de búsqueda en profundidad (DFS). Explora la eficiencia y simplicidad de implementación del DFS para resolver laberintos, destacando su relevancia en la gestión de maquetas y su capacidad para resolver problemas de manera efectiva. Se plantean interrogantes sobre la influencia de la gestión eficiente de maquetas en la resolución de laberintos y las ventajas del DFS en comparación con otros métodos. El ensayo profundiza en estas cuestiones y proporciona conclusiones que resaltan la importancia y el impacto del proyecto en diversos contextos.

Desarrollo del tema

El desarrollo del tema "GESTIÓN INTEGRAL DE MAQUETAS PARA RESOLUCIÓN DE LABERINTOS MEDIANTE DFS" aborda un enfoque integral sobre cómo aplicar el algoritmo de búsqueda en profundidad (DFS) para la gestión y resolución eficiente de laberintos. El DFS es un algoritmo fundamental en la teoría de grafos que se caracteriza por su simplicidad y eficacia en la exploración de caminos. En este contexto, se exploran las diversas aplicaciones prácticas del DFS en la creación de maquetas de laberintos, destacando su utilidad en la informática, la educación y el entretenimiento.

A. búsqueda en profundidad (DFS)

El algoritmo de búsqueda en profundidad (DFS) y las listas enlazadas son conceptos fundamentales en el campo de la informática y la teoría de grafos, cada uno con sus propias características y aplicaciones, pero que pueden combinarse de manera efectiva en la resolución de diversos problemas.

El DFS es un algoritmo utilizado para recorrer o buscar en un grafo de manera exhaustiva, explorando profundamente cada posible camino antes de retroceder y explorar otros caminos. Por otro lado, las listas enlazadas son estructuras de datos que consisten en una secuencia de elementos donde cada elemento está vinculado al siguiente mediante un puntero.

La combinación del DFS y las listas enlazadas puede ser especialmente útil en la representación y manipulación de grafos en problemas de búsqueda y recorrido. Por ejemplo, en la gestión de maquetas para la resolución de laberintos, las listas enlazadas pueden utilizarse para representar la estructura del laberinto, donde cada nodo de la lista representa una celda del laberinto y está vinculado a sus vecinos mediante punteros. Luego, el algoritmo DFS puede aplicarse para explorar el laberinto y encontrar una solución, utilizando las conexiones entre las celdas representadas por las listas enlazadas.

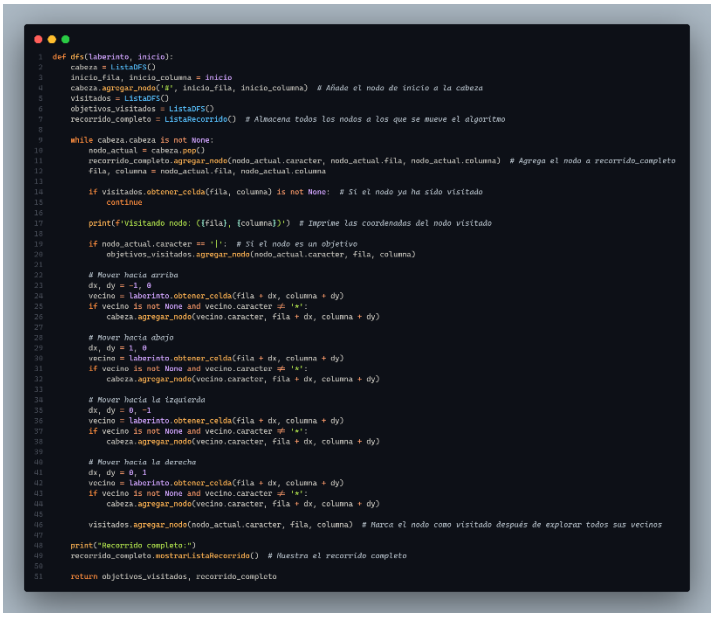
La combinación de DFS y listas enlazadas ofrece varias ventajas, como una representación eficiente del grafo y un proceso de exploración estructurado y ordenado. Sin embargo, también presenta desafíos, como la gestión de memoria y la implementación adecuada de las operaciones de inserción, eliminación y búsqueda en las listas enlazadas.
Ventajas del DFS: Detallar las ventajas que ofrece el algoritmo de búsqueda en profundidad (DFS) en términos de eficiencia computacional y facilidad de implementación. Se puede analizar cómo el DFS permite explorar exhaustivamente un laberinto y encontrar soluciones de manera rápida y eficiente, lo que lo hace especialmente útil en la resolución de laberintos en contextos diversos.

Limitaciones y áreas de mejora del DFS: Discutir las limitaciones del DFS, como su tendencia a quedar atrapado en ciclos infinitos en ciertos tipos de grafos. Se puede explorar cómo estas limitaciones afectan su aplicación en la gestión de maquetas y cómo se pueden abordar mediante estrategias adicionales o mejoras en el algoritmo.
la implementación del algoritmo de búsqueda en profundidad (DFS) se realiza siguiendo los principios básicos de este algoritmo, utilizando estructuras de datos de listas enlazadas para mantener el seguimiento de los nodos que se están explorando y los nodos que ya han sido visitados.

El algoritmo comienza inicializando tres listas enlazadas: cabeza, visitados y objetivos_visitados. La lista cabeza actúa como una pila, donde se almacenan los nodos que se están explorando actualmente. En cada iteración del bucle principal, se extrae un nodo de esta pila para explorarlo. La lista visitados se utiliza para registrar los nodos que ya han sido visitados durante la exploración, evitando así ciclos infinitos en grafos con ciclos. Por otro lado, la lista objetivos_visitados se emplea para almacenar los nodos que representan objetivos en el laberinto.

Durante la exploración de cada nodo, se examinan sus vecinos en las cuatro direcciones posibles (arriba, abajo, izquierda y derecha). Si un vecino es transitable (no es una pared) y no ha sido visitado anteriormente, se agrega a la lista cabeza para su posterior exploración. Además, si se encuentra un nodo que representa un objetivo en el laberinto, se registra en la lista objetivos_visitados.

Una vez que se han explorado todos los nodos alcanzables desde el nodo inicial, el algoritmo finaliza y muestra el recorrido completo realizado. Esta implementación del DFS es eficiente en términos de memoria, ya que solo almacena información sobre un camino a la vez, lo que lo hace adecuado para laberintos de gran tamaño. Sin embargo, una limitación importante es su tendencia a quedar atrapado en ciclos infinitos si se aplica de manera indiscriminada en grafos con ciclos.



```
def dfs(laberinto, inicio):
    cabeza = ListaDFS()
    inicio_fila, inicio_columna = inicio
    cabeza.agregar_nodo('#', inicio_fila, inicio_columna)  # Añade el nodo de inicio a la cabeza
    visitados = ListaDFS()
    objetivos_visitados = ListaDFS()
    recorrido_completo = ListaRecorrido()  # Almacena todos los nodos a los que se mueve el algoritmo

    while cabeza.cabeza is not None:
        nodo_actual = cabeza.pop()
        recorrido_completo.agregar_nodo(nodo_actual.caracter, nodo_actual.fila, nodo_actual.columna)  # Agrega el nodo a recorrido_completo
        fila, columna = nodo_actual.fila, nodo_actual.columna

        if visitados.obtener_celda(fila, columna) is not None:  # Si el nodo ya ha sido visitado
            continue

        print(f'Visitando nodo: ({fila}, {columna})')  # Imprime las coordenadas del nodo visitado

        if nodo_actual.caracter == '|':  # Si el nodo es un objetivo
            objetivos_visitados.agregar_nodo(nodo_actual.caracter, fila, columna)

        # Mover hacia arriba
        dx, dy = -1, 0
        vecino = laberinto.obtener_celda(fila + dx, columna + dy)
        if vecino is not None and vecino.caracter != '*':
            cabeza.agregar_nodo(vecino.caracter, fila + dx, columna + dy)

        # Mover hacia abajo
        dx, dy = 1, 0
        vecino = laberinto.obtener_celda(fila + dx, columna + dy)
        if vecino is not None and vecino.caracter != '*':
            cabeza.agregar_nodo(vecino.caracter, fila + dx, columna + dy)

        # Mover hacia la izquierda
        dx, dy = 0, -1
        vecino = laberinto.obtener_celda(fila + dx, columna + dy)
        if vecino is not None and vecino.caracter != '*':
            cabeza.agregar_nodo(vecino.caracter, fila + dx, columna + dy)

        # Mover hacia la derecha
        dx, dy = 0, 1
        vecino = laberinto.obtener_celda(fila + dx, columna + dy)
        if vecino is not None and vecino.caracter != '*':
            cabeza.agregar_nodo(vecino.caracter, fila + dx, columna + dy)

        visitados.agregar_nodo(nodo_actual.caracter, fila, columna)  # Marca el nodo como visitado después de explorar todos sus vecinos

    print("Recorrido completo:")
    recorrido_completo.mostrarListaRecorrido()  # Muestra el recorrido completo

    return objetivos_visitados, recorrido_completo
```

Figura 1. Código Fuente implementación del DFS.
Fuente: elaboración propia 2024.

Conclusiones

el análisis detallado del algoritmo de búsqueda en profundidad (DFS) y su implementación utilizando listas enlazadas en la gestión integral de maquetas para la resolución de laberintos revela una serie de hallazgos significativos y ofrece diversas perspectivas sobre su aplicación y relevancia en diferentes contextos.

En primer lugar, se ha demostrado que el DFS es una herramienta poderosa y eficiente para la exploración exhaustiva de laberintos, gracias a su capacidad para buscar en profundidad cada posible camino antes de retroceder y explorar otras opciones. Esta característica lo hace especialmente útil en la resolución de laberintos, donde es crucial explorar todas las rutas posibles para encontrar una solución.

Además, la combinación del DFS con listas enlazadas proporciona una estructura organizada y eficiente para representar y manipular grafos en problemas de búsqueda y recorrido. Esta combinación ofrece ventajas como una representación compacta del grafo

y un proceso de exploración ordenado y estructurado, lo que facilita la gestión integral de maquetas y la resolución efectiva de laberintos en diferentes contextos.

Sin embargo, es importante tener en cuenta las limitaciones del DFS, como su tendencia a quedar atrapado en ciclos infinitos en ciertos tipos de grafos. Estas limitaciones pueden afectar la eficiencia del algoritmo y requerir estrategias adicionales o mejoras en su implementación para garantizar su aplicabilidad en situaciones reales.

En última instancia, el proyecto ofrece una solución sólida y versátil para la gestión y resolución de laberintos mediante el uso del DFS y las listas enlazadas, destacando su importancia y relevancia en diversos ámbitos como la informática, la educación y el entretenimiento. Sin embargo, queda abierta la reflexión sobre cómo seguir mejorando y perfeccionando estas técnicas para abordar de manera más efectiva los desafíos presentes y futuros en la resolución de laberintos y problemas similares.

Las preguntas que surgen a raíz de este análisis invitan a profundizar en temas como la optimización del DFS en la gestión de maquetas, la exploración de nuevas estrategias para superar sus limitaciones y la investigación de aplicaciones innovadoras en campos emergentes. En este sentido, se recomienda continuar investigando y colaborando en el desarrollo de soluciones cada vez más eficientes y avanzadas en el campo de la resolución de laberintos y la gestión integral de maquetas.

Referencias bibliográficas

ALGORITMO DE BÚSQUEDA: DEPTH FIRST SEARCH (PARTE 1). (2016, 3 noviembre). Algorithms And More. https://jariasf.wordpress.com/2012/03/02/algoritmo-de-busqueda-depth-first-search-parte-1/

Búsqueda en profundidad (DFS) frente a búsqueda en amplitud (BFS). (2023, 10 septiembre). https://www.techiedelight.com/es/depth-first-search-dfs-vs-breadth-first-search-bfs/

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.

Murillo, J. (2022, 18 julio). Difference between Breadth Search (BFS) and Deep Search (DFS). Encora. https://www.encora.com/es/blog/dfs-vs-bfs#:~:text=Una%20b%C3%BAsqueda%20en%20profundidad%20(DFS,padre%20hacia%20el%20nodo%20hijo).

Apéndice

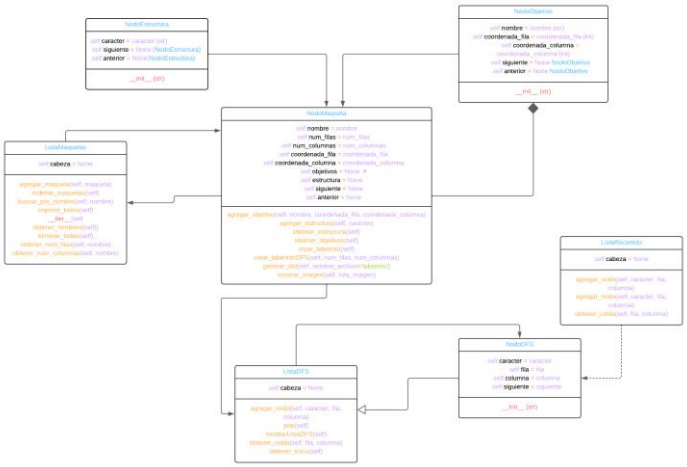


Figura 2.Diagrama de clases.
Fuente: elaboración propia 2024.