búsqueda en anchura – breadth-first search

Authored by

memjavad

November 10, 2025

Búsqueda en Amplitud (Breadth-First Search - BFS)

Primary Disciplinary Field(s): Ciencias de la Computación, Teoría de Grafos, Inteligencia Artificial

1. Definición Central

La Búsqueda en Amplitud (BFS) es un algoritmo fundamental en la teoría de grafos y la informática, diseñado para recorrer o buscar sistemáticamente en estructuras de datos de tipo árbol o grafo. Su característica distintiva reside en la forma en que explora la estructura: comienza en un nodo raíz y explora todos los nodos vecinos a ese nivel de profundidad antes de pasar a los nodos del siguiente nivel de profundidad. Este método de expansión nivel por nivel asegura una exploración exhaustiva y metódica de toda la estructura, garantizando que se descubran primero los caminos más cortos desde el nodo inicial hasta cualquier otro nodo alcanzable. La BFS se clasifica como un algoritmo de búsqueda ciega o no informada, lo que significa que no utiliza heurísticas o información adicional sobre el objetivo durante el proceso de búsqueda.

El mecanismo operativo del BFS se basa intrínsecamente en el uso de una estructura de datos de tipo **Cola (Queue)**, la cual opera bajo el principio FIFO (First-In, First-Out). Cuando el algoritmo visita un nodo, este nodo se retira de la cola, y todos sus vecinos no visitados son inmediatamente añadidos al final de la cola. Este estricto ordenamiento de la cola es lo que impone la exploración por niveles concéntricos. Los nodos que están a una distancia de *k* aristas del nodo inicial serán siempre procesados antes que cualquier nodo a una distancia de *k+1*, lo cual es crucial para su propiedad de optimalidad.

La principal fortaleza conceptual de la Búsqueda en Amplitud radica en su capacidad para determinar el camino más corto entre el nodo de inicio y cualquier otro nodo en un **grafo no ponderado**. Al garantizar que todos los nodos a una distancia menor se visiten primero, el primer momento en que se encuentra el nodo objetivo, se garantiza que el camino recorrido hasta ese punto es el que contiene el menor número de aristas. Esta propiedad de optimalidad, combinada con la completitud (la seguridad de que encontrará una solución si esta existe), establece al BFS como una herramienta indispensable en el análisis de conectividad y la resolución de problemas de camino mínimo.

2. Etimología y Desarrollo Histórico

Aunque la formalización y el reconocimiento general de la Búsqueda en Amplitud se produjeron a mediados del siglo XX, las ideas subyacentes a los recorridos sistemáticos de estructuras complejas tienen raíces más profundas en la resolución de acertijos y la lógica matemática. El concepto de recorrer una estructura de manera organizada para garantizar la cobertura total se

hizo especialmente relevante con el surgimiento de la teoría de grafos como disciplina matemática. Sin embargo, su formalización algorítmica estuvo estrechamente ligada al desarrollo de las primeras máquinas programables.

Un precursor significativo, aunque inicialmente no publicado, fue el trabajo realizado por el ingeniero alemán **Konrad Zuse**. En el contexto del desarrollo de su lenguaje de programación, el Plankalkül, Zuse describió métodos en la década de 1940 para recorrer estructuras de grafos en el contexto de la planificación de redes, que contenían elementos esenciales de lo que hoy conocemos como BFS. No obstante, la publicación que selló la identidad del BFS y lo introdujo en el canon de la informática fue la realizada por el matemático estadounidense **Edward F. Moore** en 1959. Moore desarrolló el algoritmo explícitamente para encontrar el camino más corto fuera de un laberinto, un problema que se mapea directamente a la búsqueda del camino mínimo en un grafo.

Tras su formalización, la Búsqueda en Amplitud fue rápidamente adoptada por el campo emergente de la Inteligencia Artificial (IA) en las décadas de 1960 y 1970. Se convirtió en uno de los principales algoritmos de "búsqueda ciega" utilizados para explorar espacios de estados, donde los estados de un problema (como las posiciones en un juego de mesa) se representan como nodos de un grafo. Su garantía de encontrar la solución óptima (el camino con menos movimientos o estados) la convirtió en una herramienta estándar para la resolución de problemas donde la profundidad de la solución era la métrica principal.

3. Características Clave y Principios Operacionales

La eficacia y el comportamiento predecible del BFS se derivan de un conjunto estricto de principios operacionales que dictan cómo se expande la búsqueda. El principio más fundamental es la exploración nivel por nivel. A diferencia de otros algoritmos que pueden adentrarse profundamente en una rama antes de explorar el resto, el BFS garantiza que la periferia de la búsqueda se expanda uniformemente en todas las direcciones desde el nodo inicial, como si se propagara una onda.

Para gestionar esta expansión ordenada, la estructura de datos de la **Cola** es indispensable. La cola actúa como un listado de espera de los nodos que han sido descubiertos pero que aún no han sido explorados. Cuando un nodo (N) se visita, se procesa su información, y todos sus vecinos (V1, V2, V3...) que no hayan sido visitados previamente se añaden al final de la cola. Dado que la cola es FIFO, V1, V2 y V3 serán procesados solo después de que todos los nodos que fueron agregados a la cola antes que ellos (es decir, los nodos en el nivel actual) hayan sido completamente explorados.

Dos propiedades teóricas hacen que el BFS sea particularmente valioso en la informática:

Completitud (Completeness): Si existe al menos un camino que conecta el nodo inicial con el

nodo objetivo, el BFS está garantizado para encontrarlo. Esto lo diferencia de algunos algoritmos de búsqueda heurística que podrían atascarse en caminos subóptimos o infinitos.
Optimalidad (Optimality): En el contexto de grafos no ponderados o donde todas las aristas tienen el mismo "costo", el BFS garantiza que el primer camino que encuentra hacia el nodo objetivo es el camino con la menor cantidad de aristas (el camino más corto).

4. Detalles de Implementación: Uso de Colas

La implementación algorítmica del BFS requiere la gestión precisa de dos conjuntos de datos: la Cola (para ordenar los nodos a visitar) y un conjunto o lista de **Nodos Visitados** (para evitar ciclos y reprocesamiento). El proceso comienza inicializando la cola con el nodo raíz y marcando dicho nodo como visitado. A partir de ese punto, el algoritmo entra en un bucle continuo que se ejecuta mientras la cola no esté vacía.

Dentro del bucle, el algoritmo realiza tres pasos esenciales en cada iteración. Primero, extrae (dequeue) el nodo frontal de la cola, convirtiéndolo en el "nodo actual". Segundo, si el nodo actual es el nodo objetivo, la búsqueda finaliza con éxito. Tercero, si no es el objetivo, el algoritmo examina todos los vecinos adyacentes al nodo actual. Para cada vecino, se verifica si ya ha sido visitado. Si no lo ha sido, el vecino se marca inmediatamente como visitado y se añade al final de la cola. Este proceso asegura que el recorrido mantenga el orden de amplitud requerido.

La función del conjunto de nodos visitados es crucial, especialmente en la exploración de grafos (donde los ciclos son comunes) en contraposición a árboles (donde los ciclos son inexistentes). Si no se llevara un registro de los nodos visitados, el algoritmo podría entrar en un bucle infinito al oscilar entre dos nodos que se apuntan mutuamente, o desperdiciar recursos visitando y procesando repetidamente subgrafos ya explorados. La marcación inmediata de un nodo como visitado tan pronto como es descubierto (antes de ser extraído de la cola) previene eficazmente estos problemas y asegura la terminación del algoritmo.

5. Análisis de Complejidad

El rendimiento de la Búsqueda en Amplitud se analiza típicamente en términos de complejidad temporal y complejidad espacial, utilizando la notación de la Gran O. Estos análisis dependen de la estructura del grafo, específicamente del número de vértices (V) y el número de aristas (E).

En cuanto a la **Complejidad Temporal** (Tiempo), el BFS es altamente eficiente. Dado que cada nodo (vértice) se extrae de la cola y se procesa exactamente una vez, y cada arista se examina una o dos veces (dependiendo de si el grafo es dirigido o no dirigido) durante la búsqueda de vecinos, la complejidad temporal es lineal en relación con el tamaño del grafo. Formalmente, la complejidad de tiempo es O(V + E). Este rendimiento lineal es óptimo para los algoritmos de recorrido de grafos, ya que requiere, en el peor de los casos, examinar cada parte de la

estructura.

El principal desafío del BFS reside en su **Complejidad Espacial** (Memoria). El algoritmo debe almacenar en memoria la cola de todos los nodos que han sido descubiertos pero que aún no han sido expandidos. En el peor de los casos, la cola puede contener casi todos los vértices del grafo. Por lo tanto, la complejidad espacial es O(V). En el contexto de la búsqueda en árboles de estados, donde *b* es el factor de ramificación (número promedio de hijos) y *d* es la profundidad de la solución, la complejidad espacial es O(b^d). Este requisito de memoria puede ser la principal limitación práctica del BFS cuando se aplica a problemas con un factor de ramificación muy grande o una gran cantidad de estados a la misma profundidad, ya que podría agotar rápidamente la memoria disponible.

6. Aplicaciones y Ejemplos Prácticos

Gracias a su optimalidad en la búsqueda del camino más corto, el BFS se emplea en una amplia gama de aplicaciones en informática y matemáticas. Su uso no se limita únicamente a la teoría de grafos, sino que se extiende a la optimización de redes, la inteligencia artificial y el análisis de datos.

La aplicación más canónica es la **determinación del camino más corto** en estructuras no ponderadas. Esto es fundamental en sistemas de navegación y cartografía digital, donde el objetivo es encontrar la ruta con el menor número de paradas o segmentos (ignorando la distancia física, que requeriría un algoritmo ponderado como Dijkstra). De manera similar, en el análisis de redes sociales, el BFS se utiliza para calcular los "Grados de Separación" entre dos usuarios, encontrando la cadena de conexiones más corta.

En el ámbito de las redes y la infraestructura tecnológica, el BFS es vital. Se utiliza en algoritmos de rastreo de la web (web crawling) para explorar sistemáticamente las páginas de un sitio, asegurando que se acceda a todos los enlaces de una página antes de profundizar en los enlaces de las páginas subsiguientes. También es crucial en los sistemas de gestión de memoria, como los algoritmos de **recolección de basura** (garbage collection), donde se utiliza para identificar todos los objetos de memoria que son accesibles desde un conjunto de raíces y, por lo tanto, no deben ser eliminados.

Otras aplicaciones notables incluyen la identificación de los componentes conectados de un grafo, es decir, el subconjunto de nodos que están interconectados entre sí. También puede utilizarse para determinar si un grafo es bipartito. En la programación de juegos sencillos, como el buscaminas, el BFS se utiliza para expandir automáticamente áreas vacías al hacer clic en una celda sin minas, garantizando que toda la región contigua sea revelada de manera eficiente.

7. Comparación con Búsqueda en Profundidad (DFS)

El algoritmo de Búsqueda en Profundidad (DFS, Depth-First Search) es la contraparte conceptual principal del BFS. La distinción crucial entre ambos radica en la estructura de datos utilizada para gestionar la exploración: mientras que BFS utiliza una Cola (FIFO), DFS emplea una **Pila (Stack)**, que opera bajo el principio LIFO (Last-In, First-Out). Esta diferencia impone patrones de recorrido radicalmente distintos.

El BFS prioriza la exploración del ancho: expande la búsqueda horizontalmente, garantizando que el camino más corto sea encontrado primero. Por el contrario, el DFS prioriza la exploración de la profundidad: se adentra lo más posible en una rama o camino antes de retroceder (backtracking) y explorar ramas alternativas. Esta diferencia tiene implicaciones directas en la eficiencia y la idoneidad para ciertos problemas. Por ejemplo, si se sabe que la solución se encuentra a gran profundidad en un árbol muy estrecho, el DFS puede encontrarla mucho más rápido que el BFS, que se vería obligado a explorar todos los nodos superficiales primero.

En términos de complejidad de memoria, el DFS generalmente supera al BFS. Si se implementa de manera iterativa (o recursiva en el caso de árboles o grafos con profundidad limitada), el DFS solo necesita almacenar la ruta actual desde la raíz hasta el nodo en expansión, lo que resulta en una complejidad espacial de O(d), donde d es la profundidad máxima. El BFS, al tener que almacenar todos los nodos de un nivel completo en la cola, puede requerir mucha más memoria, especialmente en grafos anchos. Sin embargo, la ventaja definitiva del BFS es su garantía de optimalidad para el camino más corto, una propiedad que el DFS no posee.

8. Significado e Impacto

La Búsqueda en Amplitud no es solo un algoritmo; es un concepto fundacional que sirve como punto de partida para la comprensión de algoritmos de grafos más complejos y sofisticados. Su simplicidad, combinada con las garantías teóricas de completitud y optimalidad en grafos no ponderados, lo establece como una herramienta educativa esencial y una solución práctica robusta.

El impacto del BFS se extiende a la base de muchos algoritmos avanzados. Por ejemplo, el algoritmo de Dijkstra, utilizado para encontrar el camino más corto en grafos ponderados, puede verse como una generalización del BFS que utiliza una cola de prioridad en lugar de una cola simple para manejar los costos de las aristas. De manera similar, los algoritmos de búsqueda heurística como A* a menudo integran la lógica de exploración sistemática del BFS con funciones heurísticas para mejorar la eficiencia sin sacrificar la optimalidad.

En última instancia, el legado del BFS radica en su contribución a la modelización y resolución eficiente de problemas de conectividad en el mundo real. Desde la planificación de rutas en la

logística hasta la optimización de la estructura de la información en bases de datos y la exploración de espacios de estados en la inteligencia artificial, la Búsqueda en Amplitud sigue siendo un pilar conceptual que define la manera en que las máquinas interactúan y navegan por estructuras de datos complejas.

Lecturas Adicionales

Búsqueda en amplitud (Wikipedia en español)
Edward F. Moore (Wikipedia en inglés)
Grafo (matemáticas) (Wikipedia en español)
Algoritmo de Dijkstra (Wikipedia en español)