Universidad Americana UAM

Fundamentos de Robótica, grupo 7

Algoritmos de Búsqueda

Estudiante

Roberto Fabio Tercero Membreño (24014121)

Docente

MSc. José Durán García

23 de noviembre de 2025

Índice

1. INTRODUCCIÓN A LA TEORÍA DE BÚSQUEDA COMPUTACIONAL Y SU RELEVANCIA SISTÉMICA .. 4

1.1 Taxonomía y Fundamentos Matemáticos 5

2. ALGORITMOS DE BÚSQUEDA EN ESTRUCTURAS LINEALES 5

2.1 Búsqueda Lineal: Mecánica y Análisis de Hardware 6

2.1.1 Complejidad y Comportamiento Teórico 6

2.1.2 La Paradoja de la Eficiencia en Hardware Moderno 6

2.2 Variantes: Búsqueda Lineal con Centinela 7

3. BÚSQUEDA BINARIA: EL PARADIGMA LOGARÍTMICO 7

3.1 Mecánica Algorítmica y Precondiciones 8

3.2 Variaciones Avanzadas de la Búsqueda de Intervalo 8

3.2.1 Búsqueda de Interpolación 8

3.2.2 Búsqueda Exponencial (o Galopante) 8

3.3 Análisis Comparativo Estructurado: Lineal vs. Binaria 9

4. BÚSQUEDA EN ESTRUCTURAS NO LINEALES: GRAFOS Y ÁRBOLES 9

4.1 Búsqueda en Profundidad (DFS - Depth-First Search) 9

4.2 Búsqueda en Anchura (BFS - Breadth-First Search) 9

4.3 Búsqueda Informada: Algoritmo A (A-Star)* 10

5. IMPLEMENTACIONES EN BASES DE DATOS Y SISTEMAS DE ALMACENAMIENTO 10

5.1 B-Trees y B+ Trees: El Estándar Relacional 10

5.2 Árboles LSM (Log-Structured Merge) 10

6. MOTORES DE BÚSQUEDA Y RECUPERACIÓN DE INFORMACIÓN 11

6.1 El Índice Invertido 11

6.2 Algoritmos de Ranking: PageRank 11

7. INTELIGENCIA ARTIFICIAL Y BÚSQUEDA VECTORIAL MODERNA 11

7.1 Búsqueda en Árbol de Monte Carlo (MCTS) 11

7.2 Búsqueda Vectorial y HNSW 12

7.3 Indexación Geoespacial: Hexágonos H3 12

8. CONCLUSIÓN 12

REFERENCIAS 13

1. Introducción a la Teoría de Búsqueda Computacional y su Relevancia Sistémica

En el vasto y complejo panorama de las ciencias de la computación, pocos conceptos son tan omnipresentes y críticos como los algoritmos de búsqueda. Estos procedimientos no son meramente herramientas auxiliares; constituyen la columna vertebral sobre la que se erige la funcionalidad de los sistemas modernos de información. Desde la recuperación instantánea de un registro en una base de datos distribuida hasta la compleja toma de decisiones en inteligencias artificiales que compiten en juegos de estrategia milenarios, la capacidad de localizar un estado, un dato o un patrón específico dentro de un espacio de búsqueda define la eficiencia y la viabilidad de una solución tecnológica (Kanan, 2024).

La búsqueda, en su definición más abstracta, es el proceso sistemático de navegación a través de una estructura de datos o un espacio de problemas para identificar un elemento que cumpla con una condición objetivo predefinida, devolviendo su ubicación o declarando su ausencia si el espacio se agota sin éxito (GeeksforGeeks, 2024).

Este informe se propone desglosar, con una profundidad académica y técnica rigurosa, los mecanismos, ventajas, desventajas y aplicaciones del espectro completo de algoritmos de búsqueda. Se trascenderá la explicación superficial para analizar cómo estas construcciones matemáticas interactúan con el hardware moderno —incluyendo la predicción de ramas de la CPU y la jerarquía de memoria caché— y cómo evolucionan desde búsquedas lineales simples hasta arquitecturas de recuperación vectorial en modelos de lenguaje masivos (LLMs) (Kanan, 2024).

1.1 Taxonomía y Fundamentos Matemáticos

Los algoritmos de búsqueda se clasifican tradicionalmente según la naturaleza del espacio que exploran y la información de la que disponen a priori. Esta taxonomía divide los métodos en búsquedas sobre estructuras lineales (listas, arrays) y no lineales (grafos, árboles), así como en búsquedas no informadas (ciegas) frente a búsquedas informadas (heurísticas) (Kanan, 2024).

La complejidad computacional, expresada mediante la notación Big O, sirve como métrica estándar para evaluar la escalabilidad de estos algoritmos. Sin embargo, como se demostrará a lo largo de este documento, la complejidad asintótica teórica (O(n) vs O(\log n)) a menudo cuenta solo una parte de la historia. En la práctica, factores como la localidad de referencia, la vectorización SIMD (Single Instruction, Multiple Data) y las características de los medios de almacenamiento pueden invertir las expectativas teóricas, haciendo que algoritmos "lentos" en el papel superen a sus contrapartes "rápidas" en contextos específicos de baja latencia (Kanan, 2024).

2. Algoritmos de Búsqueda en Estructuras Lineales

La forma más elemental de búsqueda ocurre en estructuras de datos secuenciales, donde los elementos se organizan de manera contigua o enlazada linealmente. Aunque conceptualmente simples, estos algoritmos presentan matices profundos cuando se ejecutan en procesadores modernos superescalares.

2.1 Búsqueda Lineal: Mecánica y Análisis de Hardware

La búsqueda lineal, o búsqueda secuencial, es el método más intuitivo: el algoritmo itera a través de la colección, comenzando desde el primer elemento, y compara cada ítem con el valor objetivo hasta encontrar una coincidencia o alcanzar el final de la estructura (GeeksforGeeks, 2024).

2.1.1 Complejidad y Comportamiento Teórico

Su complejidad temporal en el peor caso es O(n), donde n es el número de elementos. Esto ocurre cuando el objetivo está en la última posición o no existe en absoluto. En el mejor caso, donde el objetivo es el primer elemento, la complejidad es O(1). El caso promedio, asumiendo una distribución uniforme de probabilidad de la posición del objetivo, implica n/2 comparaciones, lo que asintóticamente se mantiene en O(n) (GeeksforGeeks, 2024). La característica definitoria de la búsqueda lineal es su universalidad: no impone ningún requisito sobre el orden de los datos. Funciona indistintamente en listas desordenadas, listas enlazadas o flujos de datos en tiempo real donde el acceso aleatorio no es posible.

2.1.2 La Paradoja de la Eficiencia en Hardware Moderno

A pesar de su ineficiencia teórica para grandes conjuntos de datos, la búsqueda lineal supera frecuentemente a algoritmos más complejos como la búsqueda binaria en conjuntos de datos pequeños o medianos. Este fenómeno se explica por la arquitectura de los CPUs modernos (Kanan, 2024):

1. **Predicción de Ramas (Branch Prediction):** Los procesadores modernos utilizan pipelines profundos y ejecutan instrucciones especulativamente. En una búsqueda lineal, la condición de salida del bucle es falsa repetidamente. Los predictores de rama aprenden este patrón y pre-cargan las instrucciones siguientes con alta eficiencia. La búsqueda binaria implica ramificaciones condicionales impredecibles, causando fallos en la predicción.
2. **Localidad de Referencia y Pre-fetching:** Cuando se recorre un array contiguo, la búsqueda lineal accede a la memoria de manera secuencial. Esto maximiza la utilidad de las líneas de caché L1 y L2.
3. **Vectorización:** La simplicidad de la búsqueda lineal permite a los compiladores modernos aplicar optimizaciones SIMD, comparando múltiples elementos en un solo ciclo de reloj.

2.2 Variantes: Búsqueda Lineal con Centinela

Una optimización común es la técnica del centinela. Al colocar el valor objetivo al final del array (o en una posición "centinela" adicional), se garantiza que la búsqueda siempre tendrá éxito, eliminando la necesidad de verificar el límite del array en cada paso del bucle. Esto reduce el número de comparaciones a la mitad, mejorando el rendimiento en situaciones de bucles muy ajustados (GeeksforGeeks, 2024).

3. Búsqueda Binaria: El Paradigma Logarítmico

La búsqueda binaria representa un salto cualitativo en eficiencia algorítmica, aplicando la estrategia de "divide y vencerás" para reducir el espacio de búsqueda de manera exponencial.

3.1 Mecánica Algorítmica y Precondiciones

El requisito ineludible para la búsqueda binaria es que la colección debe estar ordenada monotónicamente. El algoritmo selecciona el elemento medio del rango actual y lo compara con el objetivo. Si coinciden, termina; si es menor o mayor, descarta la mitad correspondiente y repite el proceso (GeeksforGeeks, 2024). Esta eliminación sistemática resulta en una complejidad temporal de O(\log n).

3.2 Variaciones Avanzadas de la Búsqueda de Intervalo

La búsqueda binaria es la base de una familia de algoritmos de intervalo diseñados para contextos específicos:

3.2.1 Búsqueda de Interpolación

Diseñada para datos distribuidos uniformemente, la búsqueda de interpolación no selecciona el punto medio, sino que estima la posición probable del objetivo basándose en su valor numérico. En condiciones ideales, su complejidad se reduce a O(\log(\log n)), aunque en distribuciones sesgadas puede degradarse a $O(n)$ (GeeksforGeeks, 2024).

3.2.2 Búsqueda Exponencial (o Galopante)

Esta variante es crucial cuando se busca en listas no acotadas o de tamaño desconocido. El algoritmo busca el rango donde reside el objetivo probando índices en potencias de dos. Una vez encontrado el intervalo, se ejecuta una búsqueda binaria estándar dentro de ese rango limitado (GeeksforGeeks, 2024).

3.3 Análisis Comparativo Estructurado: Lineal vs. Binaria

La elección entre estos dos paradigmas implica consideraciones sobre la estructura de datos subyacente y la frecuencia de actualización. Mientras que la búsqueda binaria ofrece un O(\log n) superior para datos estáticos masivos, su ventaja se diluye si los datos cambian constantemente, ya que el costo de mantener el orden (O(n) por inserción) supera el beneficio de la búsqueda rápida (Kanan, 2024).

4. Búsqueda en Estructuras No Lineales: Grafos y Árboles

Para navegar estructuras relacionales como redes sociales o mapas, se emplean algoritmos de recorrido de grafos.

4.1 Búsqueda en Profundidad (DFS - Depth-First Search)

El DFS adopta una estrategia agresiva de exploración, avanzando tan profundamente como sea posible a lo largo de cada rama antes de retroceder. Utiliza una estructura LIFO (pila). Su consumo de memoria es proporcional a la profundidad máxima del árbol (O(d)). Es el algoritmo de elección para el análisis topológico y resolución de laberintos (GeeksforGeeks, 2024).

4.2 Búsqueda en Anchura (BFS - Breadth-First Search)

El BFS explora el grafo capa por capa. Se apoya en una estructura FIFO (cola). En grafos no ponderados, BFS garantiza encontrar el camino más corto desde el origen hasta el objetivo. Su principal desventaja es el consumo de memoria, que puede ser exponencial, ya que debe almacenar todos los nodos de la "frontera" actual (GeeksforGeeks, 2024).

4.3 Búsqueda Informada: Algoritmo A (A-Star)*

Mientras que BFS y DFS son algoritmos "ciegos", A* utiliza conocimiento del dominio para guiar la búsqueda mediante la función f(n) = g(n) + h(n). Si la heurística es admisible, A* garantiza encontrar el camino óptimo explorando significativamente menos nodos, siendo el motor estándar para la navegación en videojuegos y robótica (GeeksforGeeks, 2024).

5. Implementaciones en Bases de Datos y Sistemas de Almacenamiento

Los algoritmos teóricos se transforman radicalmente cuando se aplican al almacenamiento persistente, donde la latencia del disco es el cuello de botella dominante.

5.1 B-Trees y B+ Trees: El Estándar Relacional

En bases de datos como PostgreSQL u Oracle, se utilizan B-Trees, árboles auto-balanceados donde cada nodo puede tener cientos de hijos. Esto reduce la altura del árbol, permitiendo encontrar registros con pocos accesos a disco. Los B+ Trees optimizan las consultas de rango al mantener los datos solo en las hojas enlazadas secuencialmente (Built In, 2024).

5.2 Árboles LSM (Log-Structured Merge)

Bases de datos como Cassandra utilizan árboles LSM para optimizar escrituras masivas. En lugar de modificar una estructura en disco, las escrituras se agregan secuencialmente en memoria (Memtable) y luego se vuelcan al disco como archivos inmutables (SSTable). Aunque la lectura puede ser más lenta, el uso de Filtros de Bloom mitiga el costo (Built In, 2024).

6. Motores de Búsqueda y Recuperación de Información

La búsqueda en la web presenta desafíos únicos de escala y semántica.

6.1 El Índice Invertido

Es la estructura fundamental de motores como Google. Invierte la relación Documento -> Palabras a Palabra -> Lista de Documentos. Esto convierte la búsqueda en un problema de intersección de listas ordenadas, una operación extremadamente rápida (GeeksforGeeks, 2024).

6.2 Algoritmos de Ranking: PageRank

PageRank trata la web como un grafo gigante, asignando una puntuación de importancia a cada página basada en la cantidad y calidad de los enlaces que apuntan a ella. Esto permite priorizar sitios de autoridad sobre sitios irrelevantes (GeeksforGeeks, 2024).

7. Inteligencia Artificial y Búsqueda Vectorial Moderna

La última frontera se encuentra en la intersección con el aprendizaje automático.

7.1 Búsqueda en Árbol de Monte Carlo (MCTS)

Popularizado por AlphaGo, MCTS construye el árbol de búsqueda asimétricamente, priorizando las ramas más prometedoras mediante simulaciones ("playouts") e integración con redes neuronales para evaluar posiciones sin necesidad de simular hasta el final (GeeksforGeeks, 2024).

7.2 Búsqueda Vectorial y HNSW

En la era de los LLMs, la búsqueda es semántica. Los textos se transforman en vectores numéricos (embeddings). Para buscar el "vecino más cercano" eficientemente, se utiliza el índice HNSW (Hierarchical Navigable Small World), que crea un grafo multicapa de "mundo pequeño" para búsquedas aproximadas ultra-rápidas, fundamentales para sistemas RAG (Neon, 2024).

7.3 Indexación Geoespacial: Hexágonos H3

Uber revolucionó la búsqueda geoespacial con el sistema H3, que discretiza el mundo en una rejilla jerárquica de hexágonos. A diferencia de los cuadrados, los hexágonos tienen la propiedad de que todos sus vecinos están a la misma distancia del centro, simplificando enormemente los algoritmos de análisis espacial y fijación de precios dinámicos (Uber, s.f.).

8. Conclusión

La evolución de los algoritmos de búsqueda refleja la historia de la computación: una búsqueda constante de eficiencia. Desde la simplicidad lineal hasta las fronteras probabilísticas de la búsqueda vectorial, la lección fundamental es que no existe un algoritmo universalmente superior. La maestría reside en comprender las compensaciones entre latencia, memoria y precisión que cada implementación conlleva.

Referencias

Built In. (2024). *How Database B-Tree Indexing Works*. https://builtin.com/data-science/b-tree-index

GeeksforGeeks. (2024). *Searching Algorithms*. https://www.geeksforgeeks.org/dsa/searching-algorithms/

Kanan, A. (2024, 27 de enero). *Understanding search algorithms in computer science*. Medium. https://medium.com/@alikanan-a/understanding-search-algorithms-in-computer-science-ec62f845cd7a

Neon. (2024). *Understanding vector search and HNSW index with pgvector*. https://neon.com/blog/understanding-vector-search-and-hnsw-index-with-pgvector

Uber. (s.f.). *H3: Uber's Hexagonal Hierarchical Spatial Index*. Uber Engineering. https://www.uber.com/blog/h3/