Algoritmos de Búsqueda

Investigación Operativa

Universidad de SanAndrés

¿Qué son los algoritmos de búsqueda?

Herramientas fundamentales en la resolución de problemas de optimización y toma de decisiones.

Árboles y Grafos

BFS, DFS, Dijkstra, A*

Listas

Secuencial, Binaria

Tipos de algoritmos de búsqueda

Algoritmos para árboles y grafos:

- **Búsqueda Ciega**: BFS, DFS
- **Búsqueda Informada**: A*, Greedy Best-First
- **Dijkstra**: Búsqueda de costo uniforme

Tipos de algoritmos de búsqueda

Algoritmos para listas:

- **Búsqueda Secuencial**: Listas desordenadas
- **Búsqueda Binaria**: Listas ordenadas

Conceptos básicos de grafos

Nodo

Elemento fundamental

Arista

Conexion entre nodos

Camino

Secuencia de nodos

No Dirigido

Sin direccion

Ponderado

Con pesos

Notación Big O

Complejidades más comunes:

- $O(1)$ - **Constante**: No depende del tamaño

Notación Big O

Complejidades más comunes:

- $O(1)$ - **Constante**: No depende del tamaño
- $O(\log n)$ - **Logarítmica**: Muy eficiente

Notación Big O

Complejidades más comunes:

- $O(1)$ - **Constante**: No depende del tamaño
- $O(\log n)$ - **Logarítmica**: Muy eficiente
- $O(n)$ - **Lineal**: Proporcional al tamaño

Notación Big O

Complejidades más comunes:

- $O(1)$ - **Constante**: No depende del tamaño
- $O(\log n)$ - **Logarítmica**: Muy eficiente
- $O(n)$ - **Lineal**: Proporcional al tamaño
- $O(n^2)$ - **Cuadrática**: Como ordenamiento simple

Notación Big O

Complejidades más comunes:

- $O(1)$ - **Constante**: No depende del tamaño
- $O(\log n)$ - **Logarítmica**: Muy eficiente
- $O(n)$ - **Lineal**: Proporcional al tamaño
- $O(n^2)$ - **Cuadrática**: Como ordenamiento simple
- $O(2^n)$ - **Exponencial**: Muy ineficiente

Búsquedas No Informadas (Ciegas)

Características:

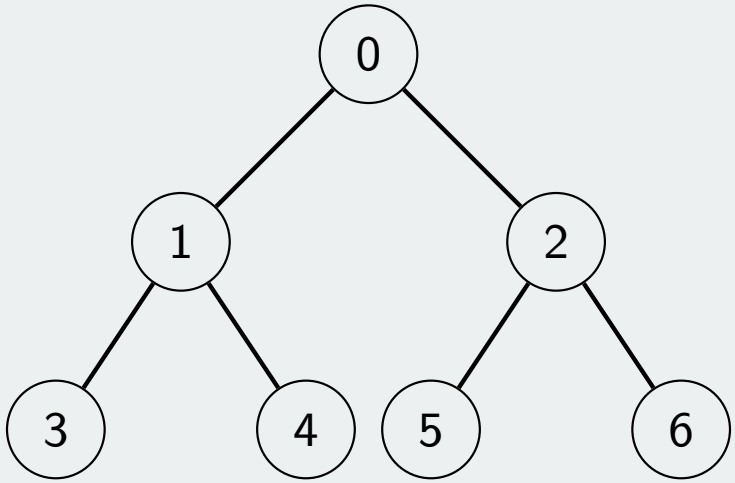
- No tienen información adicional
- No tienen función de evaluación
- Exploran sistemáticamente
- Garantizan encontrar la solución

Breadth-First Search (BFS)

Características principales:

- Explora nivel por nivel
- Usa cola FIFO
- Garantiza camino mas corto
- Complejidad: $O(V + E)$

Orden de exploración:



Ejemplo BFS: Buscando el premio

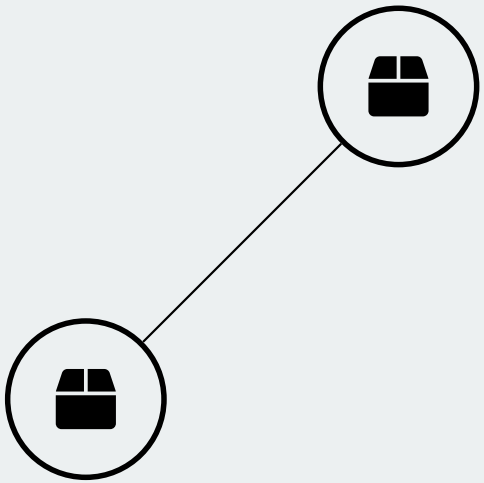
Problema: Tenemos una caja con cajas adentro que no sabemos cuántas hay, ni qué tantas cajas y subcajas tenemos adentro. En algún momento hay un regalo, pero no sabemos cuándo vamos a llegar a él.

¿Cómo funciona BFS?

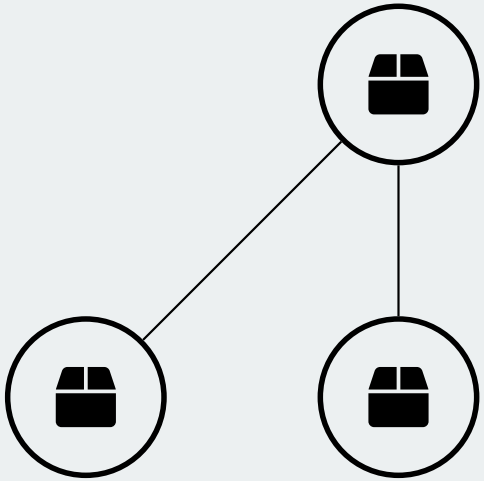
- Abrimos todas las cajas del primer nivel
- Luego todas las cajas del segundo nivel
- Y así sucesivamente nivel por nivel

BFS

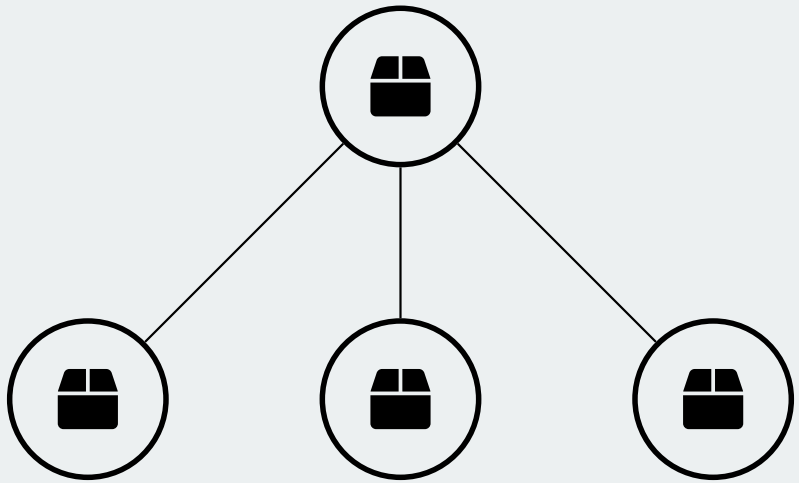
BFS



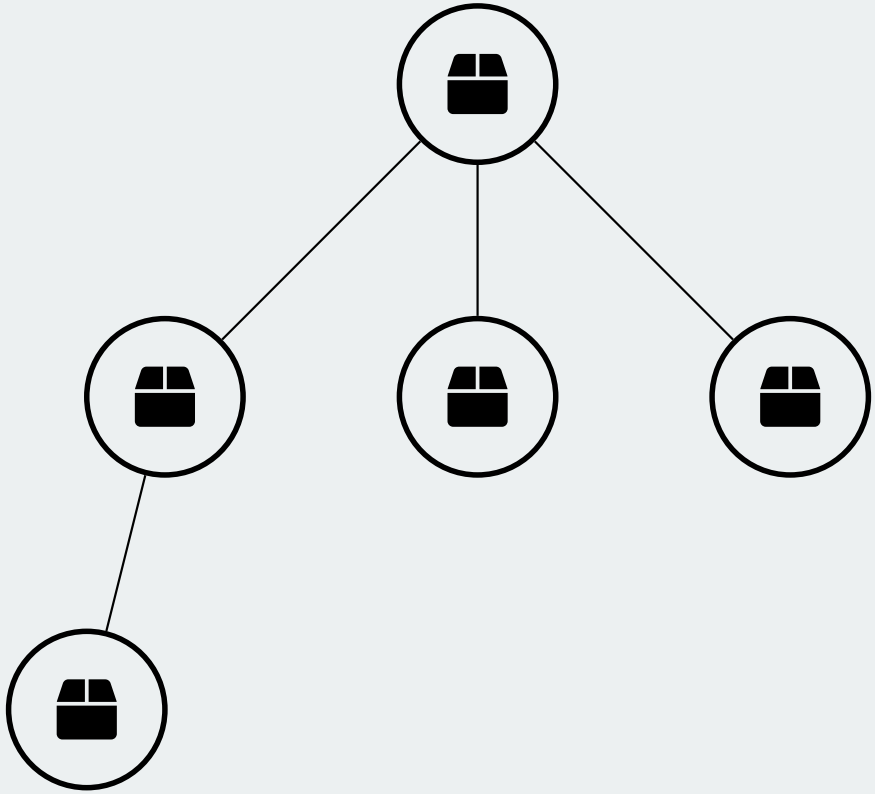
BFS



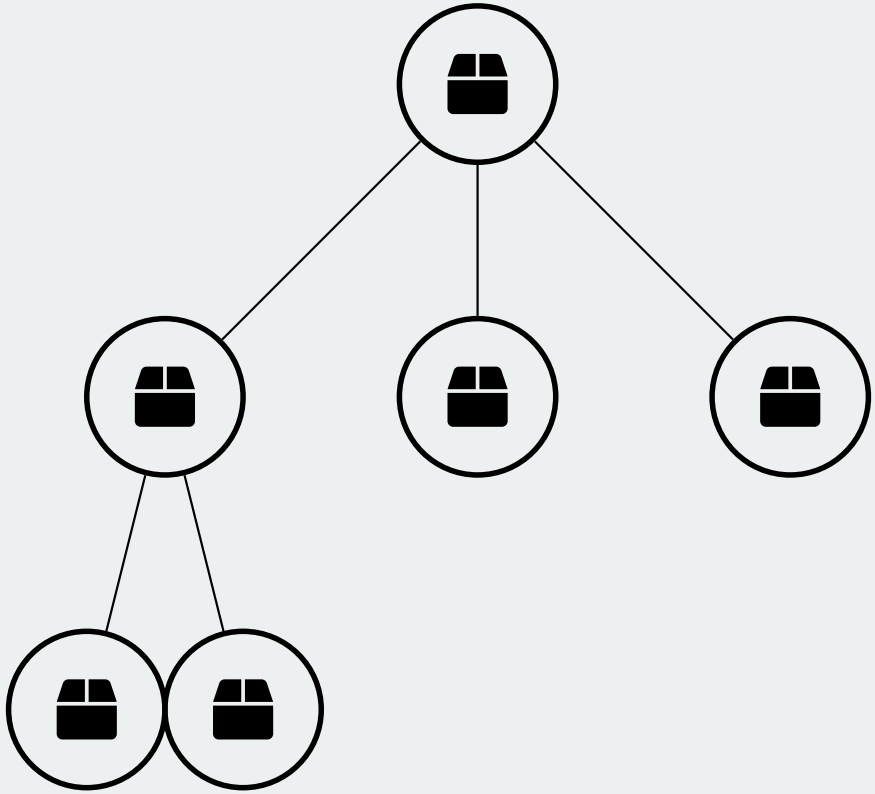
BFS



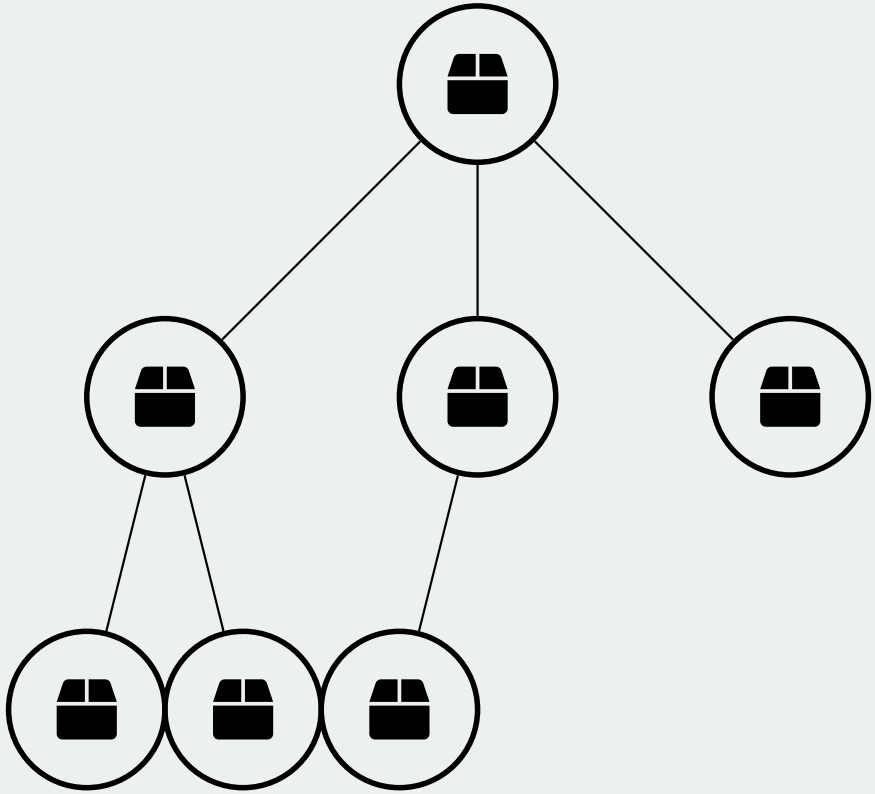
BFS



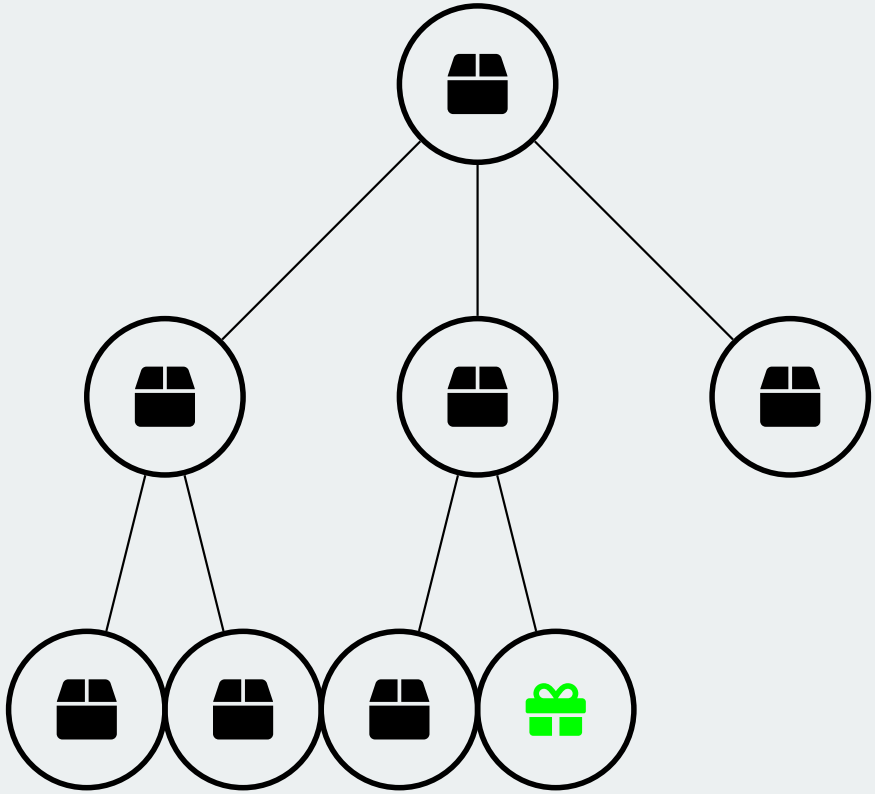
BFS



BFS



BFS

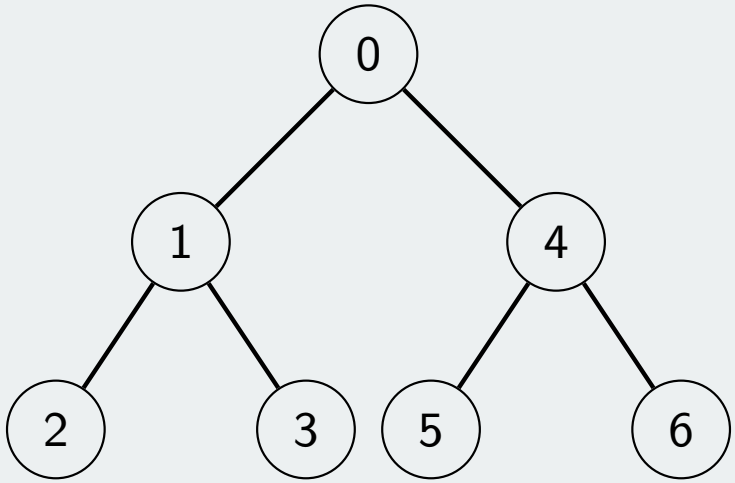


Depth-First Search (DFS)

Características principales:

- Explora una rama completa
- Usa pila LIFO
- No garantiza camino mas corto
- Complejidad: $O(V + E)$

Orden de exploración:



Ejemplo DFS: Buscando el premio

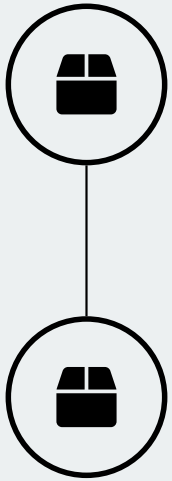
Problema: Tenemos una caja con cajas adentro que no sabemos cuántas hay, ni qué tantas cajas y subcajas tenemos adentro. En algún momento hay un regalo, pero no sabemos cuándo vamos a llegar a él.

¿Cómo funciona DFS?

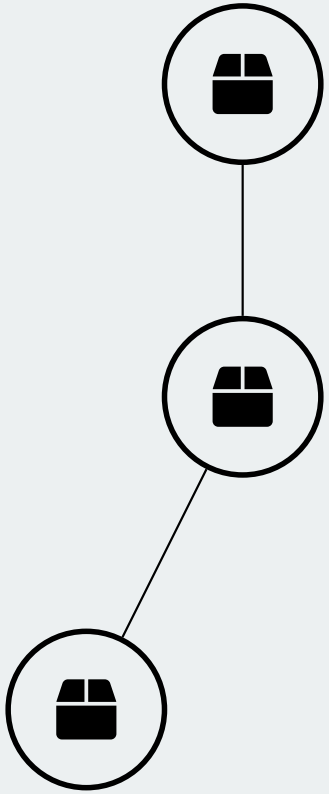
- Abrimos una caja y nos metemos en ella
- Si tiene más cajas, seguimos metiéndonos
- Solo retrocedemos cuando llegamos al final de una rama

DFS

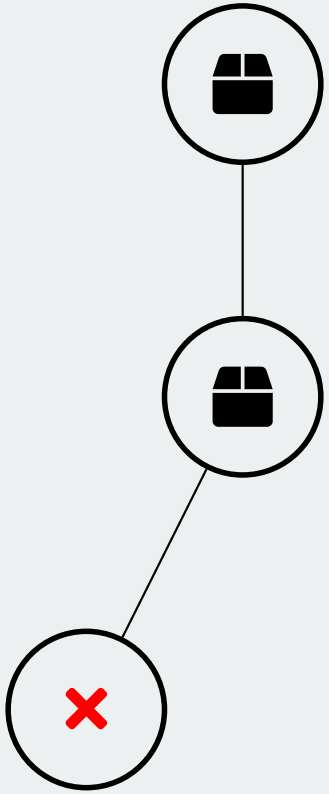
DFS



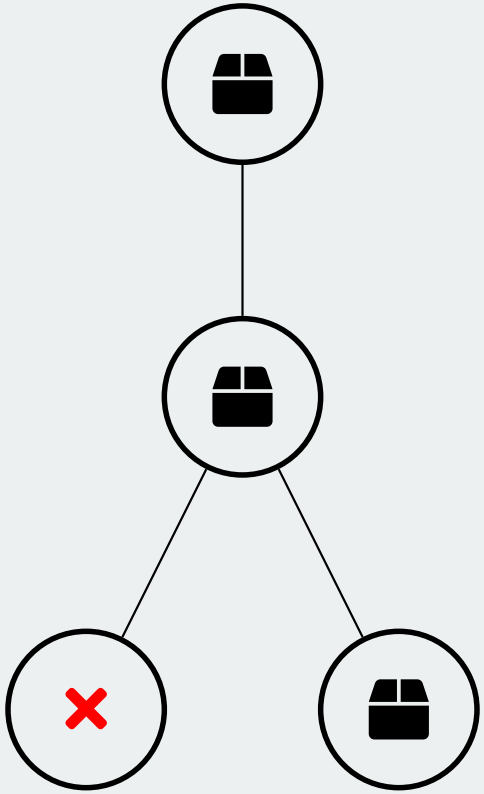
DFS



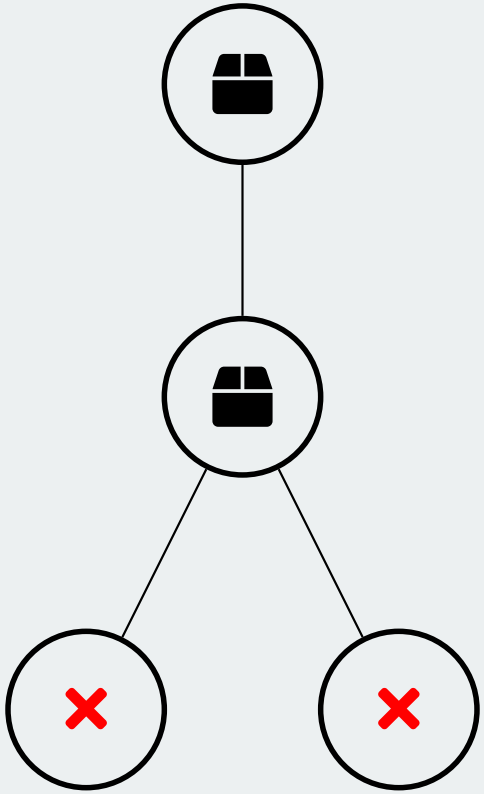
DFS



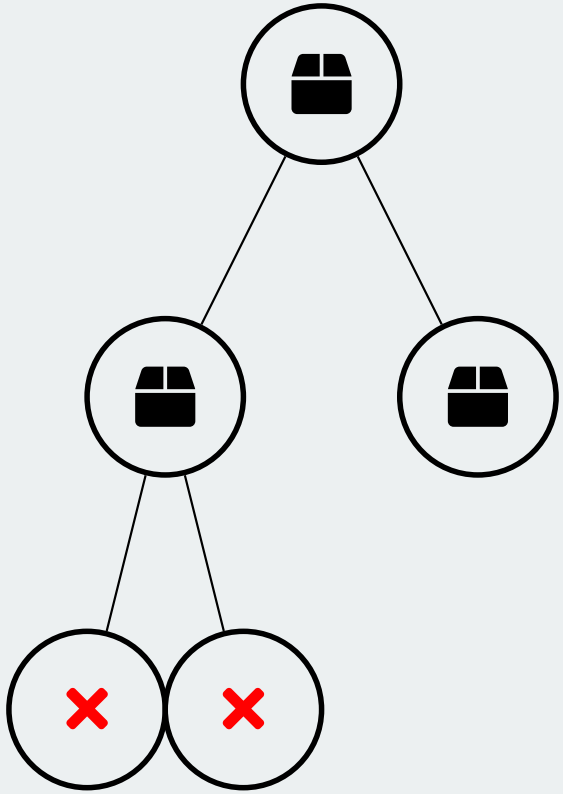
DFS



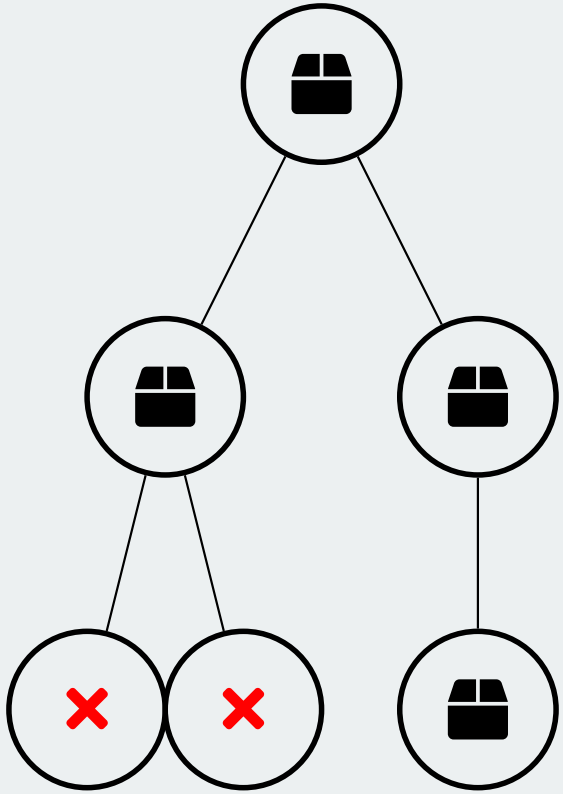
DFS



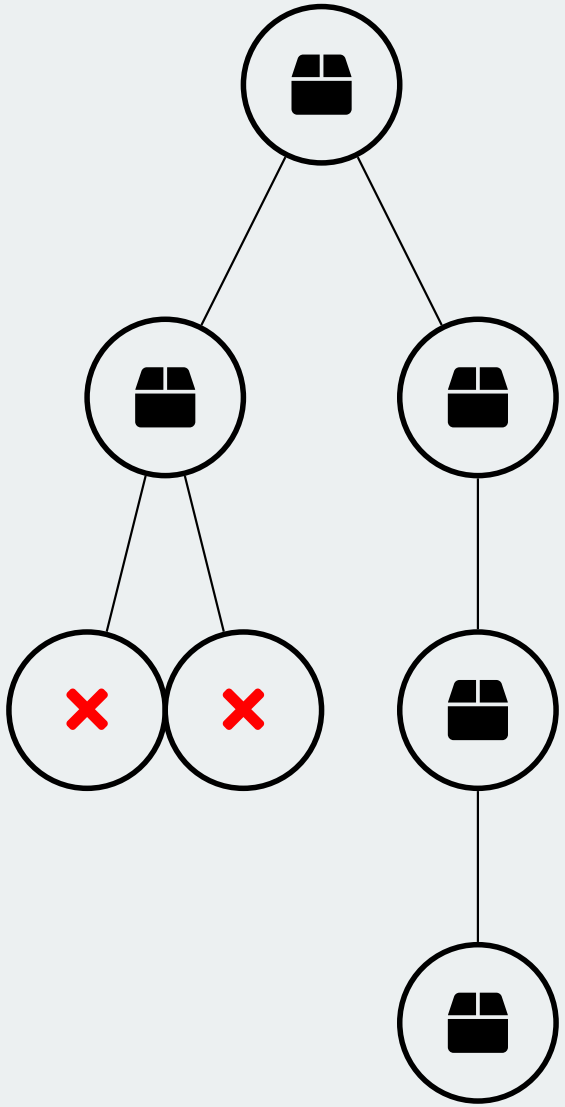
DFS



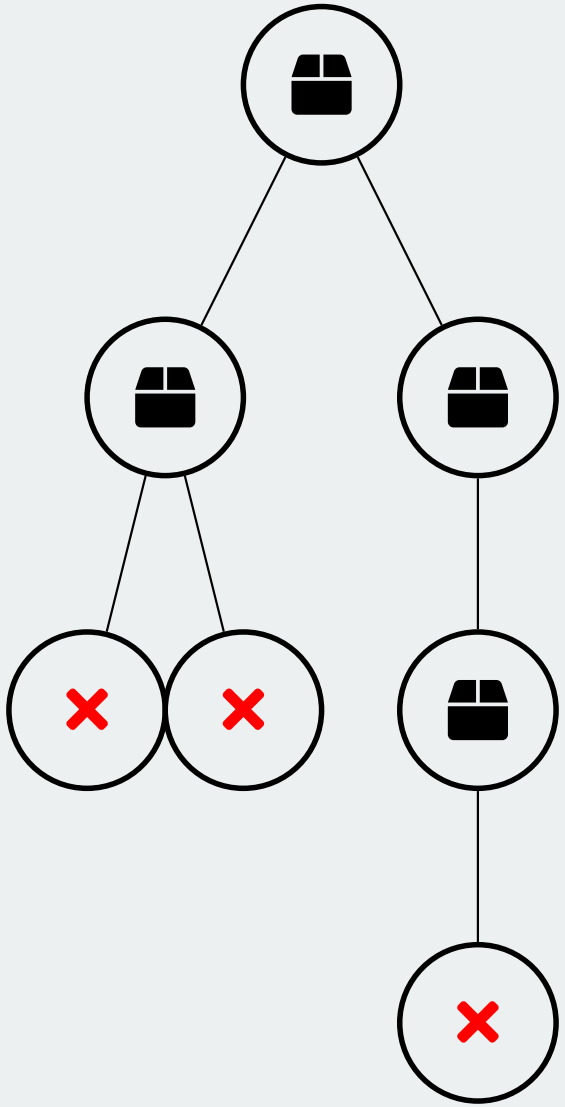
DFS



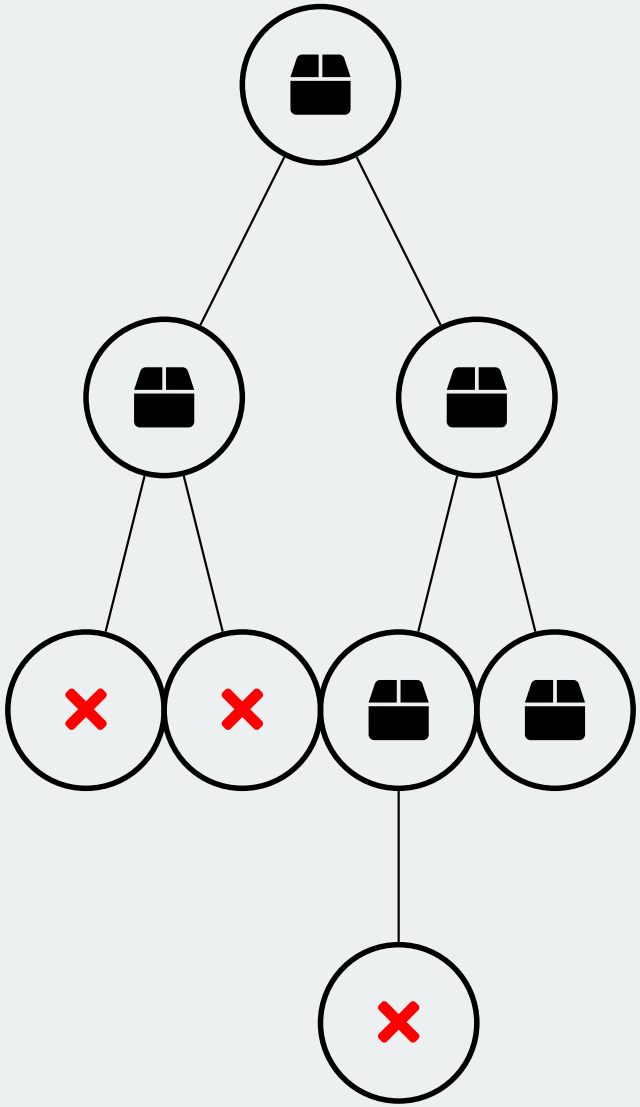
DFS



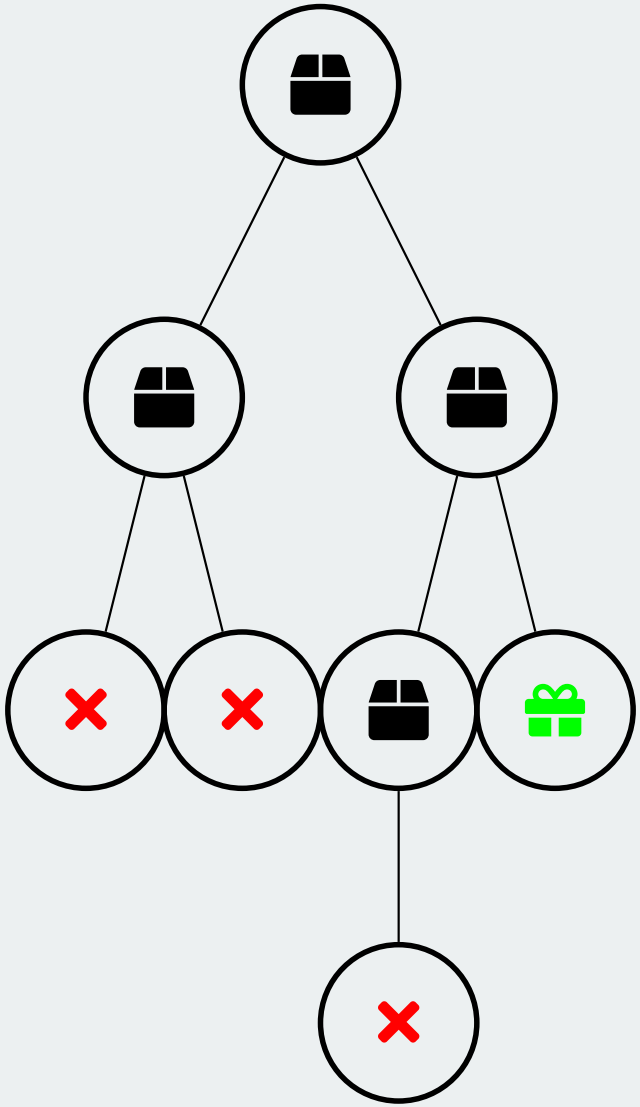
DFS



DFS



DFS



Comparación BFS vs DFS

BFS	DFS
Nivel por nivel	Rama completa

¿Cuándo usar cada uno?

- **BFS**: Cuando la solución está cerca del inicio
- **DFS**: Cuando la solución está profunda en el árbol
- **Problema**: Nunca sabemos donde está el premio

Algoritmo de Dijkstra

Propósito: Encontrar el camino más corto desde un nodo origen a **todos** los demás nodos en un grafo ponderado con pesos no negativos.

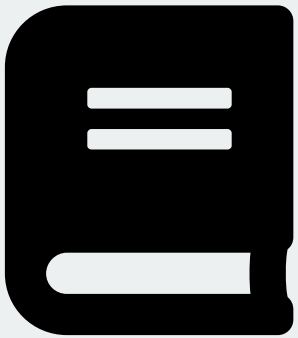
Características:

- Garantiza camino más corto
- Funciona con pesos positivos
- Complejidad: $O(V^2)$ o $O(E + V \log V)$
- Base para algoritmos más avanzados

Búsquedas Informadas

Características:

- Tienen información adicional
- Usan función de evaluación
- Exploran menos nodos
- Pueden no ser óptimas



A* (A-Star)

Función de evaluación:

$$f(n) = g(n) + h(n)$$

- $g(n)$: Costo desde inicio hasta n
- $h(n)$: Heurística de n hasta meta
- $f(n)$: Costo total estimado

Ventajas:

- Más eficiente que Dijkstra
- Garantiza optimalidad si $h(n)$ es admisible

Greedy Best-First Search

Función de evaluación:

$$f(n) = h(n)$$

Características:

- Solo considera heurística
- No garantiza camino óptimo
- Puede ser más rápido
- Puede quedar atrapado

Comparación con A*:

- A* considera costo acumulado + heurística
- GBFS solo considera heurística

Búsquedas en Listas

Secuencial

$O(n)$ - No ordenada

Binaria

$O(\log n)$ - Ordenada

Búsqueda Secuencial

Características:

- No requiere ordenamiento
- Revisa elemento por elemento
- Complejidad: $O(n)$
- Peor caso: último elemento

Ejemplo: Buscar una figurita de Messi en una bolsa de figuritas del mundial.

Búsqueda Binaria

Características:

- Requiere lista ordenada
- Divide y conquista
- Complejidad: $O(\log n)$
- Muy eficiente

Ejemplo: Buscar un número en una guía telefónica.

Aplicaciones Prácticas - BFS y DFS

Breadth-First Search y Depth-First Search:

Ejemplo: Estamos en París y queremos comer una baguette. Salimos del hotel y nos encontramos con una calle que se llama Rue de Falopini. Buscamos en esa cuadra y no hay nada, giramos a la derecha y buscamos y seguimos girando y buscando...

- **DFS**: Caminar la Rue de Falopini hasta el fondo
- **BFS**: Girar y buscar en la de al lado

Aplicación de negocios: Segmentación de marketing

- **DFS**: Fitness $\rightarrow$ fitness vegano $\rightarrow$ fitness vegano de embarazadas
- **BFS**: Fitness $\rightarrow$ ropa $\rightarrow$ computadoras

Aplicaciones Prácticas - Dijkstra, A* y GBFS

Dijkstra, A* y Greedy Best-First Search:

Aplicaciones principales:

- Rutas de entrega o logística para optimizar costos de transporte
- Calcular rutas de transporte de datos entre torres
- Optimización de costos en telecomunicaciones
- Cualquier problema de optimización de rutas

¿Cuándo usar cada uno?

- **Dijkstra**: Si no tenemos idea donde está la meta
- **A***: Si tenemos forma de estimar la meta
- **GBFS**: Para sugerir algo rápido sin importar que sea óptimo

Aplicaciones Prácticas - Búsqueda en Listas

Búsqueda Secuencial:

- La usamos cuando no tenemos las cosas ordenadas
- Como buscar un elemento en una bolsa

Búsqueda Binaria:

- Para encontrar un momento en el tiempo en que se produjo un evento
- Si tenemos un video de seguridad y queremos encontrar cuando se robaron una bici

Ejemplo: Búsqueda en video de seguridad

Problema: Encontrar el momento exacto del robo de mi bicicleta y el video es desde el comienzo de los tiempos, el Big Bang.

Solución con búsqueda binaria:

$$\text{pasos} = \left\lceil \log_2 \left(\frac{\text{rango de tiempo}}{\text{precision}} \right) \right\rceil$$

- Para encontrar al segundo: 59 pasos
- Para encontrar al milisegundo: 69 pasos
- Tiempo total: $\approx$ 3 minutos

¡God bless binary search!

Terminamos

¿Dudas?

¿Consultas?