



Ayudantía 4

Búsqueda: DFS y BFS

Por Martín Lagies y Willy Pugh

15 de abril 2024



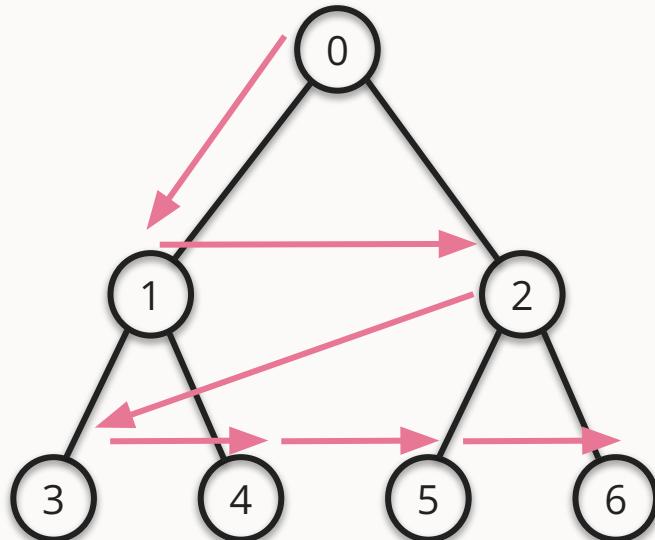
Contenidos

1. Algoritmos BFS y DFS
2. Repaso de búsqueda genérica
3. Ejemplos de BFS y DFS
4. Más algoritmos!

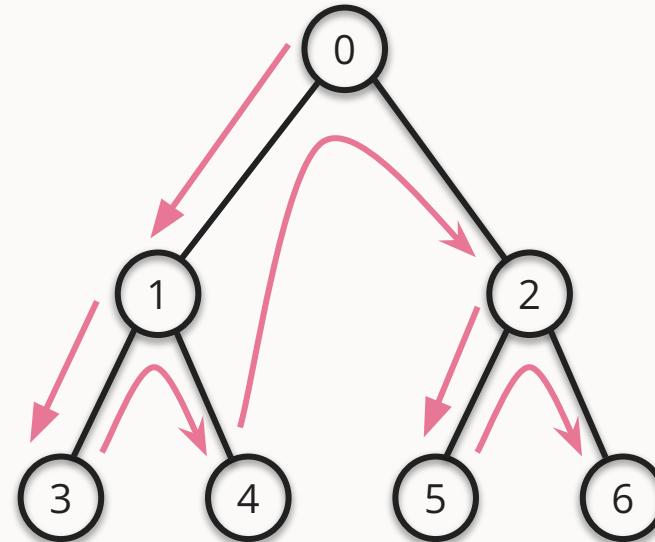


Algoritmos BFS y DFS

BFS - BREADTH FIRST SEARCH
Búsqueda en amplitud



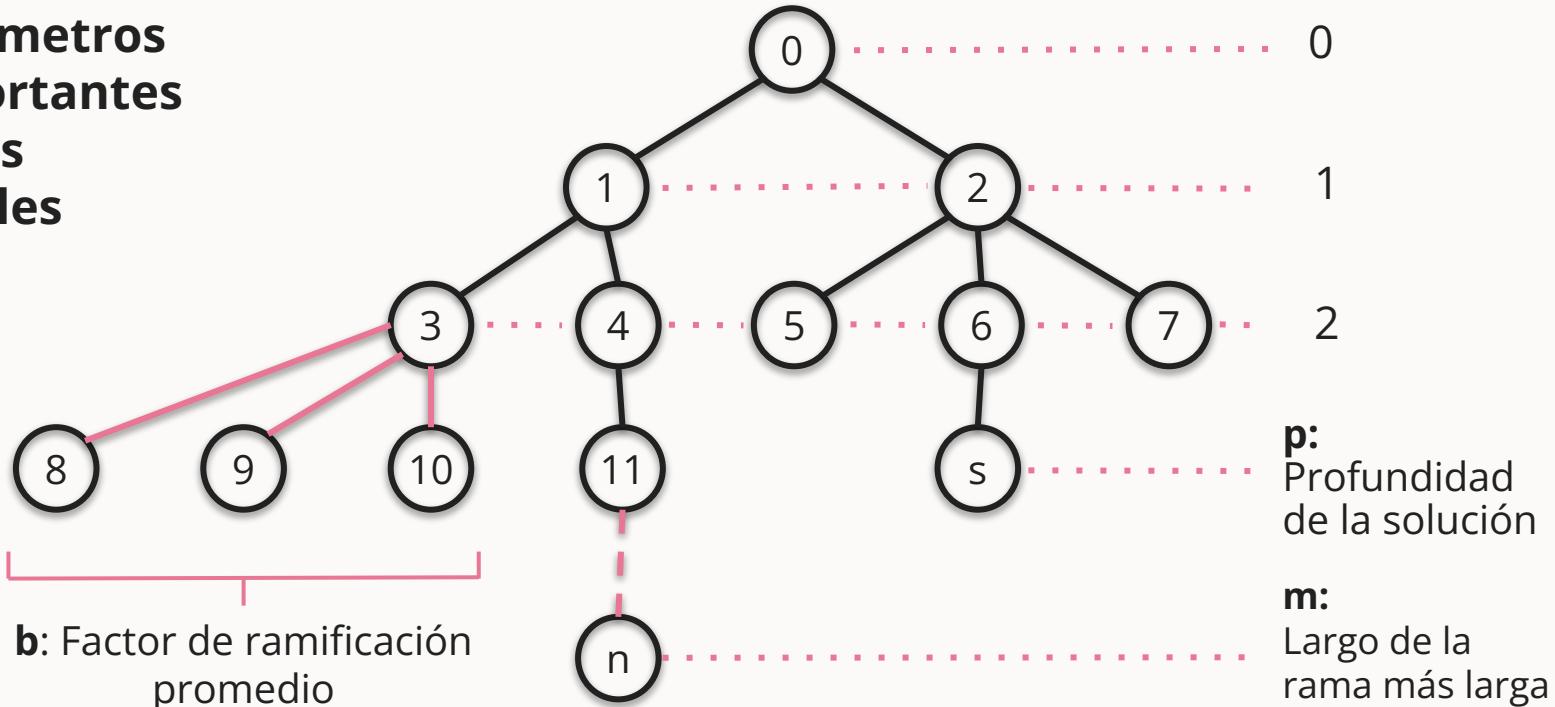
DFS - DEPTH FIRST SEARCH
Búsqueda en profundidad





Algoritmos BFS y DFS

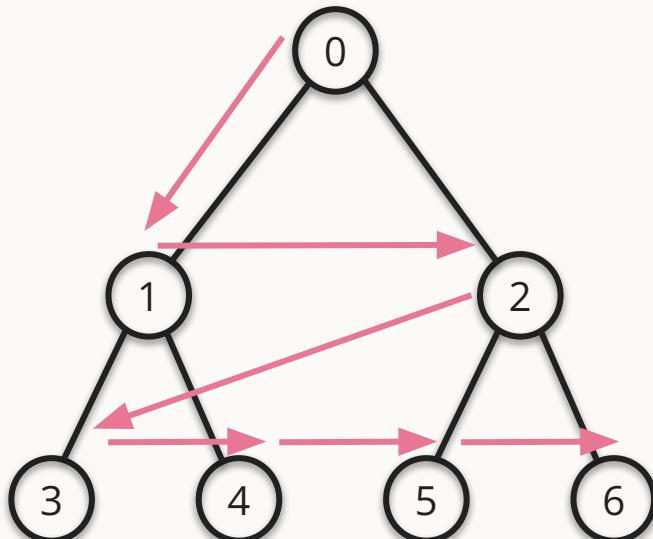
Parámetros importantes de los árboles





Algoritmos BFS y DFS

BFS - BREADTH FIRST SEARCH Búsqueda en amplitud



Memoria usada: $O(b^p)$
Tiempo requerido: $O(b^p)$

b: Factor de ramificación promedio
p: Profundidad de la solución
m: Largo de la rama más larga

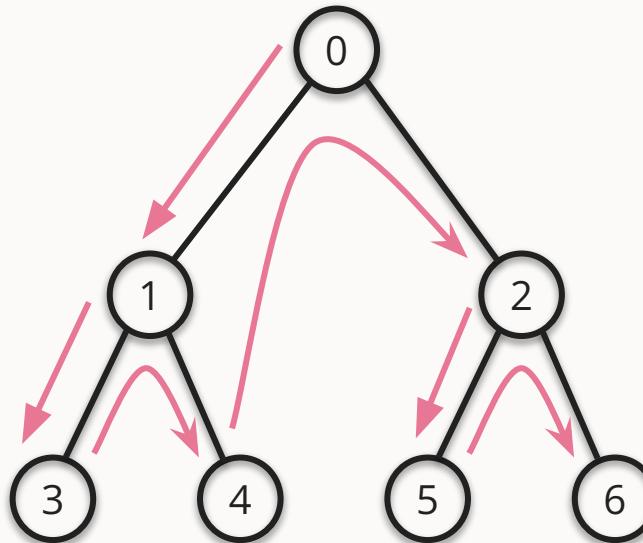


Algoritmos BFS y DFS

Memoria usada: $O(bm)$
Tiempo requerido: $O(b^m)$

b: Factor de ramificación promedio
p: Profundidad de la solución
m: Largo de la rama más larga

DFS - DEPTH FIRST SEARCH
Búsqueda en profundidad





Algoritmos BFS y DFS

BFS - BREADTH FIRST SEARCH

Búsqueda en amplitud

DFS - DEPTH FIRST SEARCH

Búsqueda en profundidad

TEOREMAS

Si el espacio de estados es finito, **búsqueda en amplitud** es **completo** y **óptimo** para problemas de búsqueda con costos uniformes

Si el espacio de estados es finito, **búsqueda en profundidad** con detección de ciclos es **completo**



IDDFS

Profundidad Limitada

Agregamos un parámetro l a DFS, es la **profundidad máxima** a la que buscará

Con esta implementación modificada de DFS, ahora el algoritmo no solo es completo, si no que también encuentra el camino **óptimo**.

Profundización Iterativa

1. Hacemos una búsqueda DFS con profundidad l (comenzando por $l=1$)
2. Si no hubo éxito, se incrementa l y repetimos

Esto es asumiendo que existe un nodo objetivo y una cantidad finita de estados

Algoritmos de Búsqueda

Búsqueda Genérica:

Input: Un problema de búsqueda (S, A, S_{init}, G)

Output: Un nodo objetivo

- 1 $Open$ es un contenedor vacío
- 2 $Closed$ es un conjunto vacío
- 3 Inserta S_{init} a $Open$
- 4 $\text{parent}(S_{init}) = \text{null}$
- 5 **while** $Open \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(Open)$
- 7 Inserta u en $Closed$
- 8 **for each** $v \in \text{Succ}(u) \setminus (Open \cup$
 $Closed)$
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a $Open$



Open (o frontera) contiene todos los nodos generados pero no expandidos

Closed contiene los nodos expandidos

Nodo expandido: sucesores calculados

Algoritmos de Búsqueda

Búsqueda Genérica:

Input: Un problema de búsqueda (S, A, S_{init}, G)

Output: Un nodo objetivo

- 1 $Open$ es un contenedor vacío
- 2 $Closed$ es un conjunto vacío
- 3 Inserta S_{init} a $Open$
- 4 $parent(S_{init}) = null$
- 5 **while** $Open \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(Open)$
- 7 Inserta u en $Closed$
- 8 **for each** $v \in \text{Succ}(u) \setminus (Open \cup$
- 9 $Closed)$
- 10 $parent(v) = u$
- 11 **if** $v \in G$ **return** v
- 12 Inserta v a $Open$



La diferencia en la implementación de DFS y BFS es la estructura de datos que se utiliza para $Open$

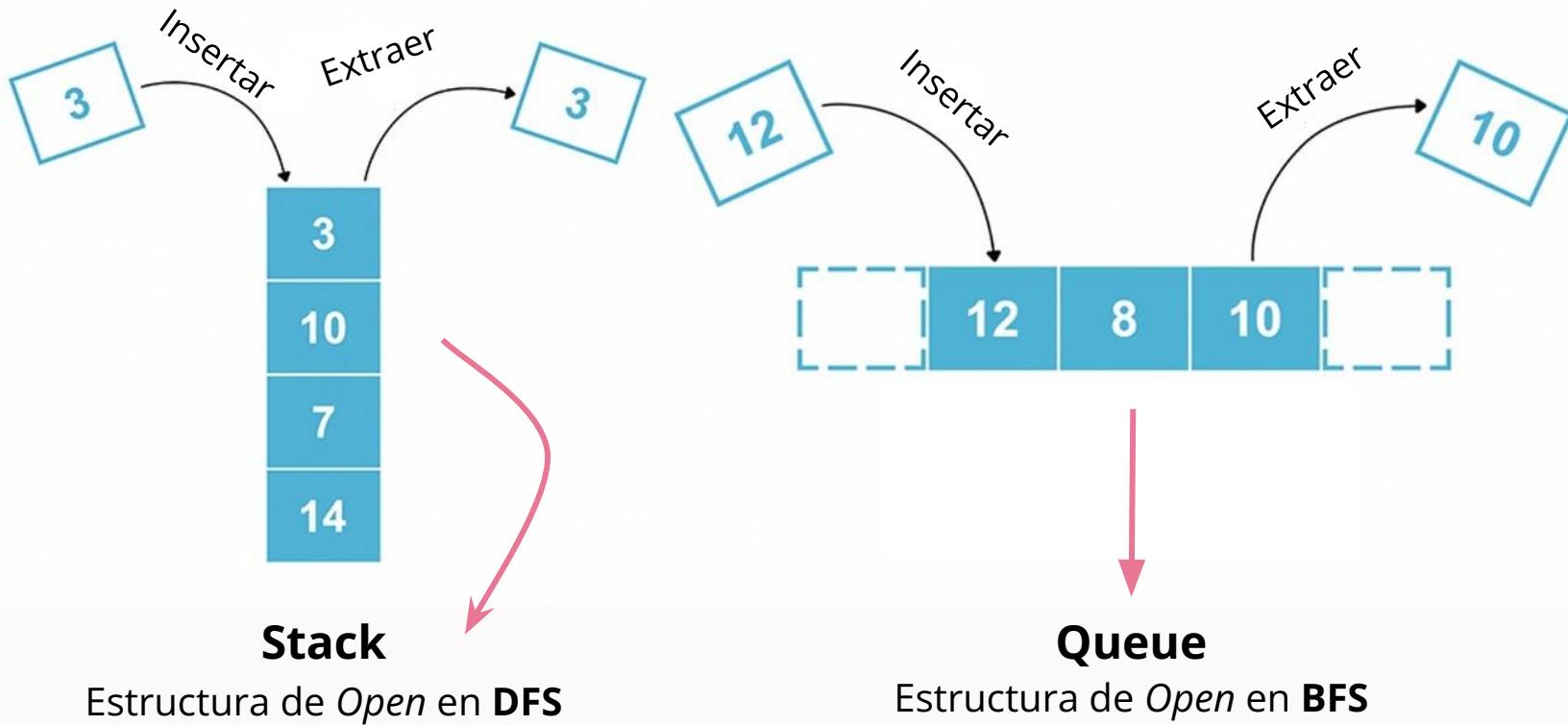


Quiz en Menti! Pregunta 1

¿Qué estructura de datos (stack o queue) utiliza cada algoritmo?



La diferencia entre DFS y BFS



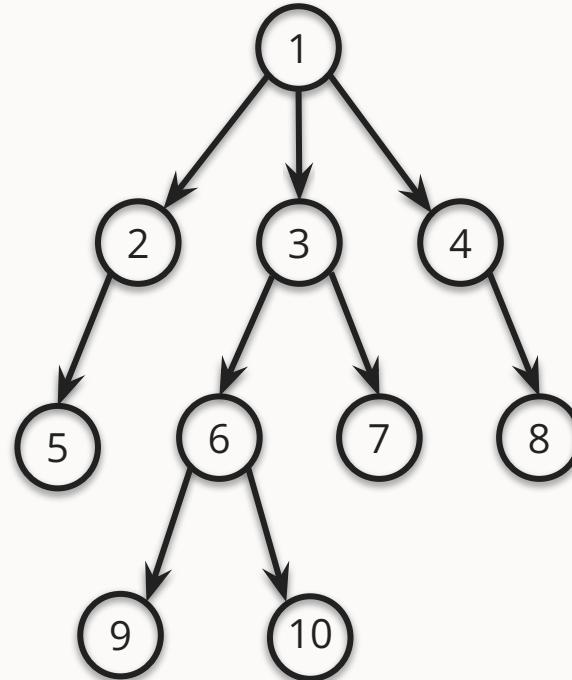


Ejemplo de BFS con algoritmo

BFS - BREADTH FIRST SEARCH
Búsqueda en amplitud

Quiz en Menti! Pregunta 2

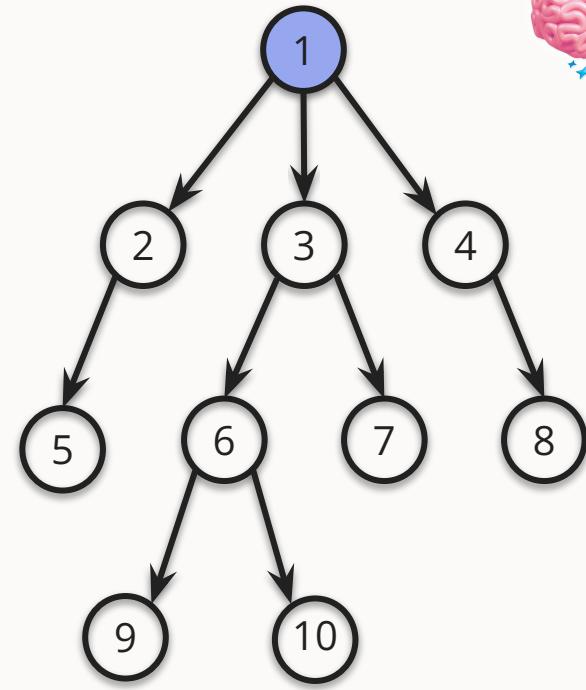
Al usar BFS sobre el árbol de la diapositiva, ¿cómo lucen open y closed luego de llegar al nodo objetivo?



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*

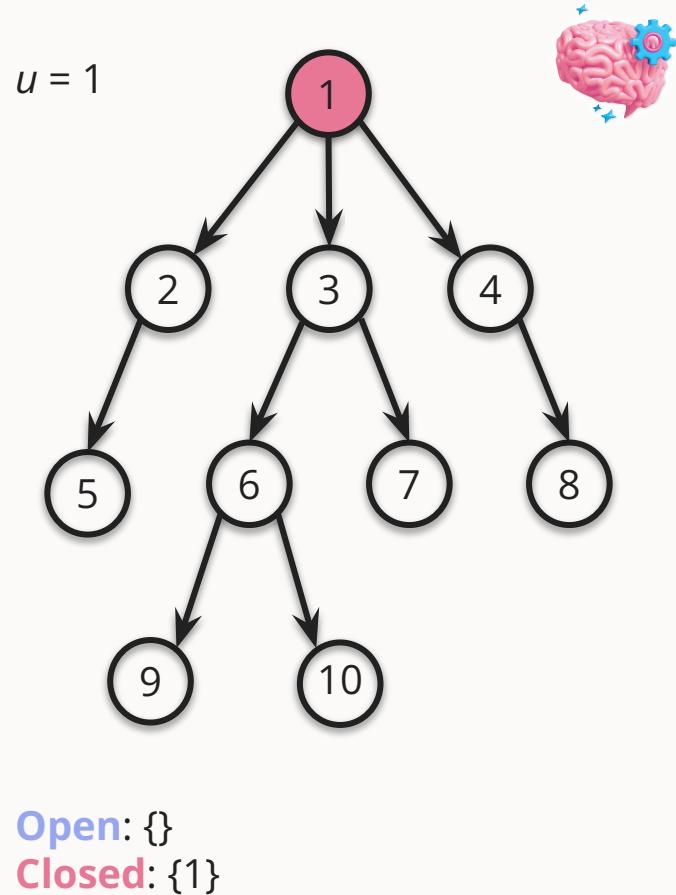


Open: {1}
Closed: {}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

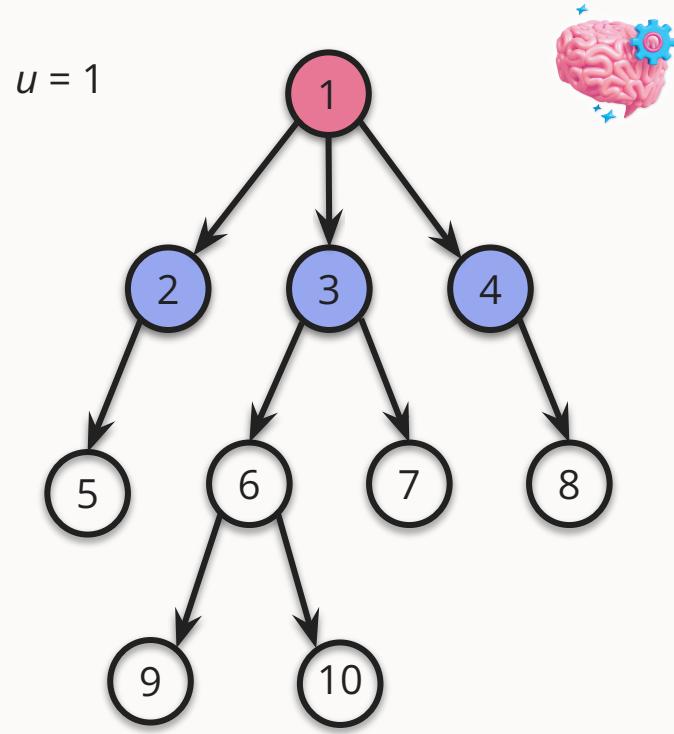
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

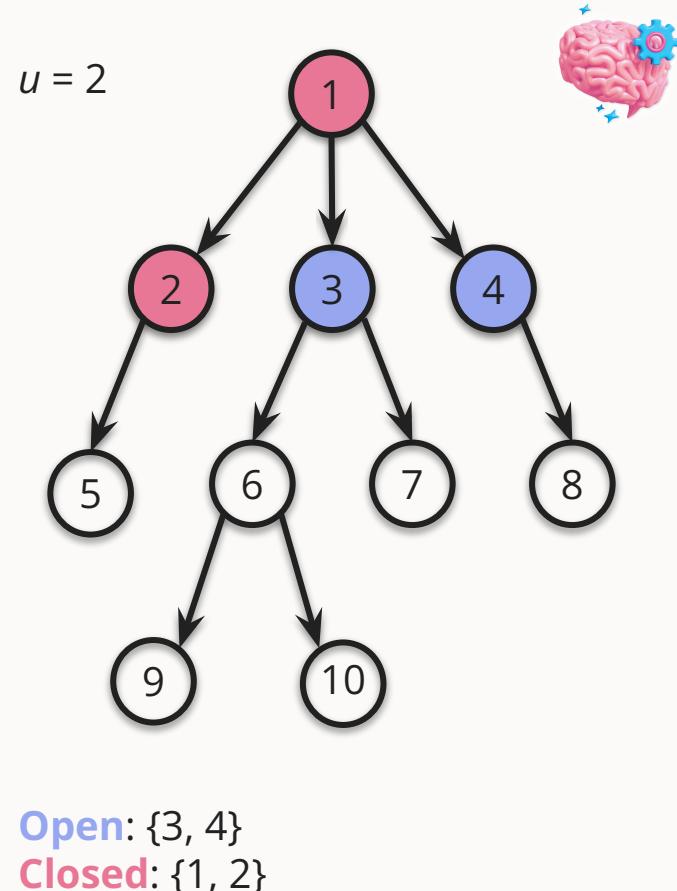
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

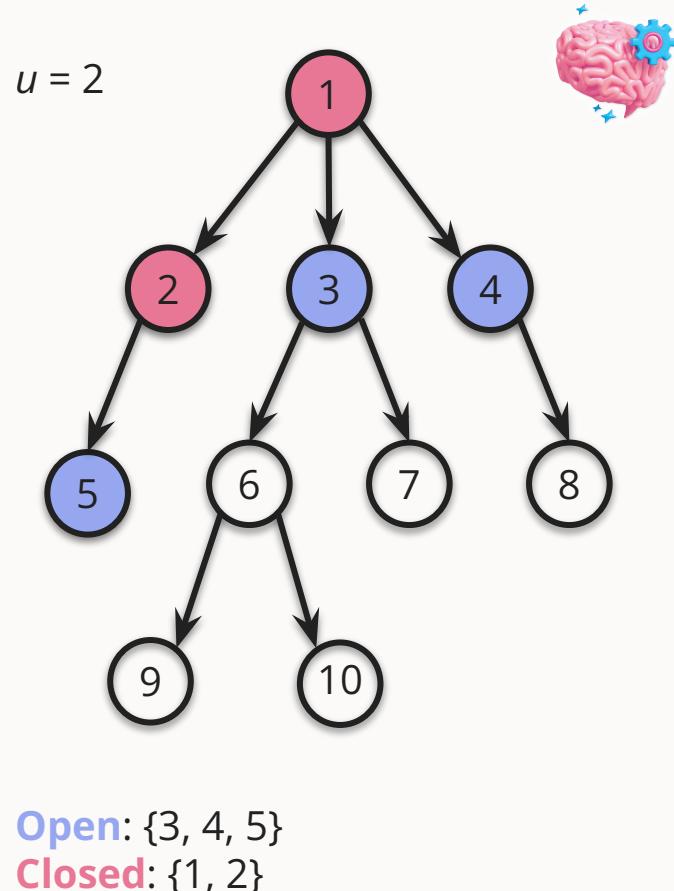
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

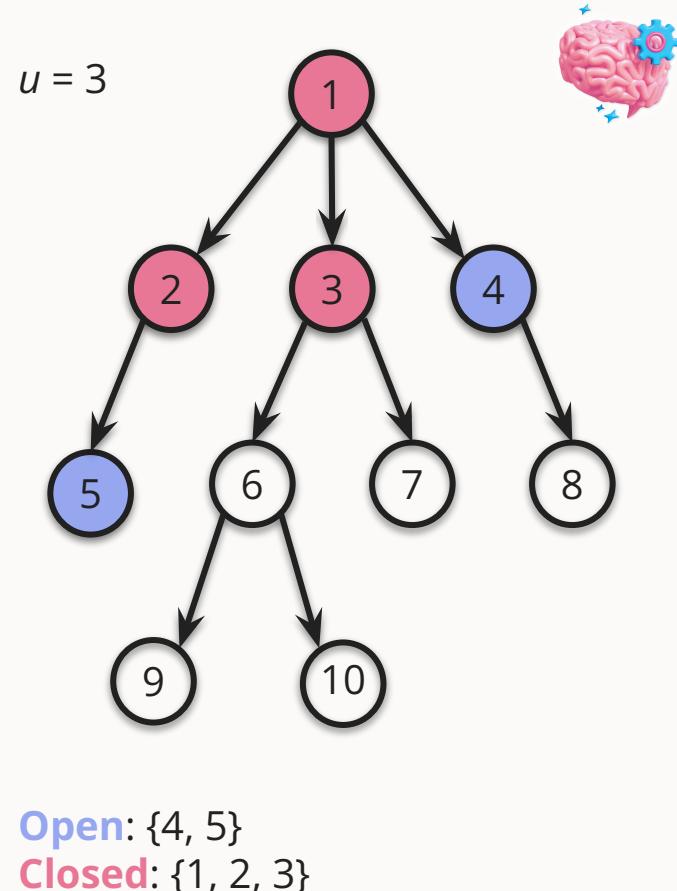
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

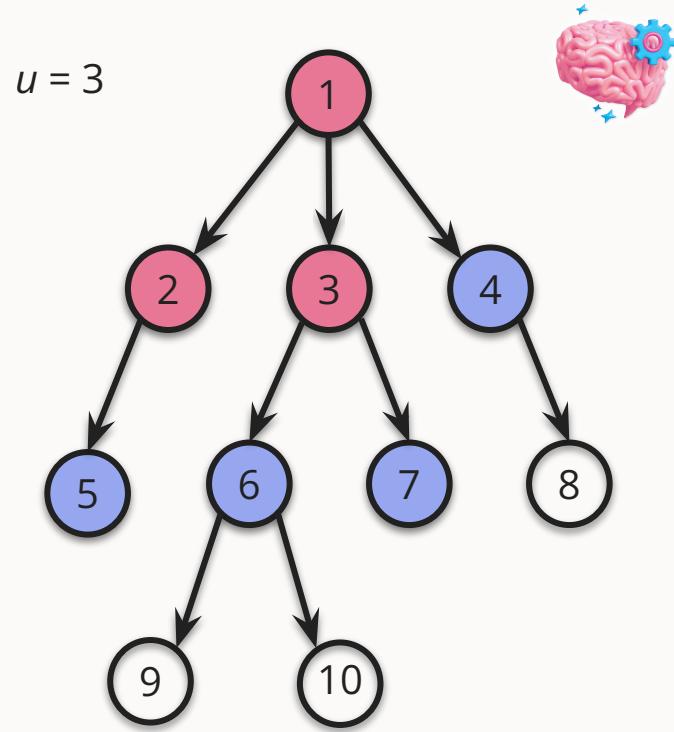
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



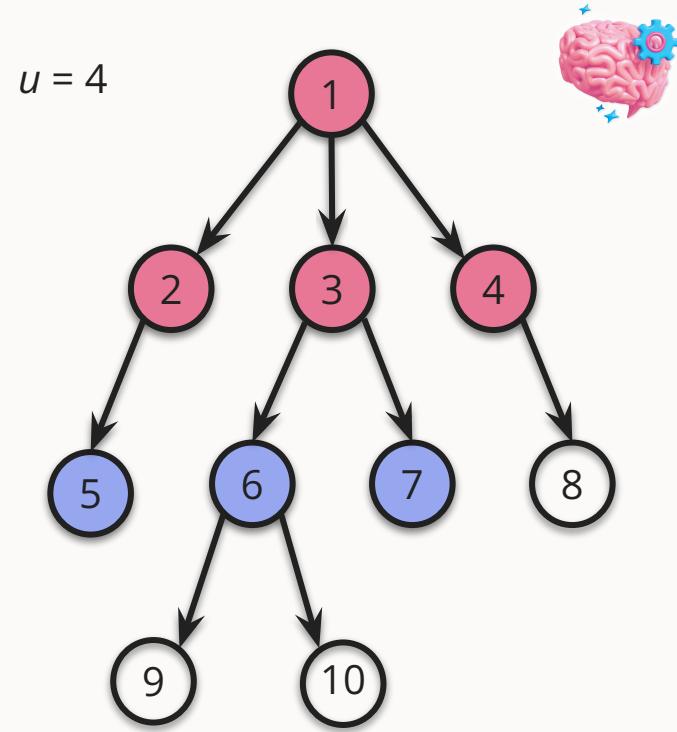
Open: {4, 5, 6, 7}

Closed: {1, 2, 3}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

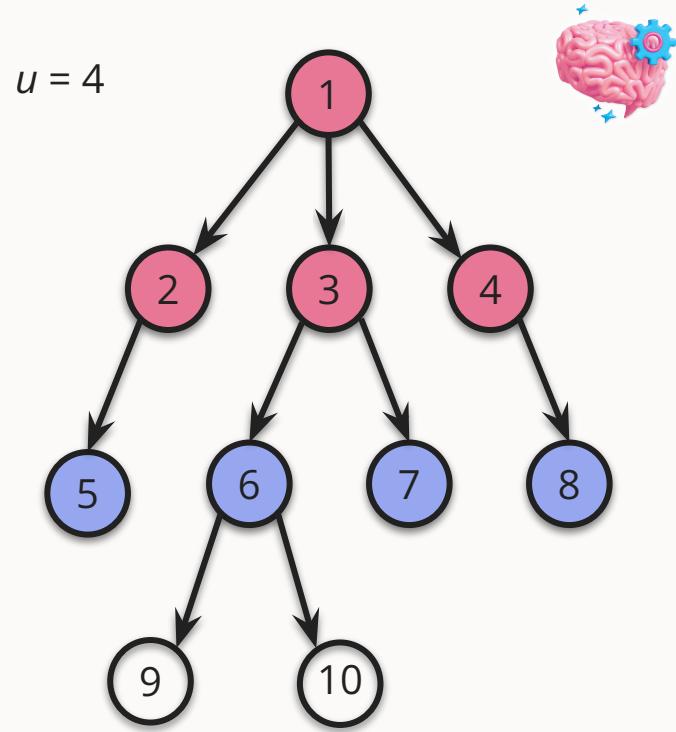
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



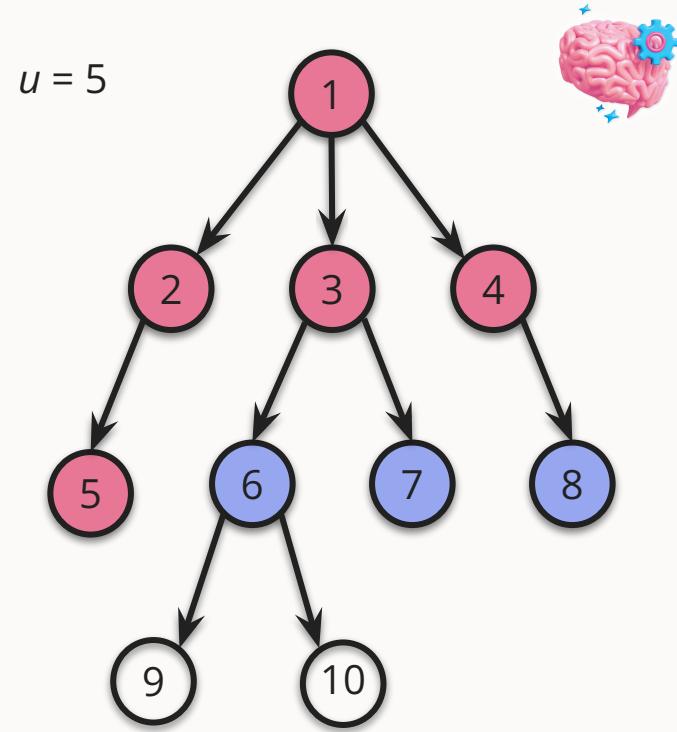
Open: {5, 6, 7, 8}

Closed: {1, 2, 3, 4}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

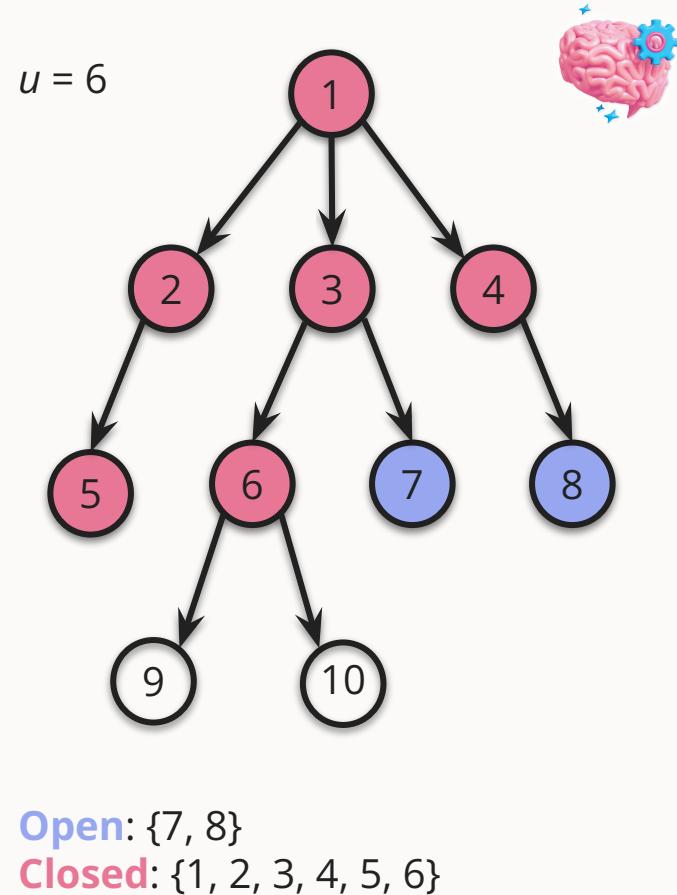
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

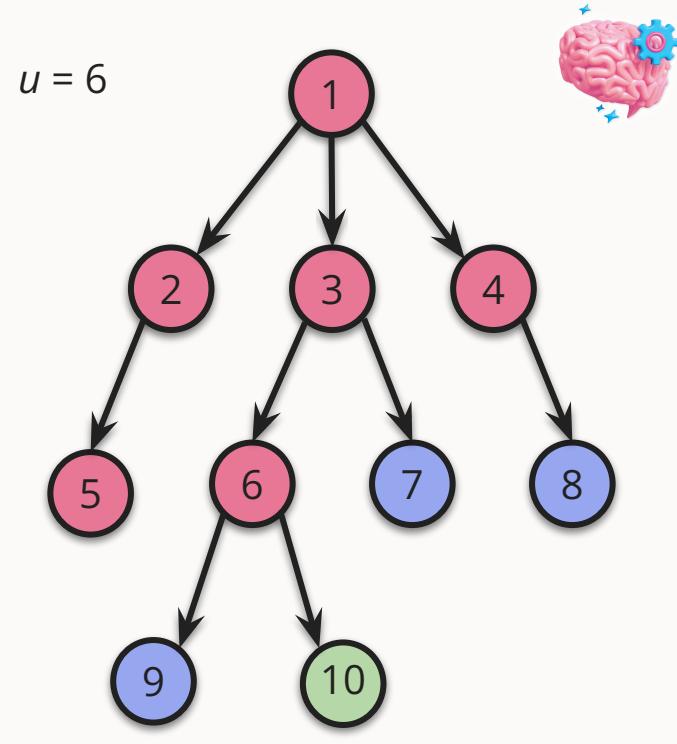
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Open: {7, 8, 9}

Closed: {1, 2, 3, 4, 5, 6}



Ejemplo de BFS con algoritmo

BFS - BREADTH FIRST SEARCH

Búsqueda en amplitud

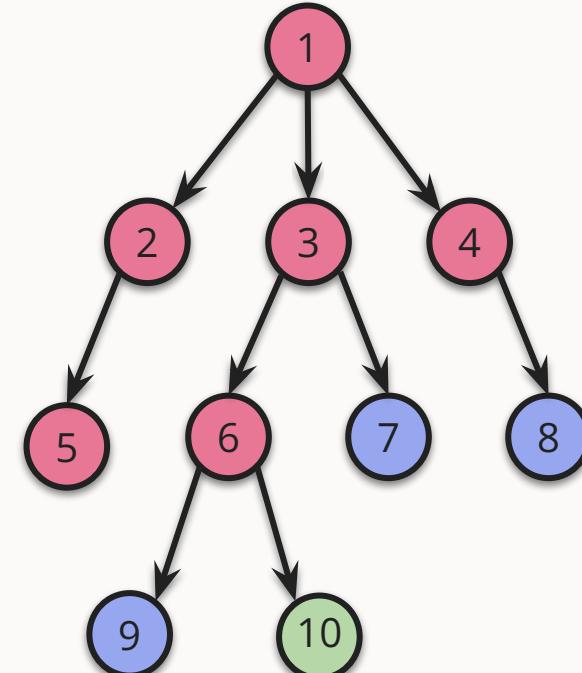
Quiz en Menti! Pregunta 2

Resultado:

Open: {7, 8, 9}

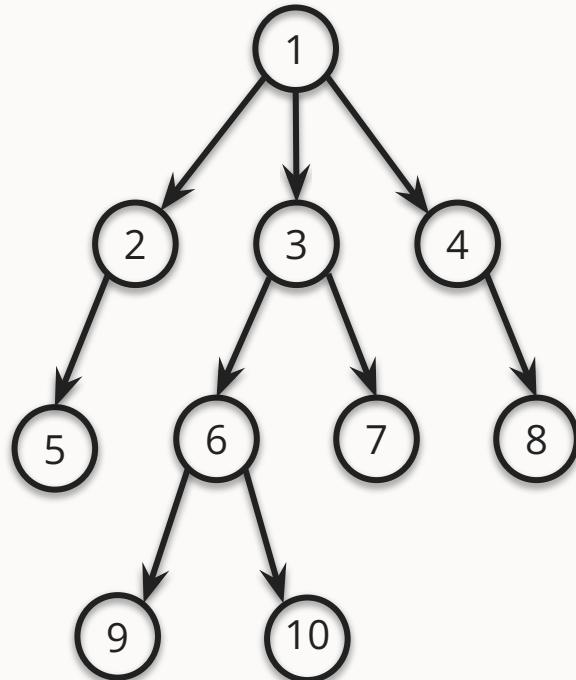
Closed: {1, 2, 3, 4, 5, 6}

*Es importante notar que los nodos se agregaron a Open de izquierda a derecha





Ejemplo de DFS con algoritmo



DFS - DEPTH FIRST SEARCH
Búsqueda en profundidad

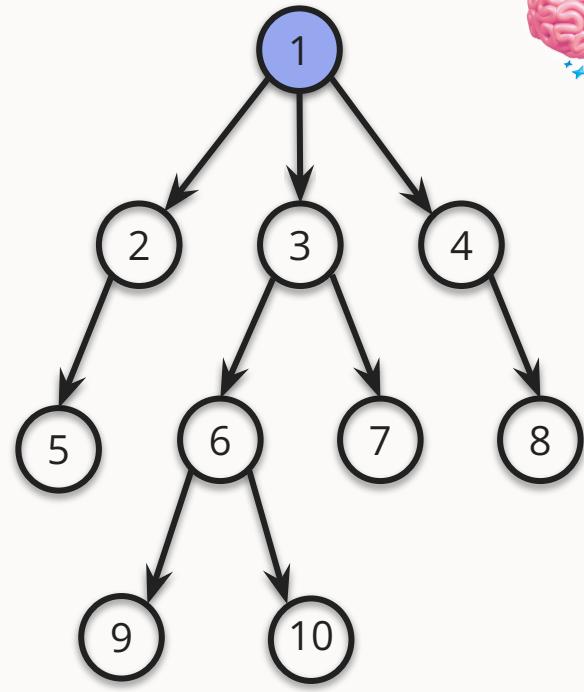
Quiz en Menti! [Pregunta 3](#)

Al usar DFS sobre el árbol de la diapositiva, ¿cómo lucen open y closed luego de llegar al nodo objetivo?

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*

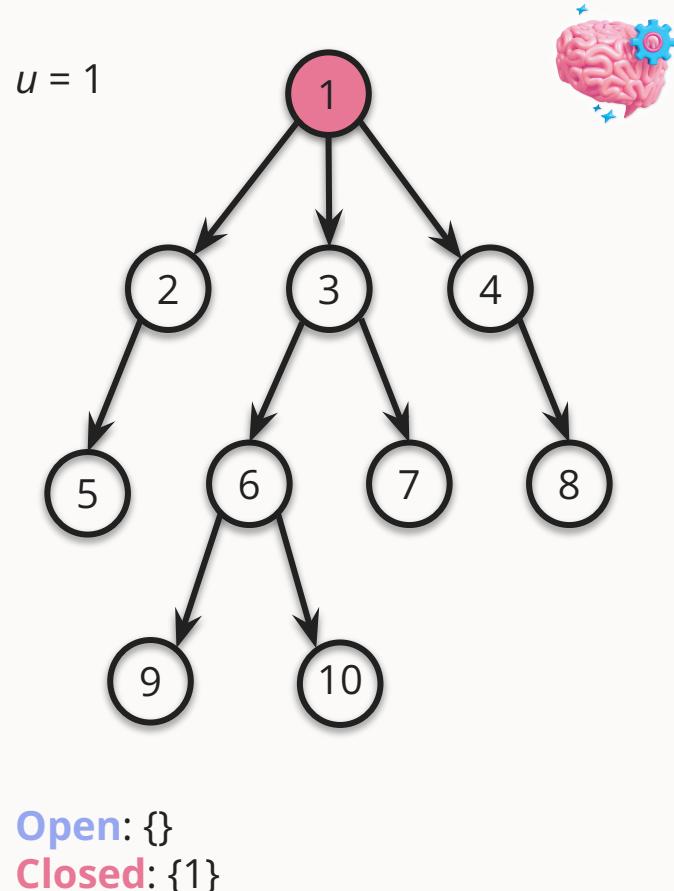


Open: {1}
Closed: {}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

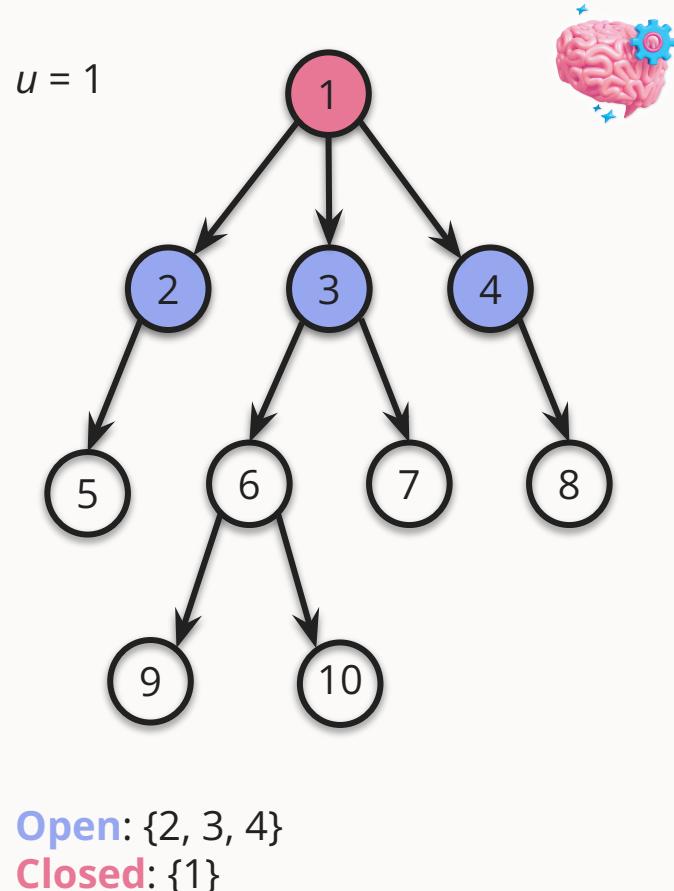
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

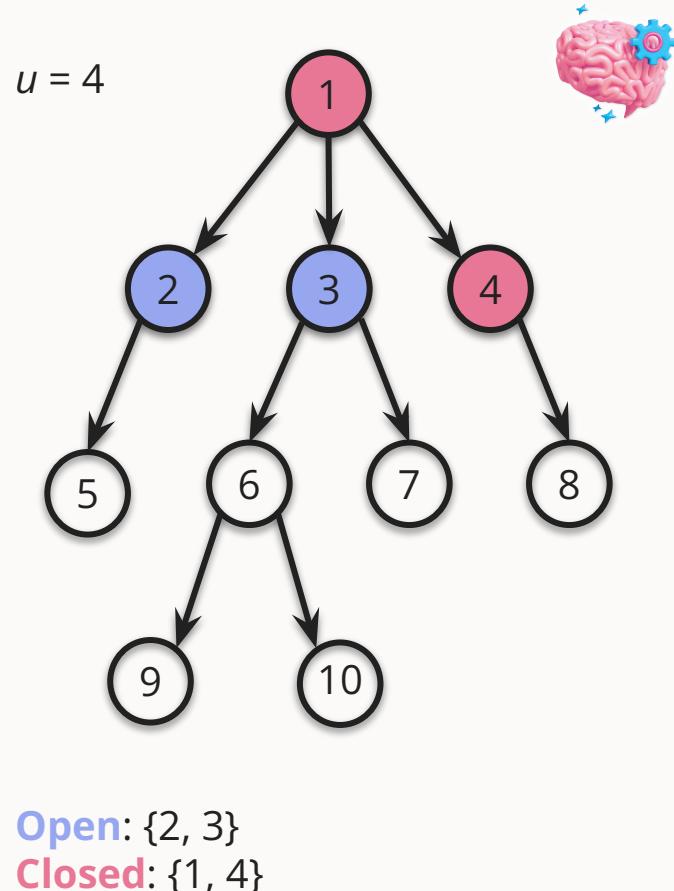
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

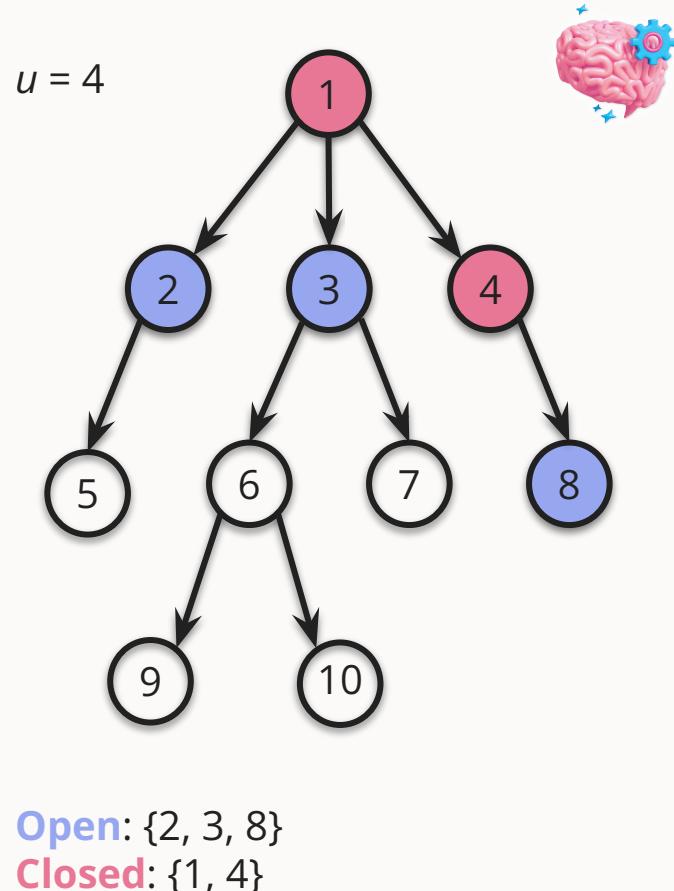
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

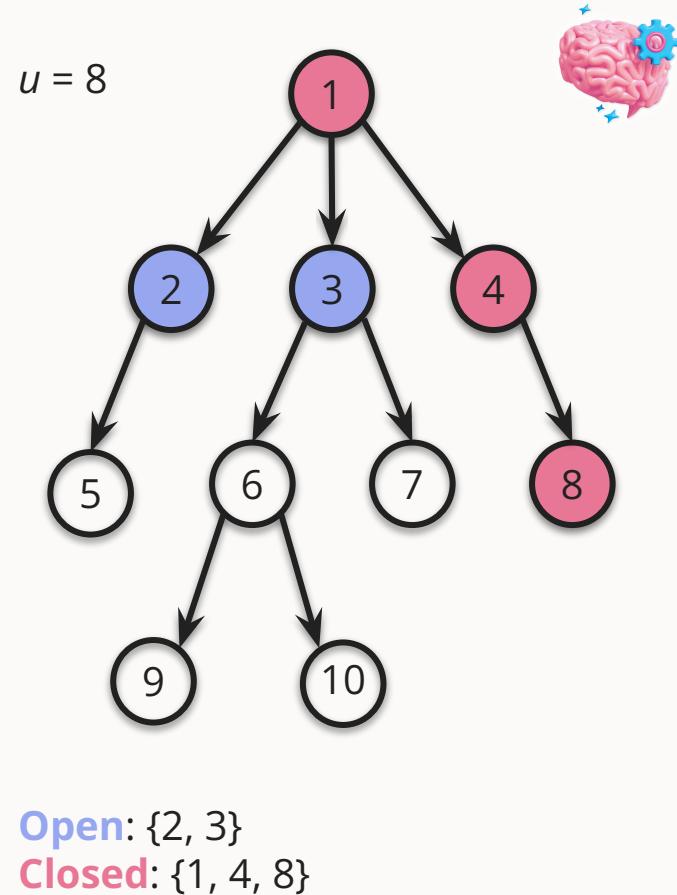
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

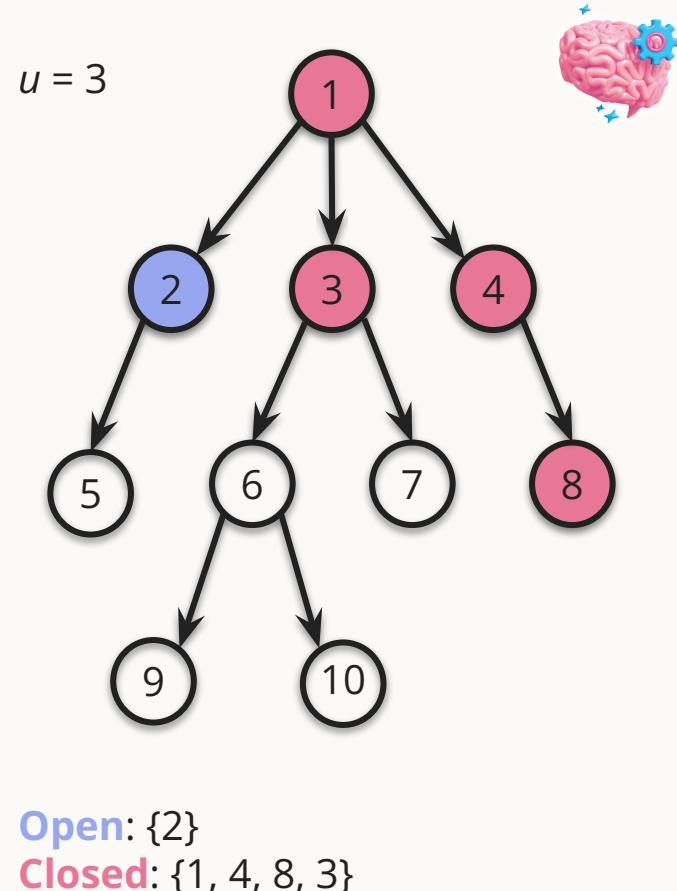
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

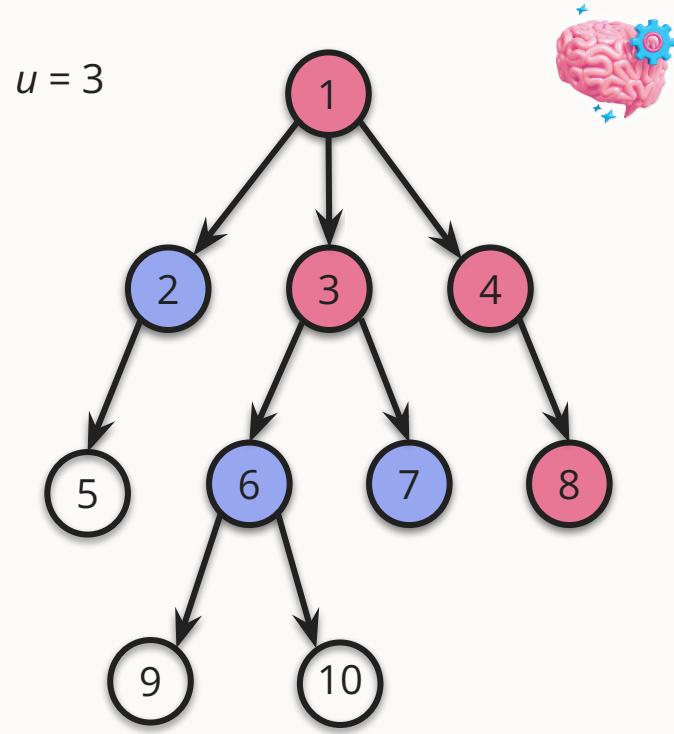
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

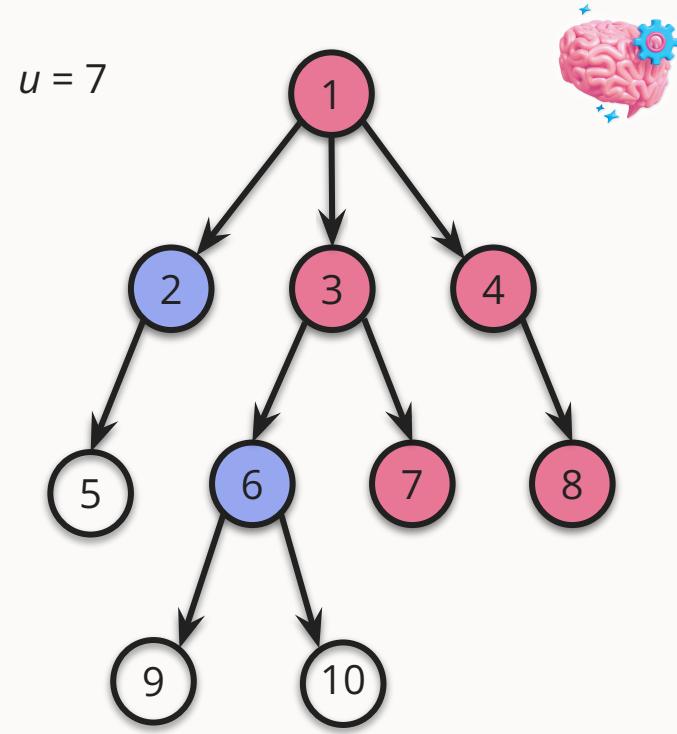
- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



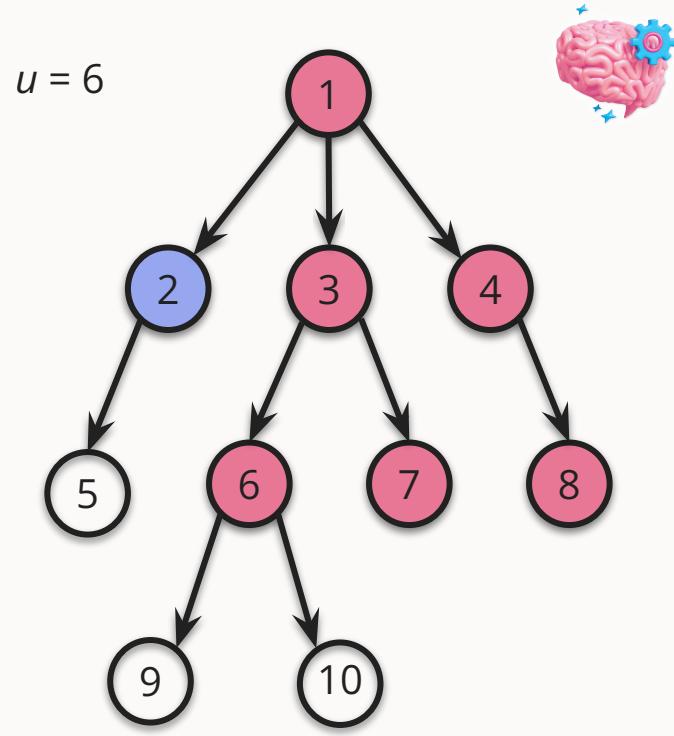
Open: {2, 6}

Closed: {1, 4, 8, 3, 7}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*



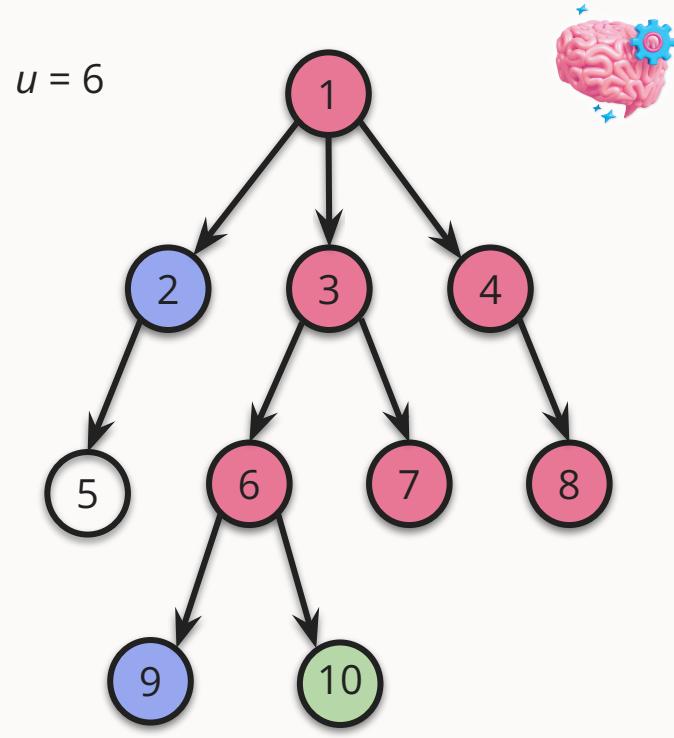
Open: {2}

Closed: {1, 4, 8, 3, 7, 6}

Input: El árbol de la figura

Output: Queremos retornar el nodo 10

- 1 *Open* es un contenedor vacío
- 2 *Closed* es un conjunto vacío
- 3 Inserta s_{init} a *Open*
- 4 $\text{parent}(s_{init}) = \text{null}$
- 5 **while** $\text{Open} \neq \emptyset$
- 6 $u \leftarrow \text{Extraer}(\text{Open})$
- 7 Inserta u en *Closed*
- 8 **for each** $v \in \text{Succ}(u) \setminus (\text{Open} \cup$
Closed)
- 9 $\text{parent}(v) = u$
- 10 **if** $v \in G$ **return** v
- 11 Inserta v a *Open*

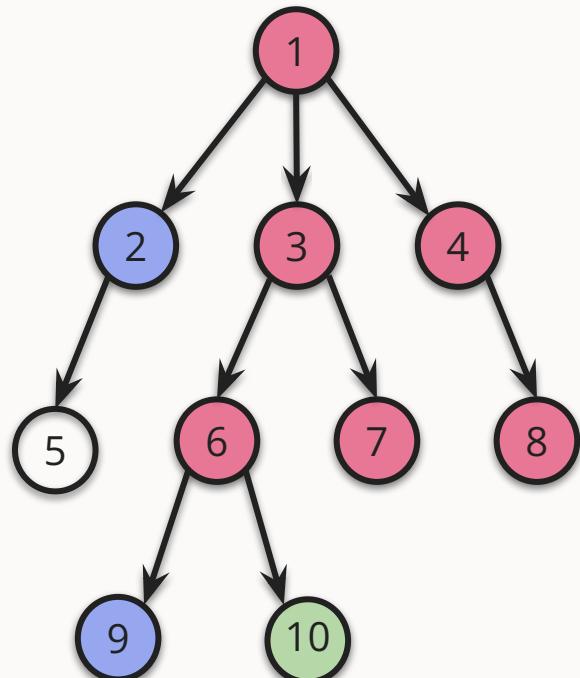


Open: {2, 9}

Closed: {1, 4, 8, 3, 7, 6}



Ejemplo de DFS con algoritmo



DFS - DEPTH FIRST SEARCH
Búsqueda en profundidad

Quiz en Menti! [Pregunta 3](#)

Resultado:

Open: {2, 9}

Closed: {1, 4, 8, 3, 7, 6}

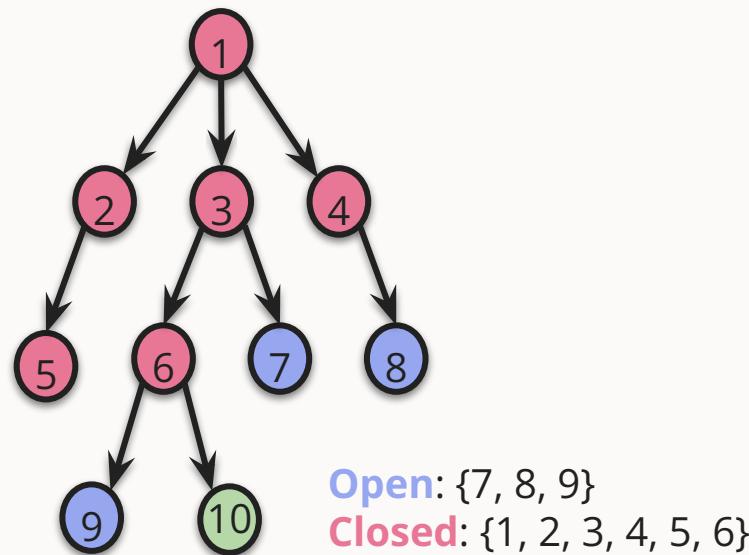
*Es importante notar que los nodos se agregaron a Open de izquierda a derecha



Comparando los algoritmos...

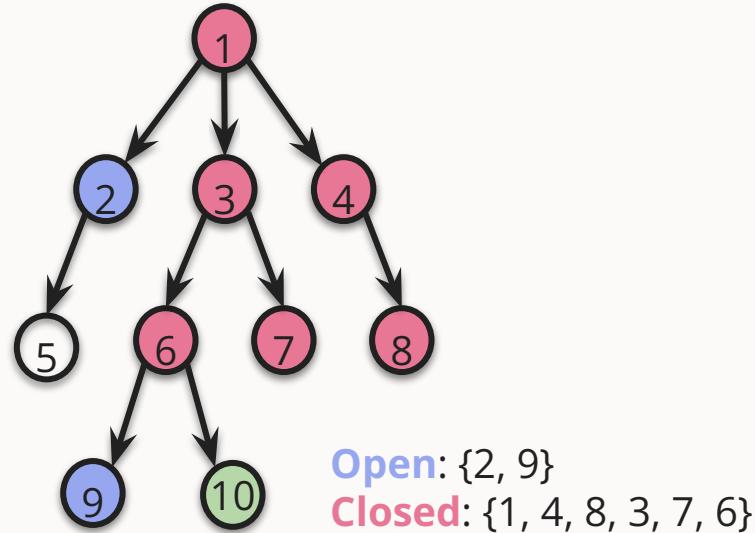
BFS - BREADTH FIRST SEARCH

Búsqueda en amplitud



DFS - DEPTH FIRST SEARCH

Búsqueda en profundidad



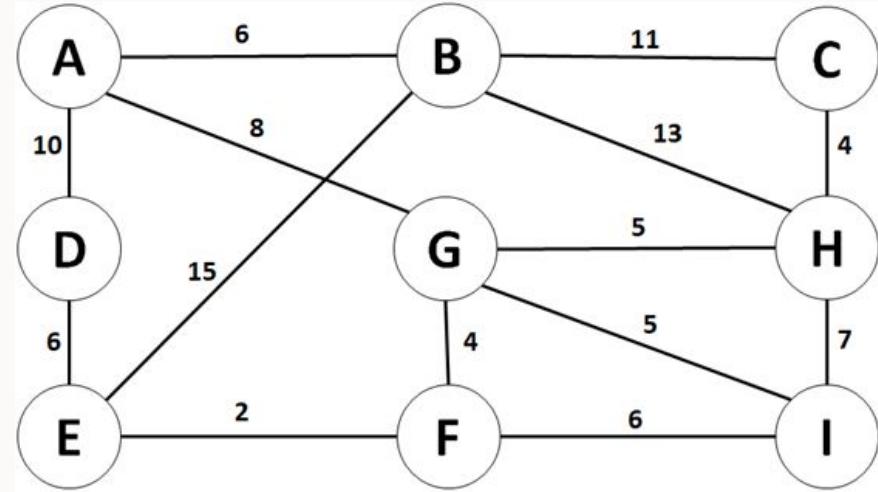


Quiz en Menti! Pregunta 4

¿Cuál es la memoria utilizada por los algoritmos BFS y DFS, respectivamente?



Algoritmo Dijkstra





Algoritmo de Dijkstra

- Es un algoritmo que encuentra **caminos más cortos en un grafo ponderado** (grafos con pesos en las aristas) → encuentra soluciones **óptimas**.
- Comienza por el nodo fuente y explora gradualmente todos los nodos adyacentes, **actualizando las distancias mínimas** a medida que se encuentran caminos más cortos.
- Mantiene una **lista de nodos por visitar** y **selecciona el nodo no visitado con la distancia mínima** para explorar en cada iteración.
- **Usos:** Se puede utilizar en sistemas de transporte para determinar la **ruta más corta entre dos puntos**.



Algoritmo A* y heurísticas

Queremos considerar que los nodos pueden tener una **prioridad**, ahora los nodos se **ordenan** *Open* según un criterio. Esta prioridad considera 2 elementos:

Heurística: Función de **estimación** del costo hasta el objetivo, la denotamos $h(n)$.

Costo incurrido: Función del costo incurrido desde el nodo raíz hasta el actual, la denotamos $g(n)$

Por lo tanto, la función para calcular la prioridad del nodo en *Open* viene dada por:

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

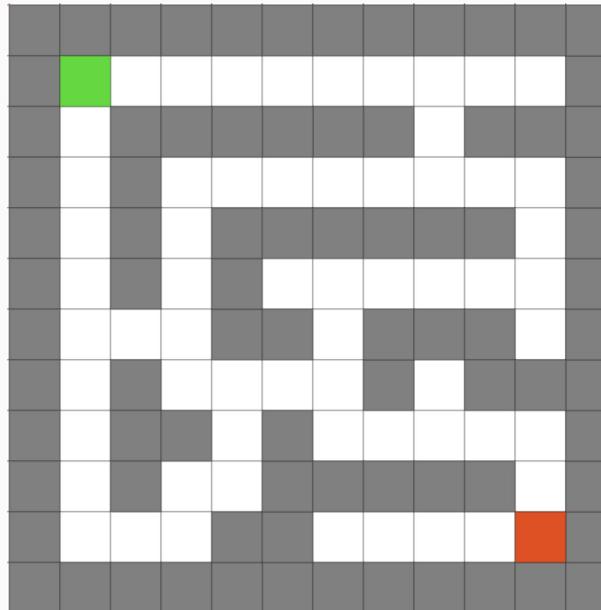
$g(n)$ calcula el costo del pasado, y $h(n)$ *estima* el costo del futuro. Por ende, hay muchas heurísticas posibles para un mismo problema.

Algoritmos de Búsqueda



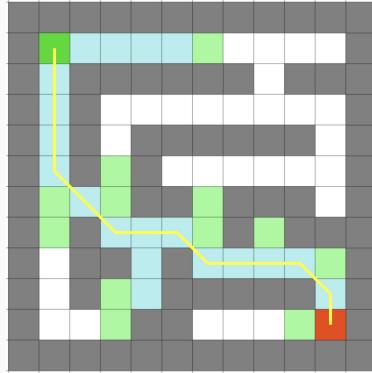
Visualización Dinámica:

PathFinding.js

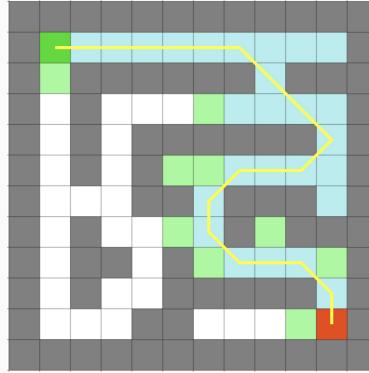


Visualización Dinámica:

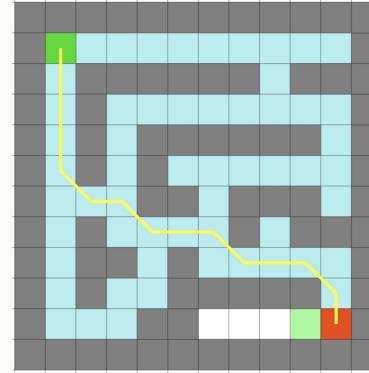
PathFinding.js



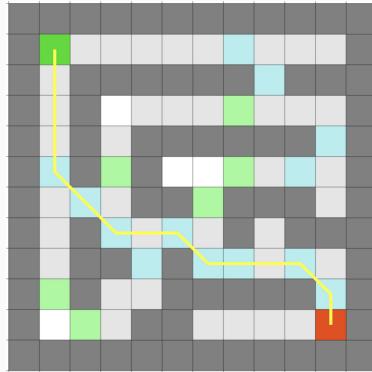
A*, weight = 1



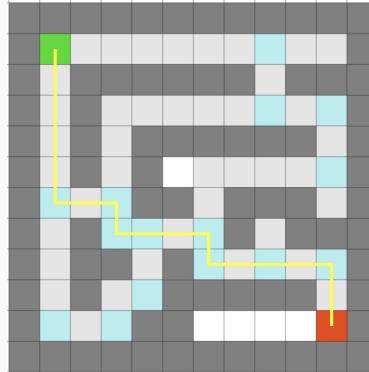
A*, weight = 2



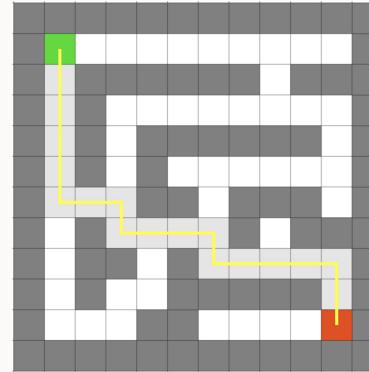
BFS



Jump point search



Orthogonal jps



IDA*





Ayudantía 4

Búsqueda: DFS y BFS

Por Martín Lagies y Willy Pugh

15 de abril 2024