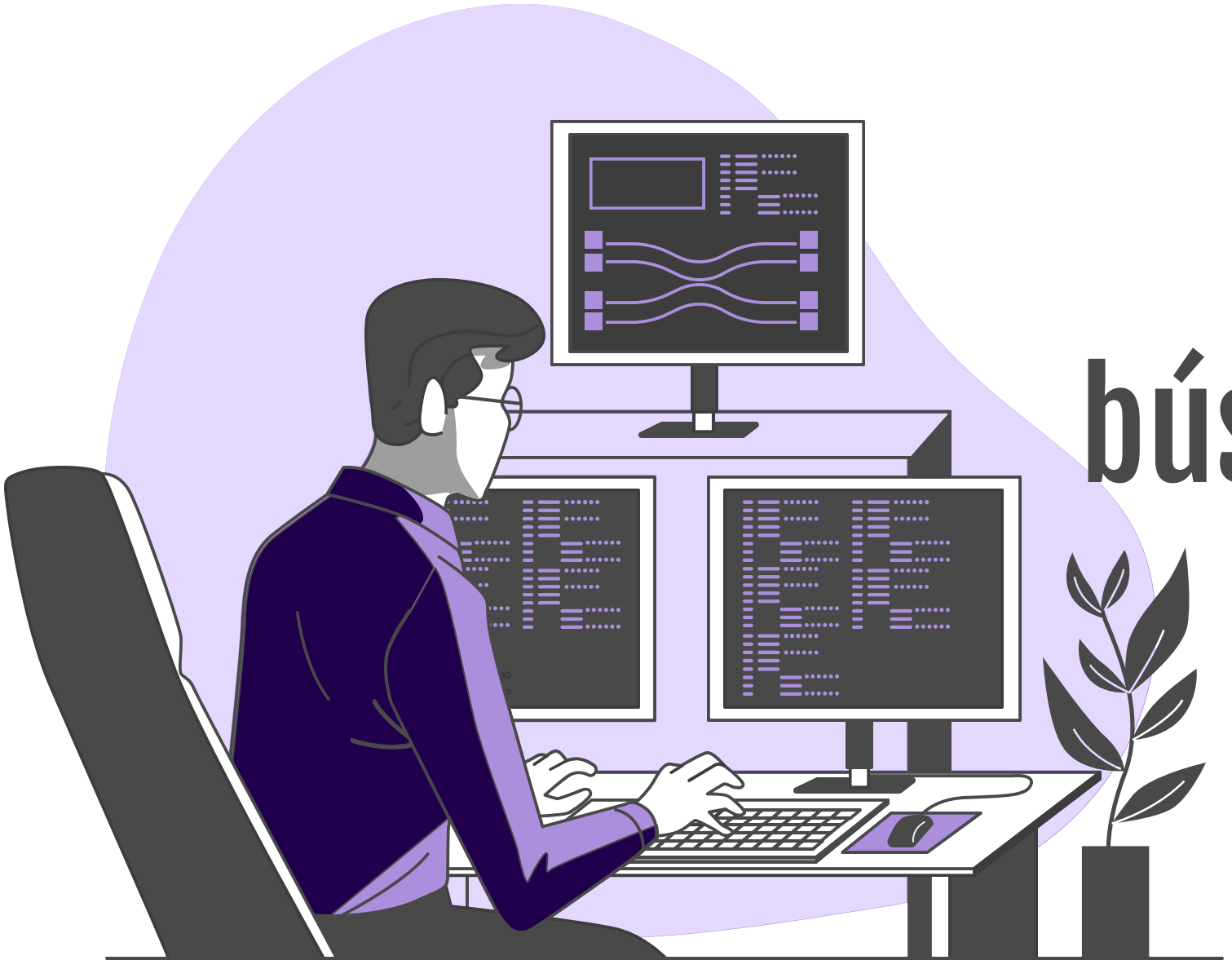
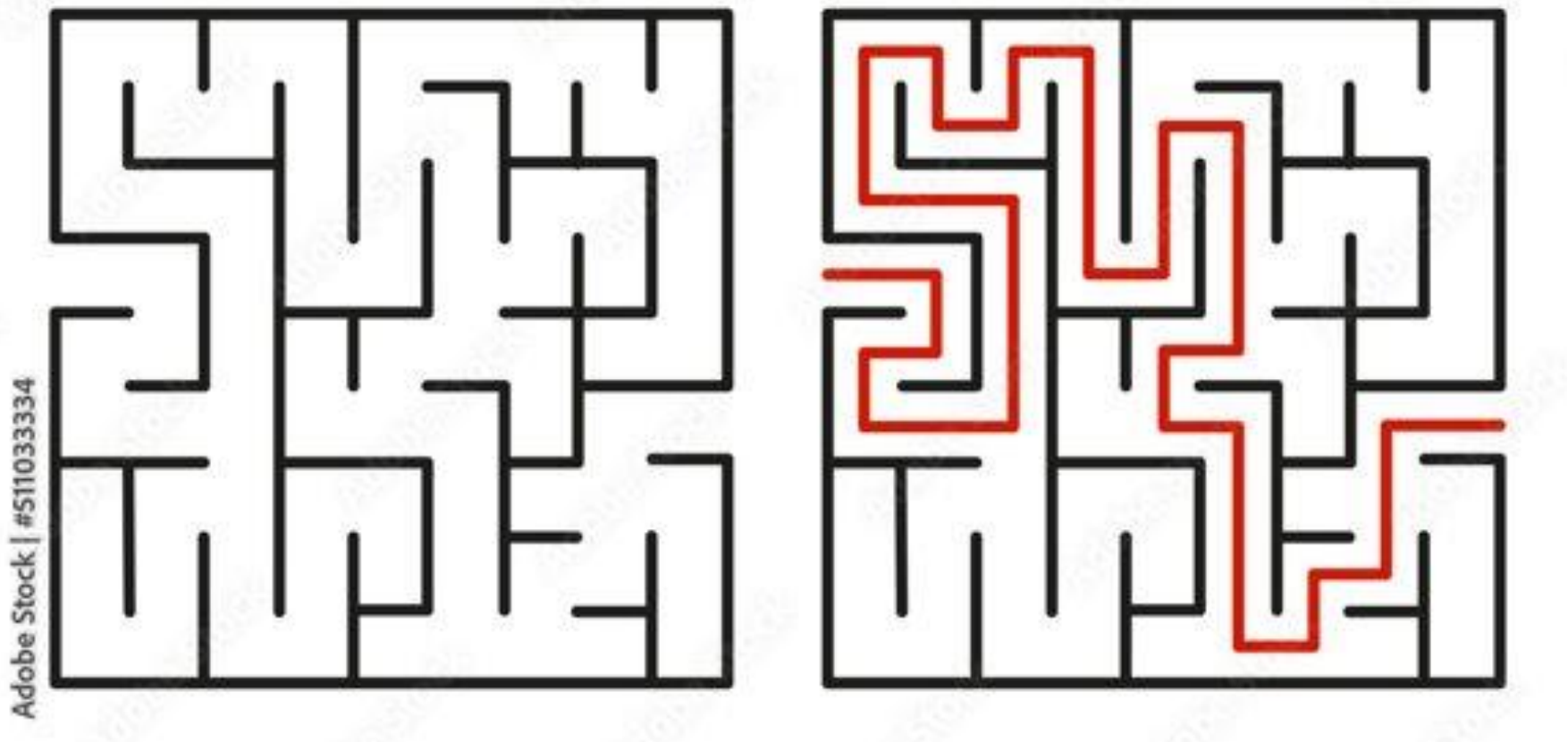


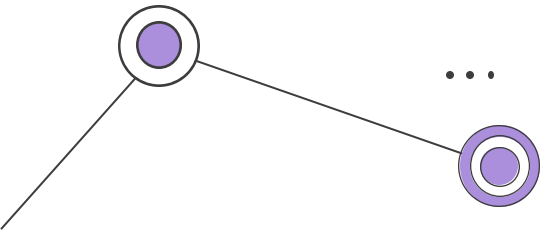
Intro a la búsqueda y A*

Blanca Romero
Daniel Toribio

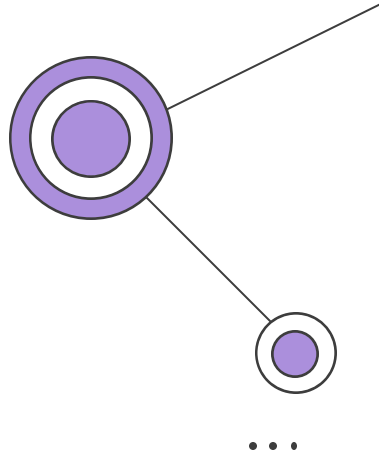
Introducción a la búsqueda

¿Qué es búsqueda? ¿Qué es buscar?





De manera formal



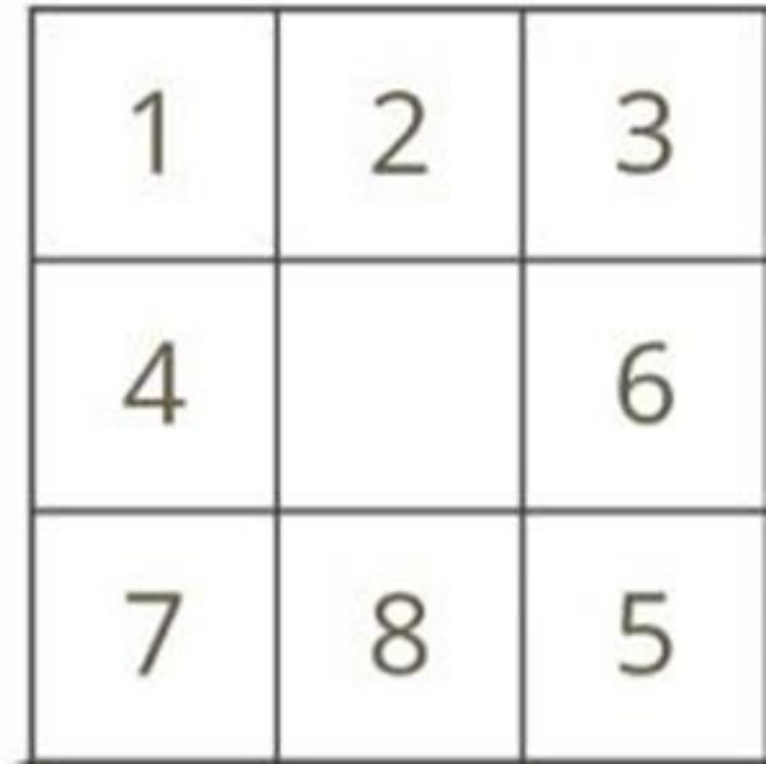
- Estado (s) : Configuración específica de un sistema.
- Acción (a) : Función que hace pasar al sistema de un estado (s) a otro (s').
- Conjunto de acciones (A): Todas las acciones posibles
- Espacio de Búsqueda (S): Conjunto de todos los estados posibles.
- Grafo de Búsqueda: Todos los estados posibles conectan por las acciones que los unen.

¿Cómo se define un problema de búsqueda?

- G es un subconjunto de S con los estados objetivo.
- S_{init} el estado inicial.
- Un problema de búsqueda en notación es...

$$(\mathbf{S}, A, s_{init}, G)$$

Caso típico: Puzzle de 8



Formalizamos el problema de búsqueda (S, A, S_init, G)

- Problema de búsqueda (S, A, s_{init}, G)
  - $\mathcal{S}$ = conjunto de estados
  - $\mathcal{A}$ = conjunto de acciones
  - s_{init} = estado inicial
  - G = conjunto de estados finales

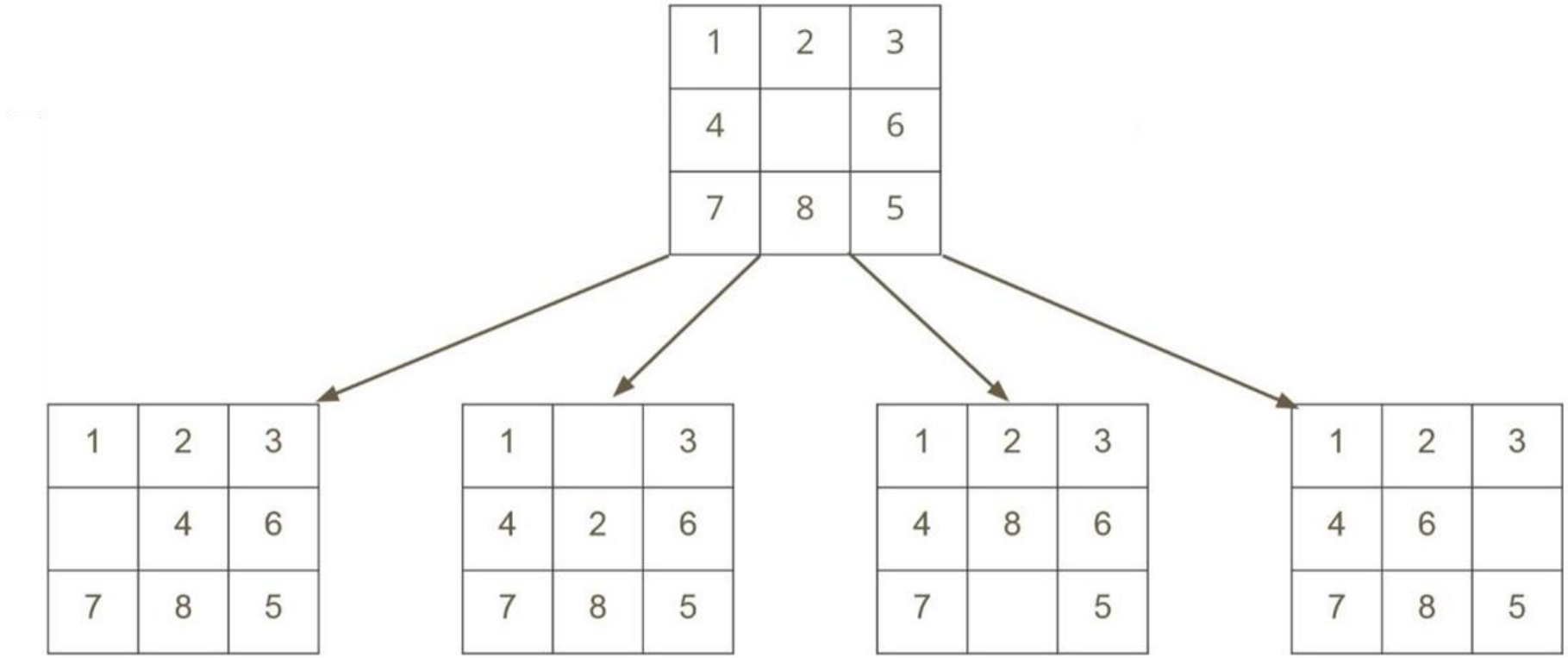
1	2	3
4		6
7	8	5

Estado inicial

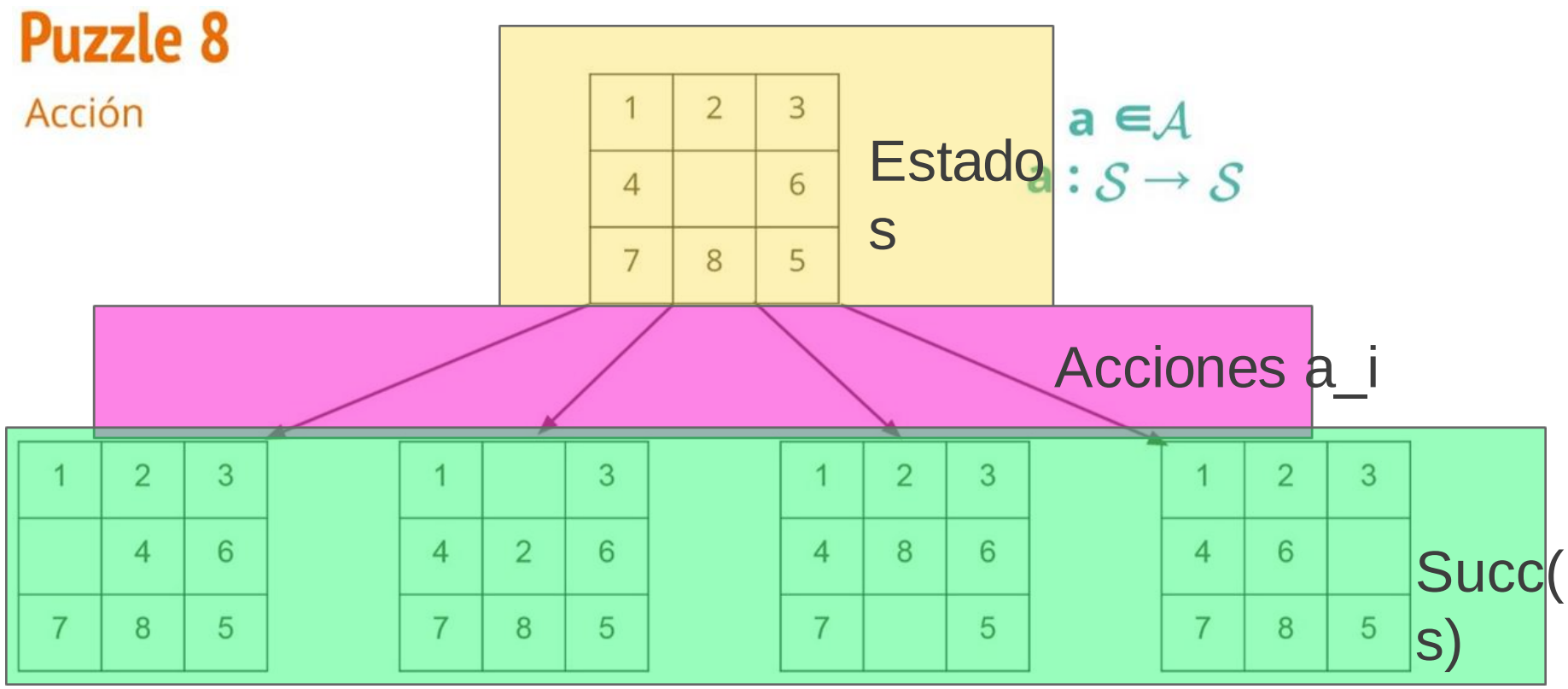
	1	2
3	4	5
6	7	8

Estado final $\in$ G

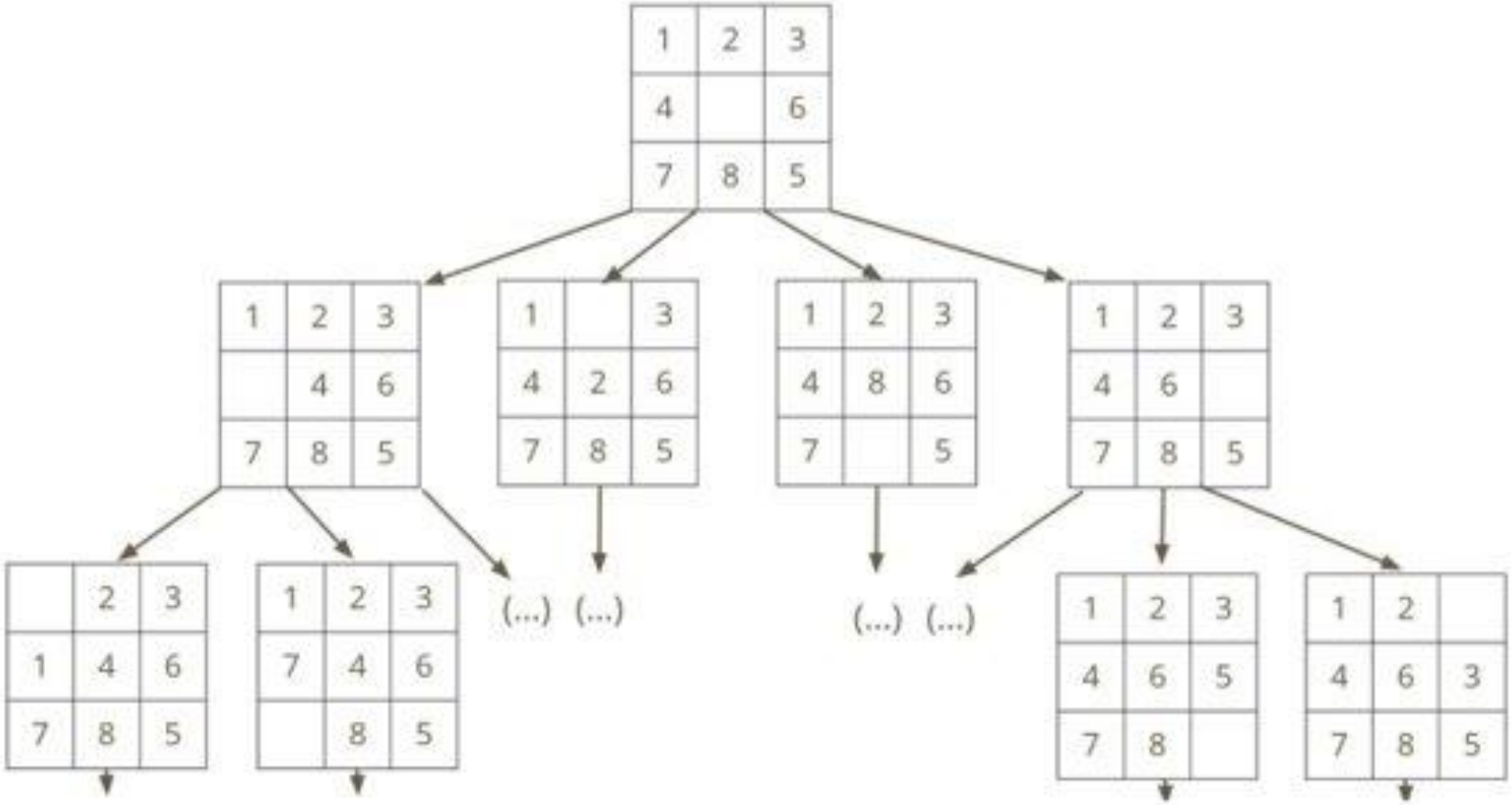
¿Cuáles son acciones? ¿Cuáles son estados?



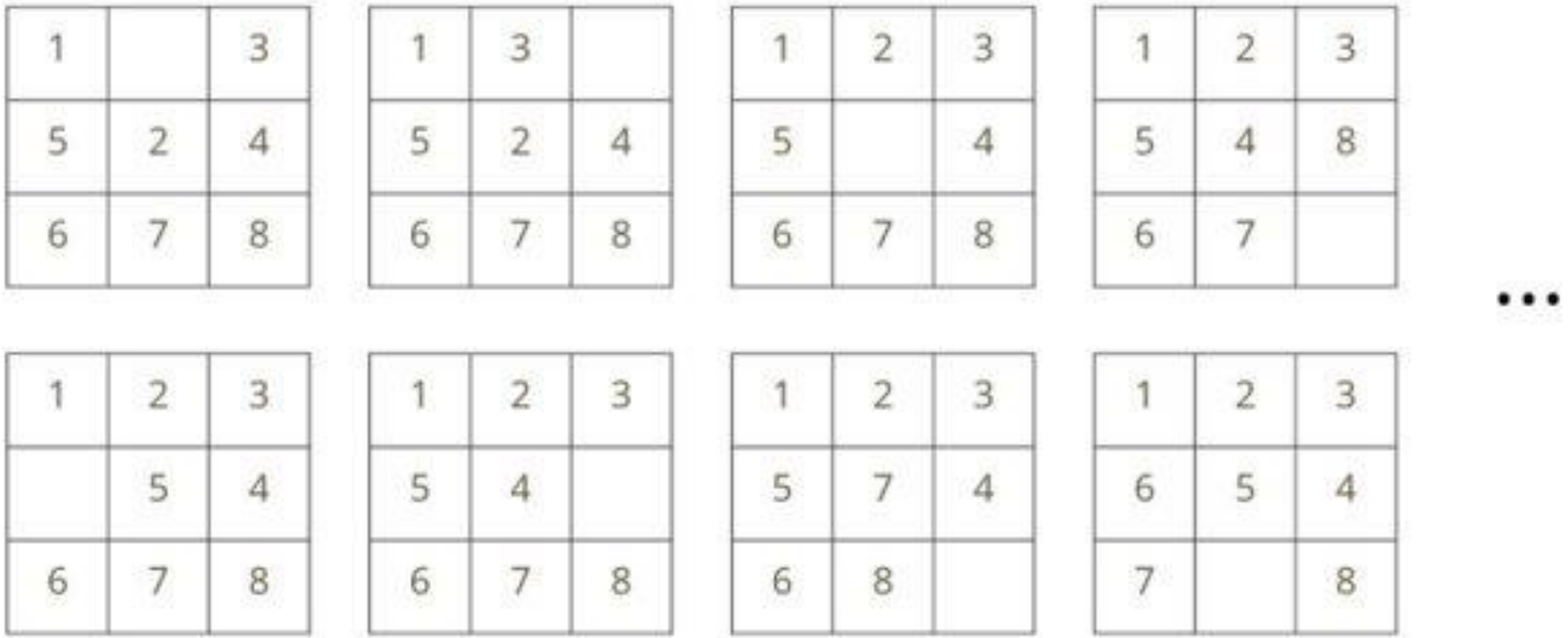
¿Cuáles son acciones? ¿Cuáles son estados?



Grafo de búsqueda



Espacio de Búsqueda



¿Cómo resolvemos un problema de búsqueda?

Memoria
usada O(b^p)

Memoria
usada O(bm)

Tiempo y espacio de BFS y DFS

Memoria usada O(b^p)
Requiere tiempo O(b^p)

Memoria usada O(bm)
Requiere tiempo O(b^m)

b : factor de ramificación
p : profundidad a la que se encuentra la solución
m : largo de la rama más larga del árbol de búsqueda

Algoritmo de búsqueda

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

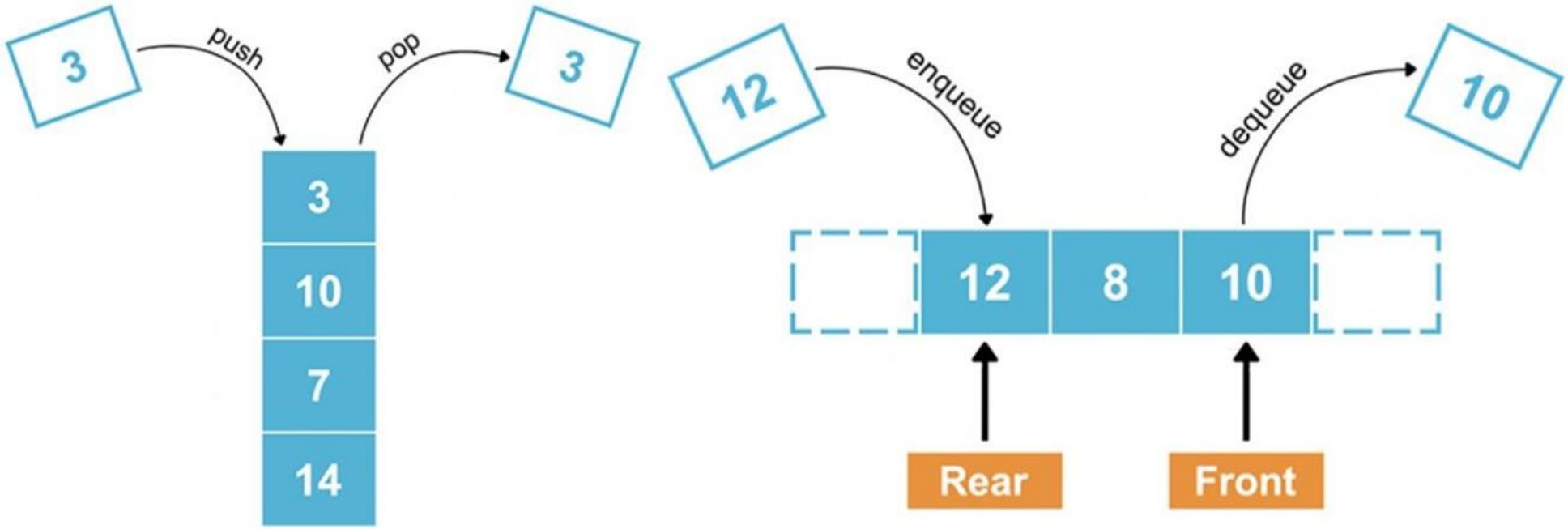
¿Cuál es la diferencia entre DFS y BFS?

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

La estructura de datos que se utiliza para mantener el Open

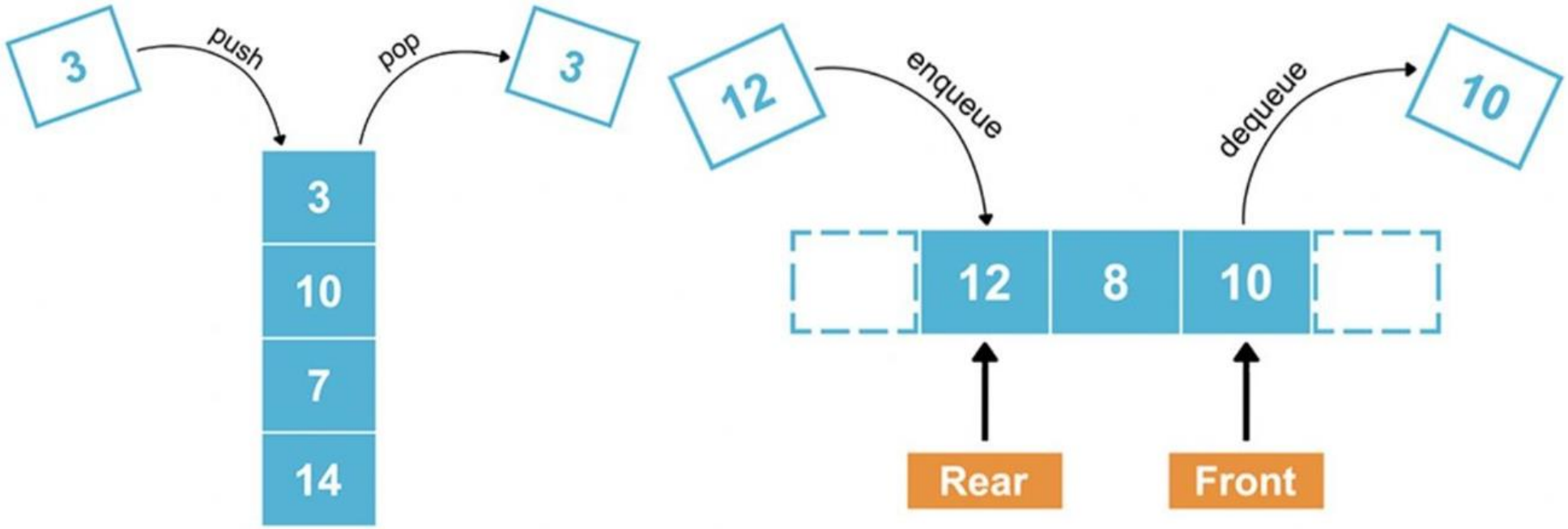
¿Cuál es la diferencia entre DFS y BFS?



STACK

QUEUE

¿Cuál es la diferencia entre DFS y BFS?



STACK en DFS

QUEUE en BFS

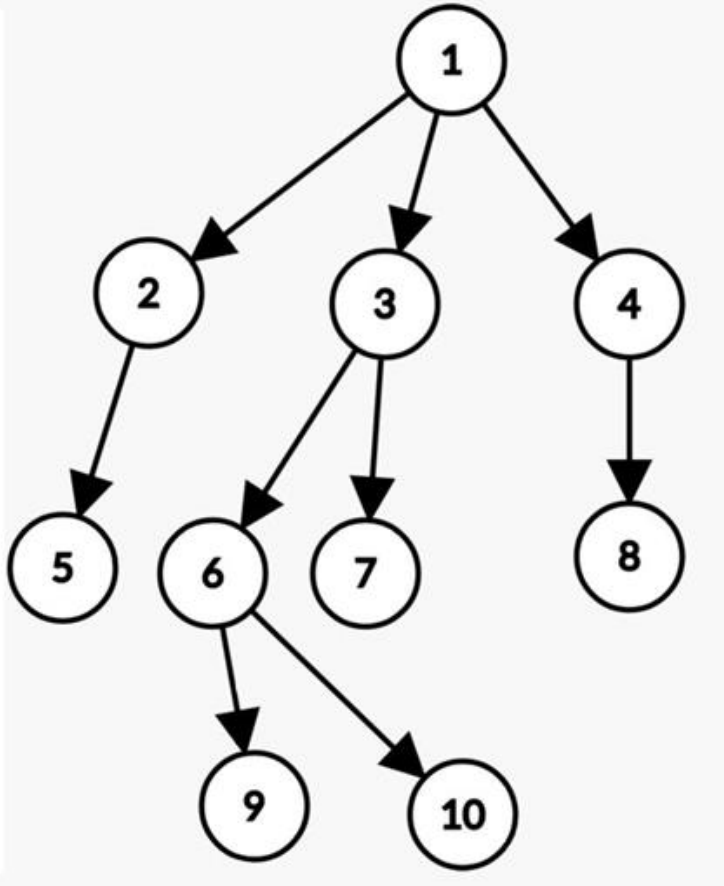
BFS

MENTI!

Al utilizar el pseudo código de BFS (ver imagen) sobre el siguiente grafo y suponiendo que el nodo 10 es el único objetivo, ¿Cómo queda Open y Closed al retornar el algoritmo

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

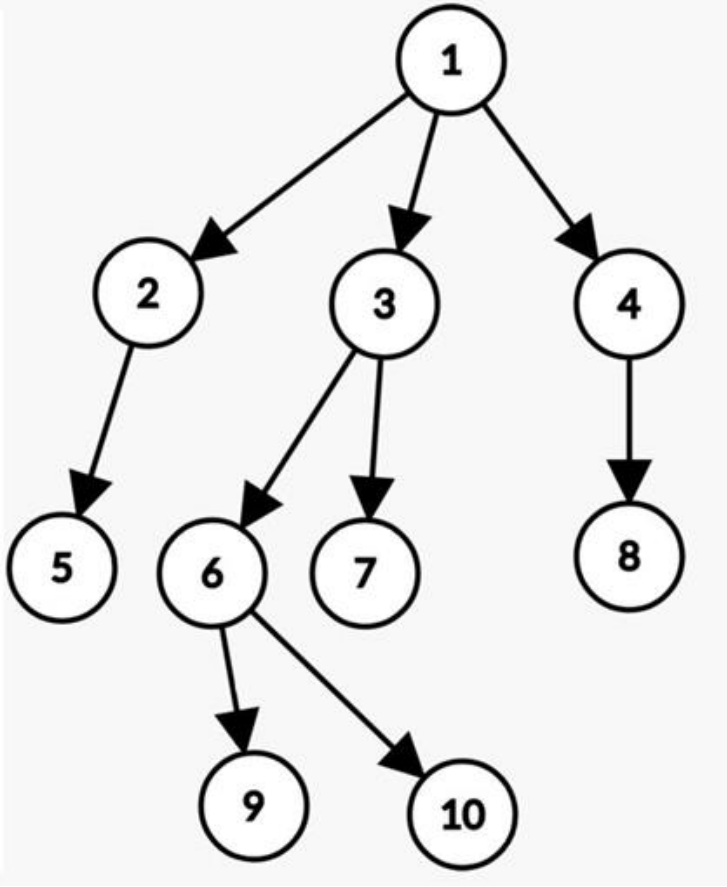
Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*



BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

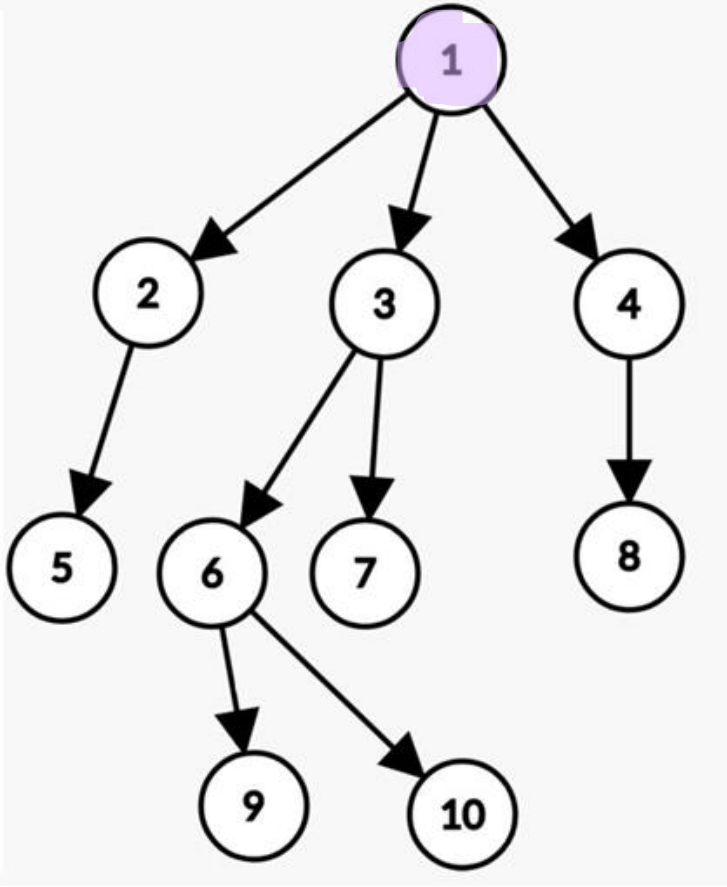



```
Closed:{}
Open:{}
Goal:{10}
```

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

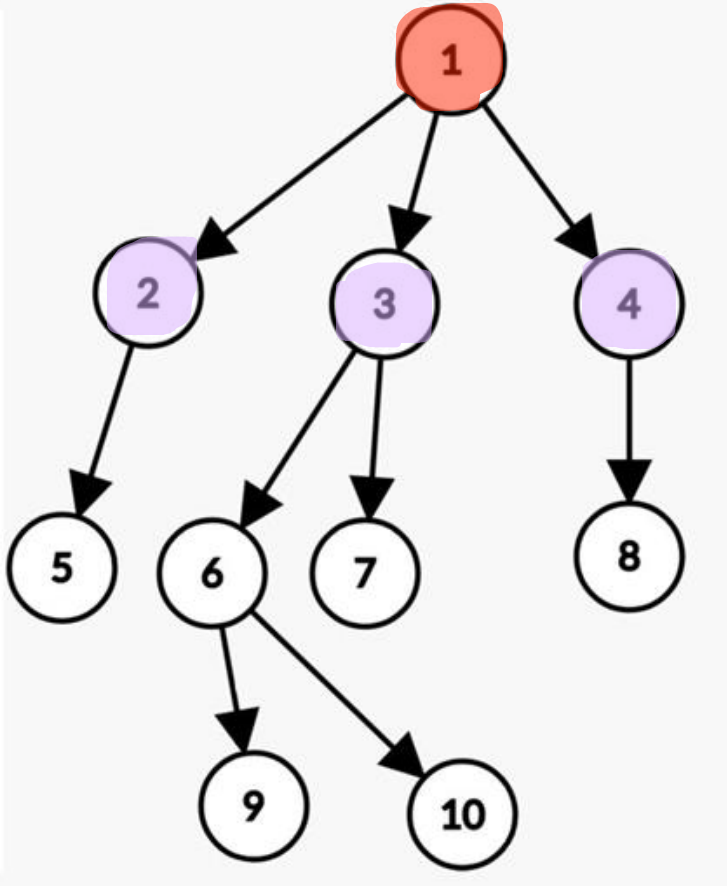



```
Closed:{}
Open:{1}
Goal:{10}
```

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*



Closed:{1}

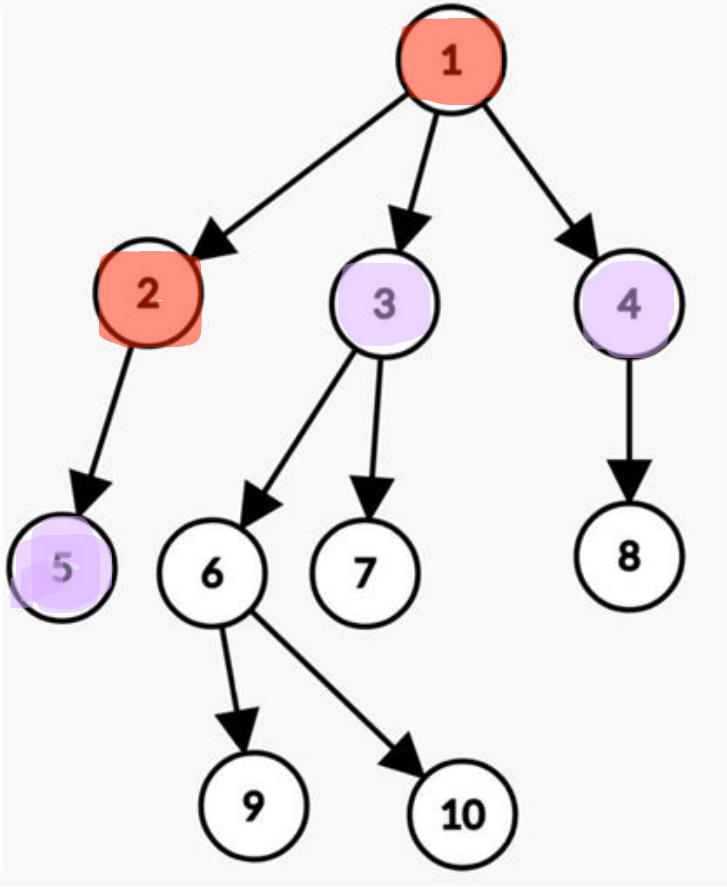
Open:{2,3,4}

Goal:{10}

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

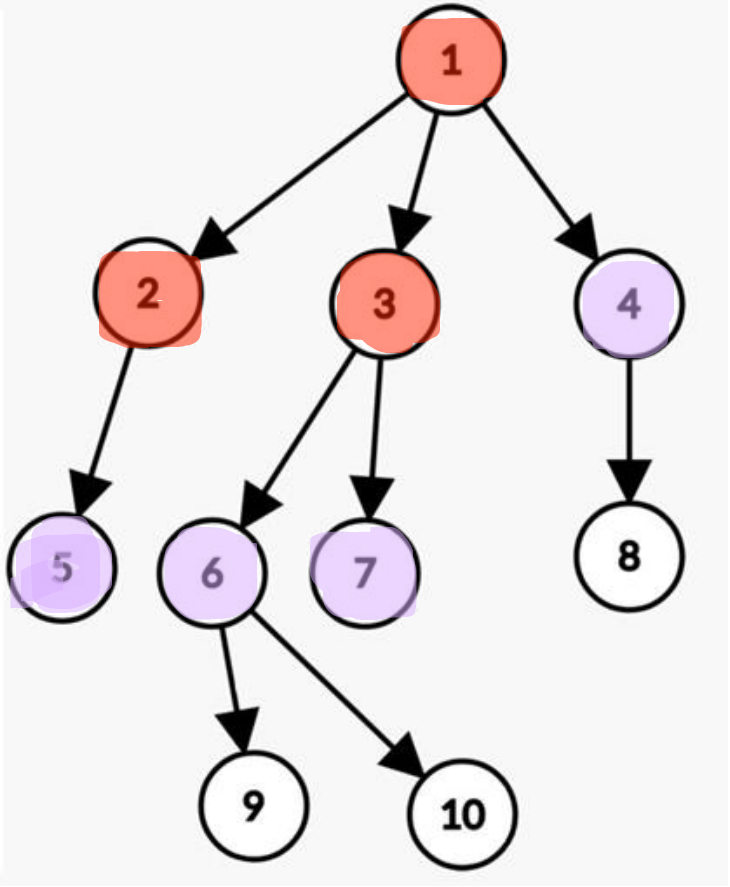



```
Closed:{1,2
}
Open:{3,4,5
}
Goal:{10}
```

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

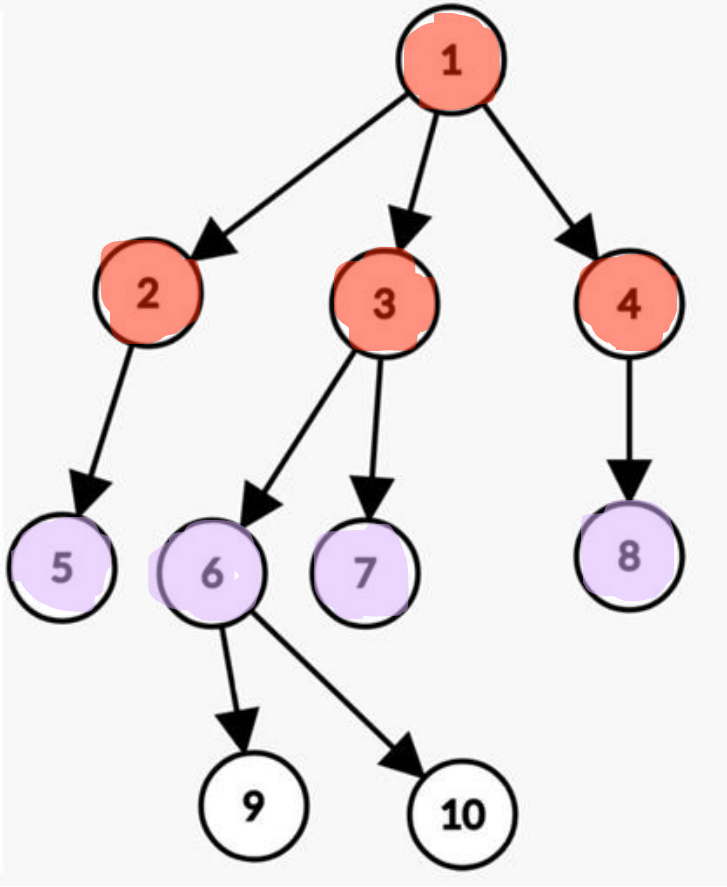


Closed:{1,2,3}
Open:{4,5,6,7}
Goal:{10}

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

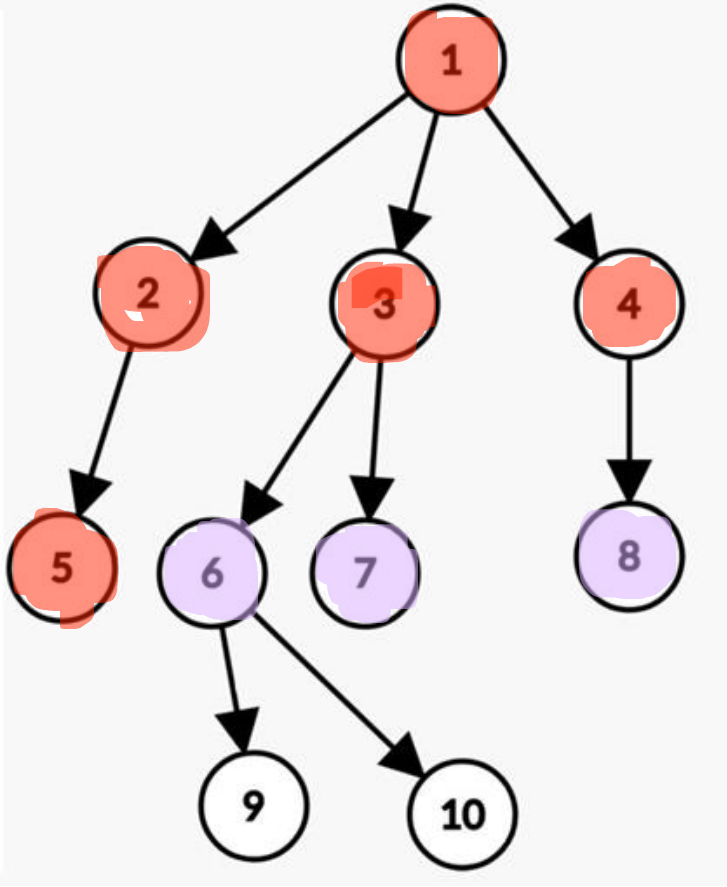


Closed:{1,2,3,4}
Open:{5,6,7,8}
Goal:{10}

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

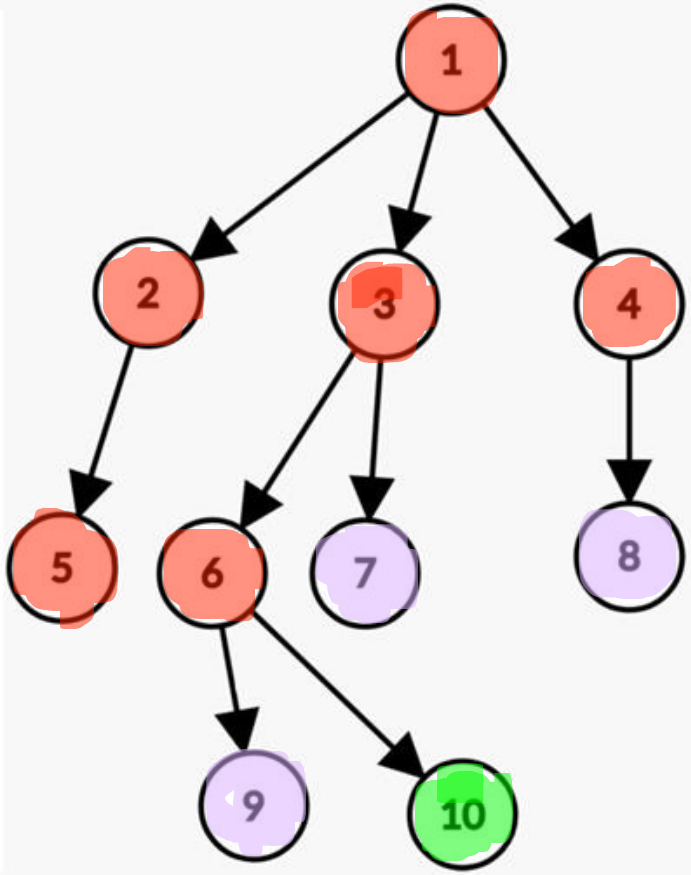



```
Closed:{1,2
,3,4,5}
Open:{6,7,8
}
Goal:{10}
```

BFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*



Closed:{1,2,3,4,5,6}
Open:{7,8,9}
Goal:{10}
RETORNA NODO 10

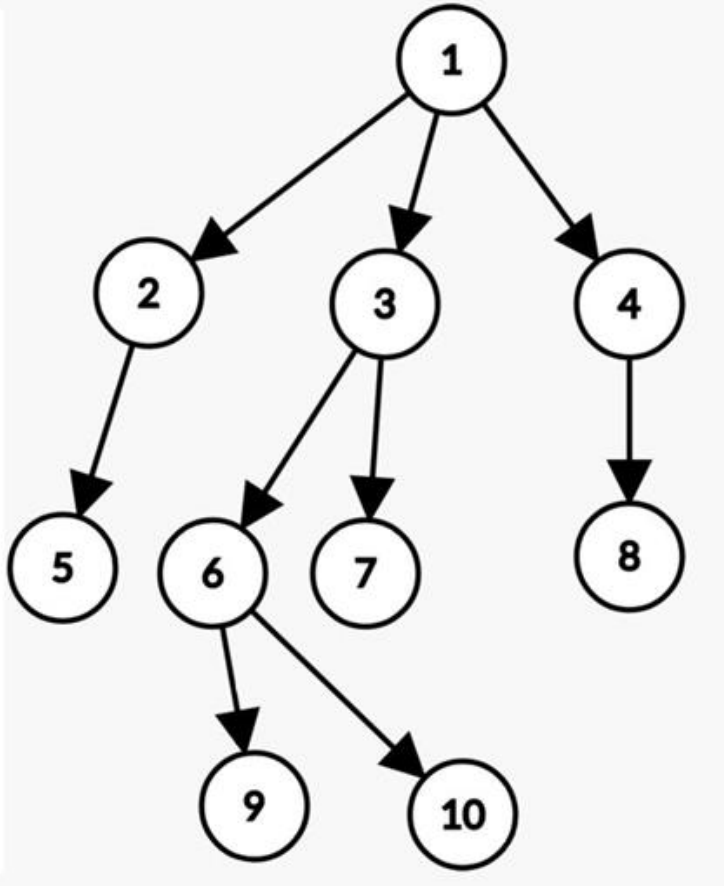
DFS

MENTI!

Al utilizar el pseudo código de DFS (ver imagen) sobre el siguiente grafo y suponiendo que el nodo 10 es el único objetivo, ¿Cómo queda Open y Closed al retornar el algoritmo si 1 es el estado inicial? Importante: Al

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

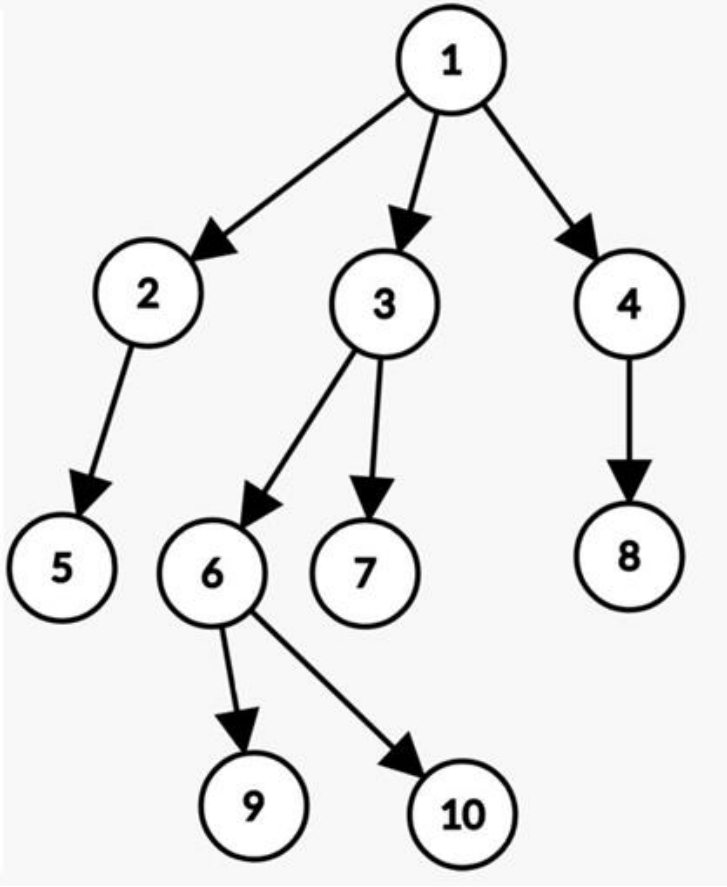
Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*



DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

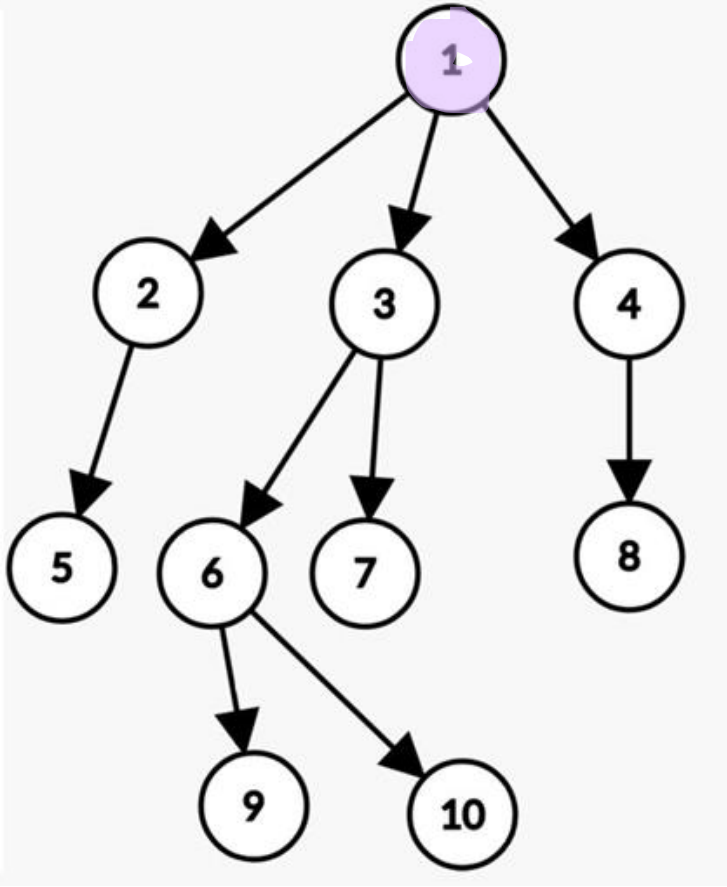



```
Closed:{}
Open:{}
Goal:{10}
```

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

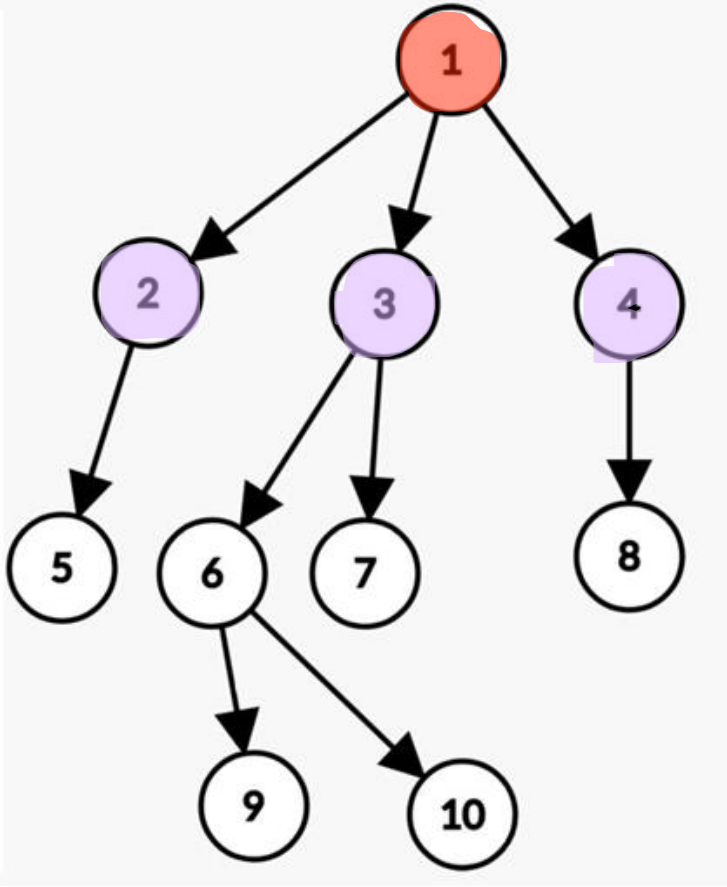


Closed:{}
Open:{1}
Goal:{10}

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*



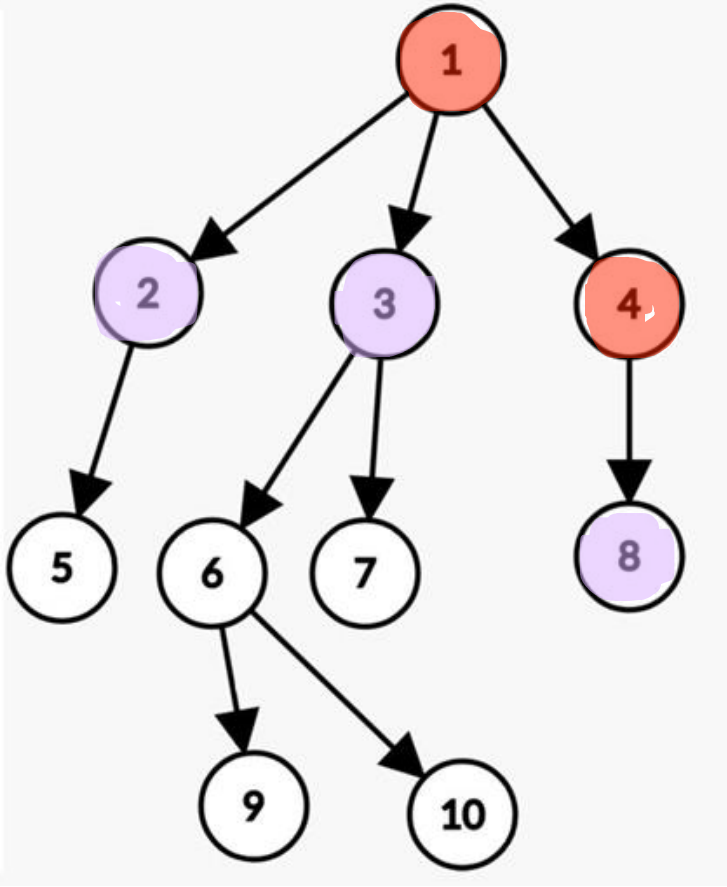

```
Closed:{1}

Open:{2,3,4
}
Goal:{10}
```

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

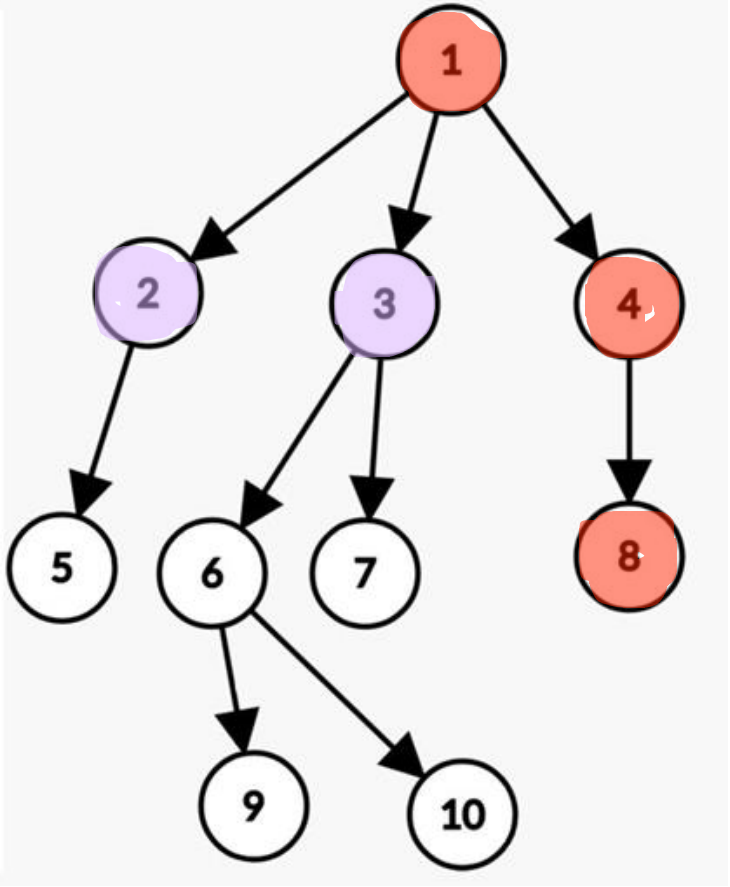



```
Closed:{1,4
}
Open:{2,3,8
}
Goal:{10}
```

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

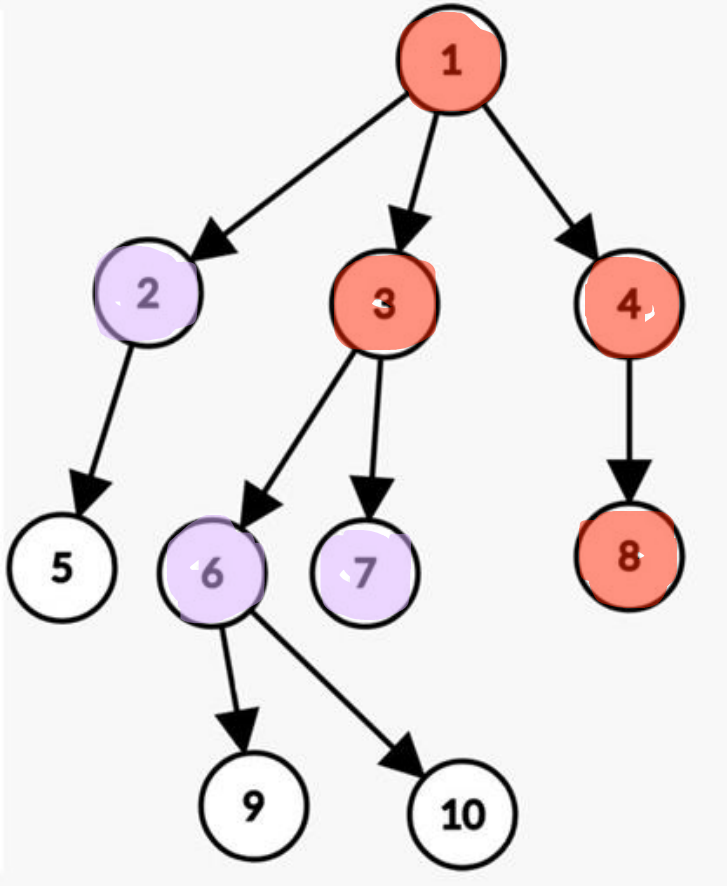


Closed:{1,4,8}
Open:{2,3}
Goal:{10}

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

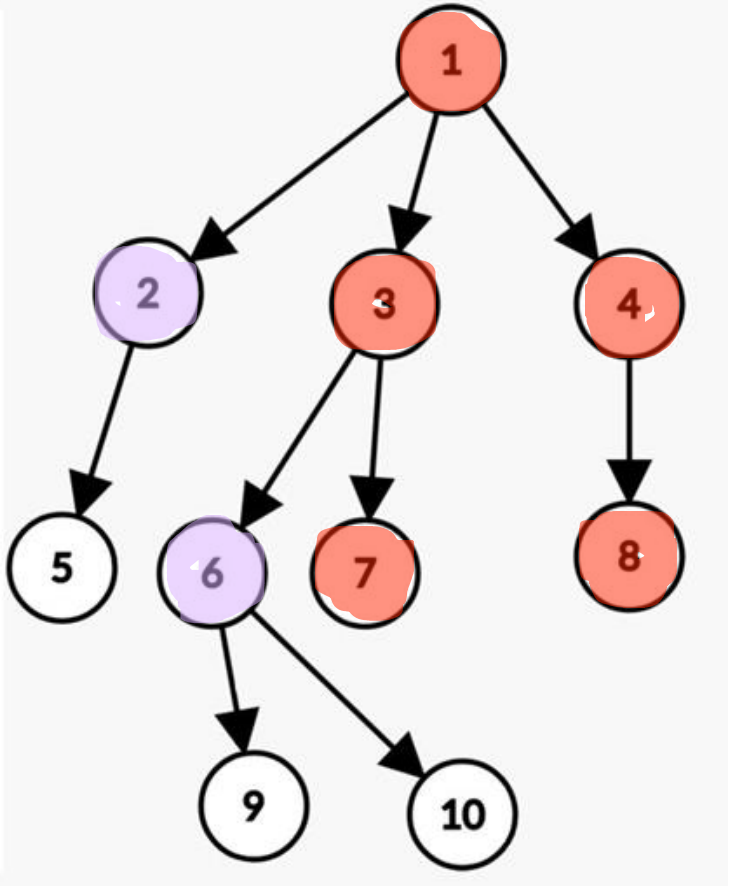



```
Closed:{1,4
,8,3}
Open:{2,6,7
}
Goal:{10}
```

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

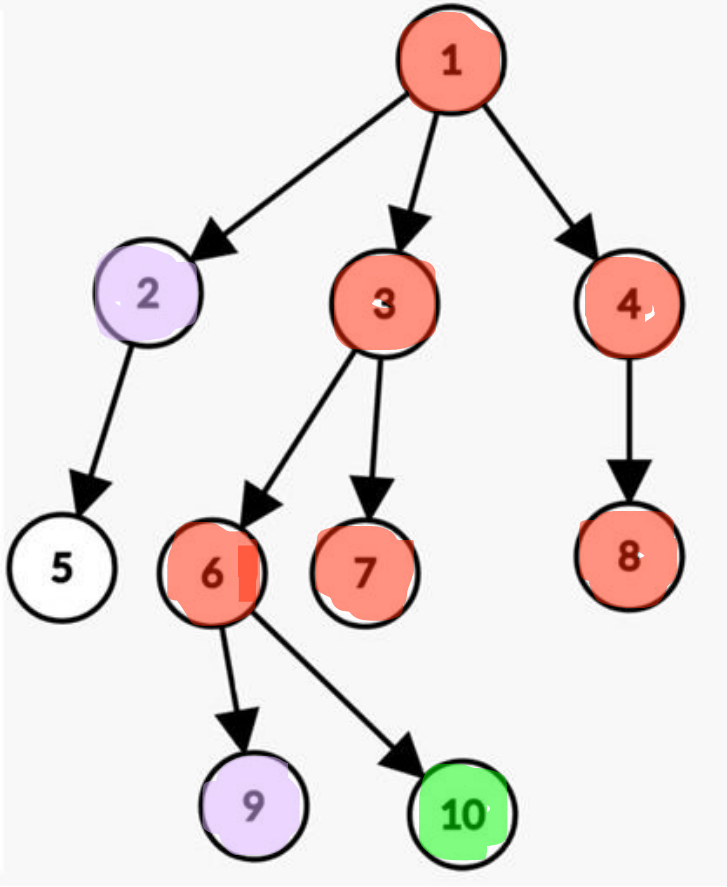


Closed:{1,4,8,3,7}
Open:{2,6}
Goal:{10}

DFS

El siguiente es un algoritmo de búsqueda genérico.
Input: Un problema de búsqueda (S, A ,s_{init}, G)
Output: Un nodo objetivo

Open es un contenedor vacío
Closed es un conjunto vacío
Inserta s_{init} a *Open*
parent(s_{init}) = null
while Open != ∅:
    u ← Extraer(*Open*)
    Inserta *u* en *Closed*
    for each *v* ∈ Succ(*u*) \ (*Open* ∪ *Closed*)
        parent(*v*) = *u*
        if *v* ∈ *G* **return** *v*
        Inserta *v* a *Open*

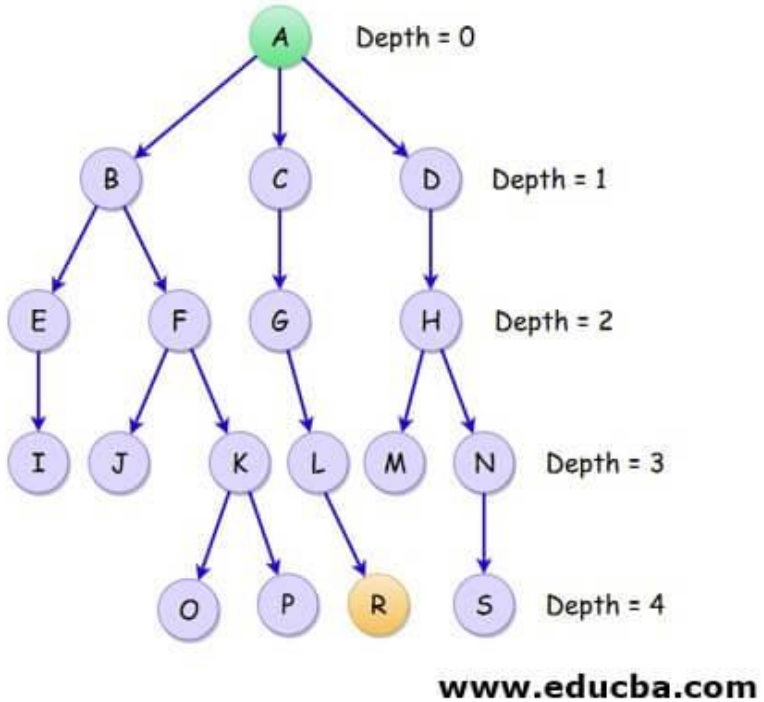


Closed:{1,4,8,3,7,6}
Open:{2,9}
Goal:{10}
RETORNA NODO 10

IDDFS

IDDFS

- Realiza una **búsqueda exhaustiva** en un **árbol o grafo**
- Se realiza de manera **iterativa**, aumentando gradualmente la profundidad máxima de búsqueda en cada iteración
- La **utilidad** de IDDFS radica en que **combina** la eficiencia de la **búsqueda en profundidad** con la **completitud de la búsqueda en anchura**

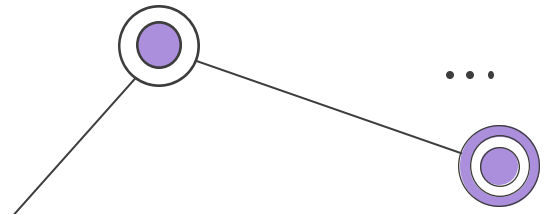


DIJKSTRA

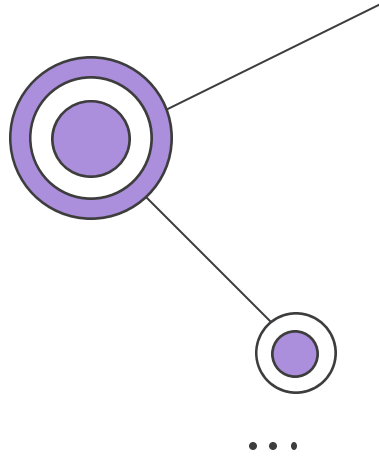
DIJKSTRA

- Algoritmo de búsqueda de **caminos más cortos en un grafo ponderado** (grafos con pesos en las aristas) y dirigido.
- 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**.

Heurísticas



Heurística



La función heurística *h(s)* es una función que estima la **distancia** entre un **estado *s*** y el estado **objetivo *G***.

Ésta es solo una **aproximación** (si entregara la distancia real al objetivo, ya estaría resuelto el problema, plop)

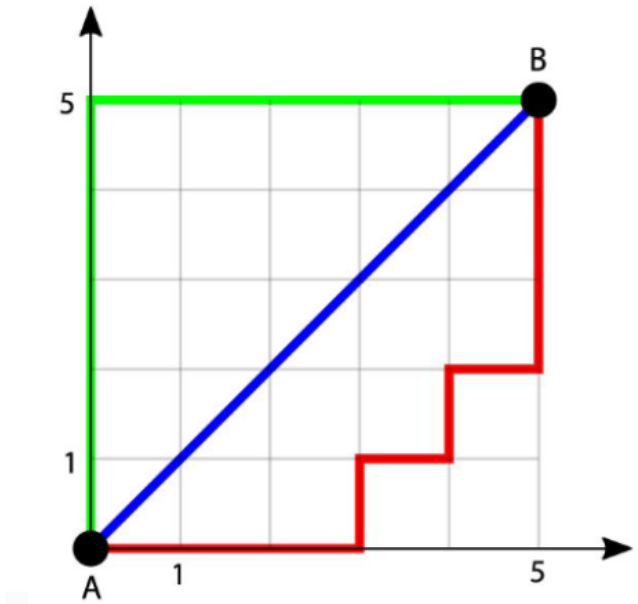
Distancia Manhattan y Euclidiana

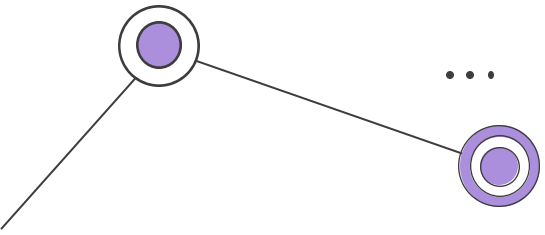
Distancia Manhattan: La suma de las diferencias absolutas de sus coordenadas

$$D_{Manhattan} = |x_1 - x_2| + |y_1 - y_2|$$

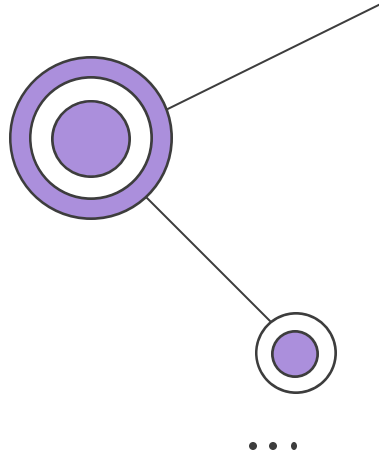
Distancia Euclidiana: La distancia en línea recta entre los puntos (pitágoras)

$$D_{Euclidiana} = \sqrt{[(x_1 - x_2)^2 + (y_1 - y_2)^2]}$$





Fin



Para el control de mañana les recomendamos **fuertemente** hacer los controles formativos de intro a la búsqueda y A*

Cualquier duda que les haya quedado nos pueden preguntar :)