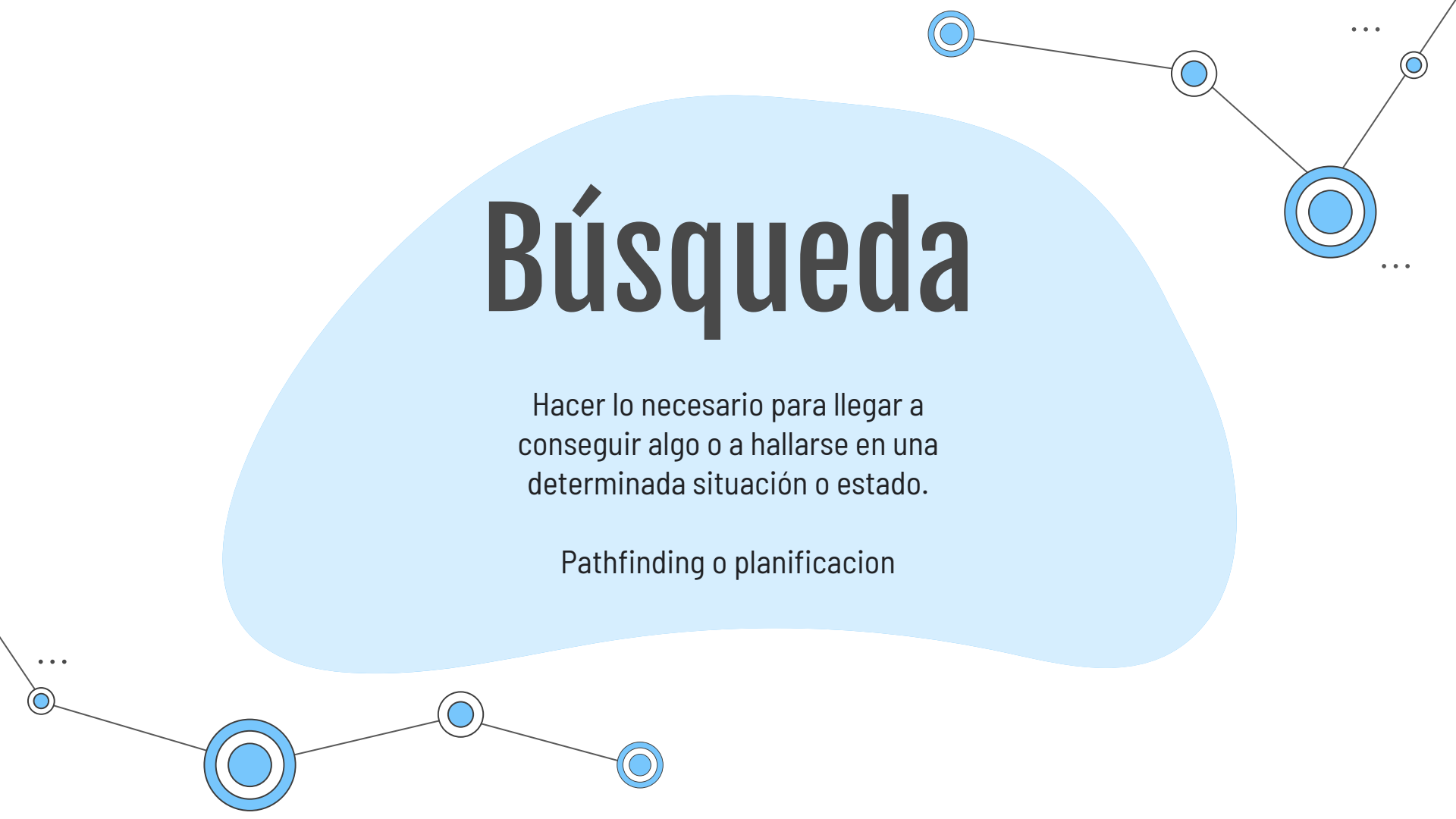


# MODELOS DE BÚSQUEDA

Andres Felipe Quebrada Agudelo  
1225091462



# Búsqueda

Hacer lo necesario para llegar a conseguir algo o a hallarse en una determinada situación o estado.

Pathfinding o planificación

# Contenido

01

...

## BÚSQUEDA NO INFORMADA O BÚSQUEDA CIEGA

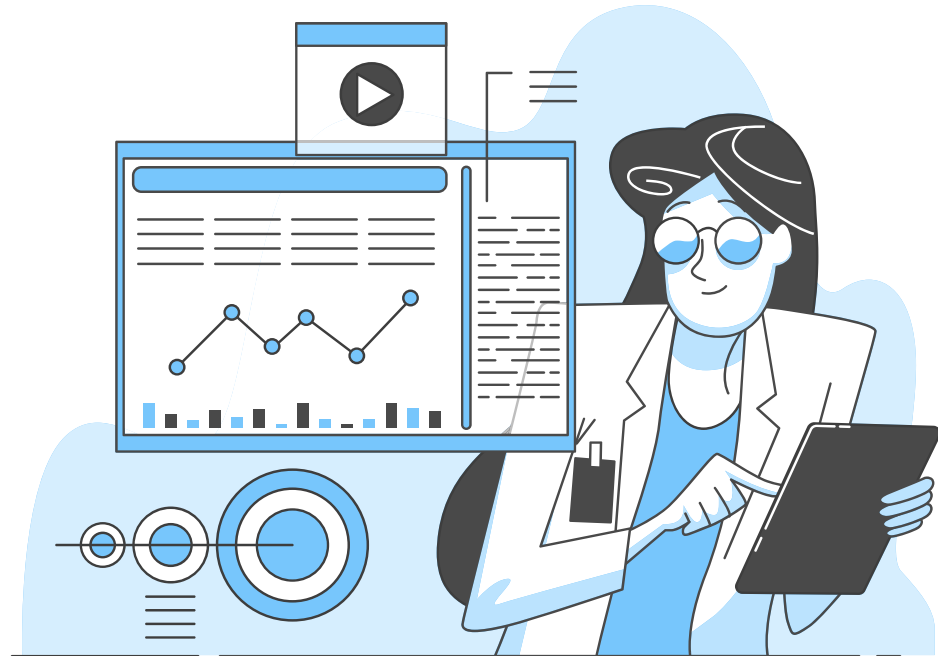
No conocemos el medio

02

...

## BÚSQUEDA INFORMADA O BÚSQUEDA GUIADA

Se tiene conocimiento del  
medio





# 01

***Búsqueda no  
informada***



# Métodos



Amplitud

Breadth First  
Search  
(BFS)

...

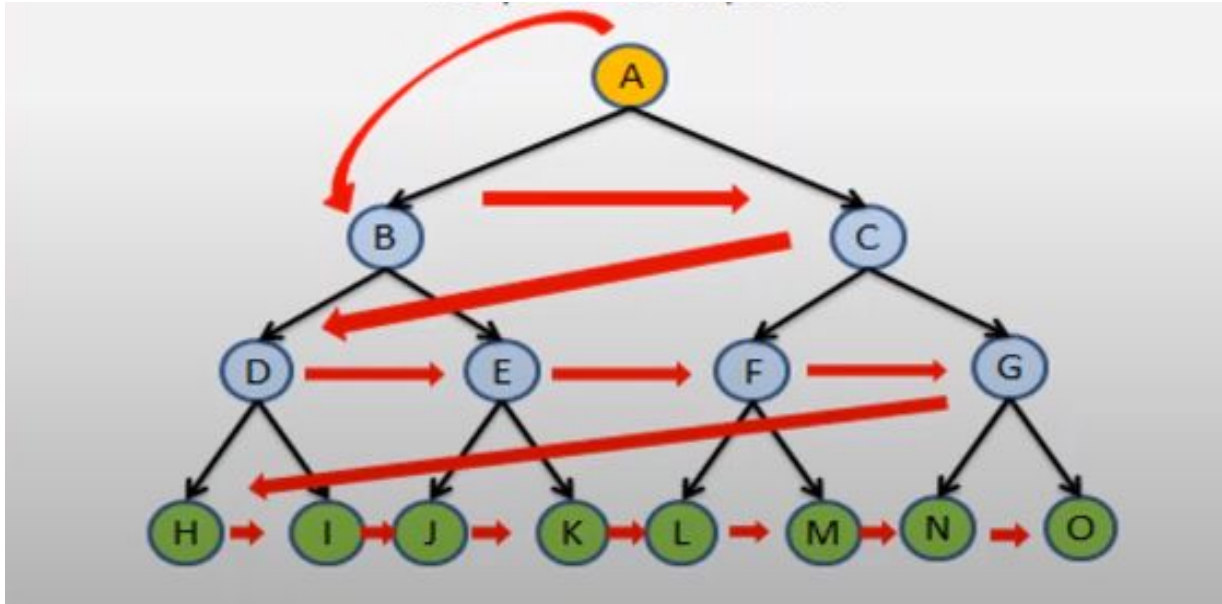


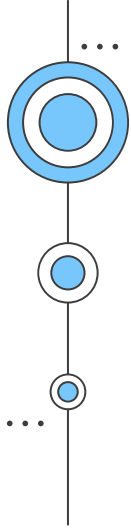
Profundidad

Depth First  
Search  
(DFS)

...

# AMPLITUD





## Pseudocódigo Algoritmo:

Establecer nodo origen

Evaluar primer hijo

    si cumple, establecer como origen y salir

    si valido, repetir búsqueda a partir del nuevo

estado

    sino valido, repetir búsqueda para todos los

hermanos

    si encuentra , establecer como origen y salir

    si no encuentra, marcar al padre como no valido

    establecer origen como abuelo y seguir buscando.





# Ventajas



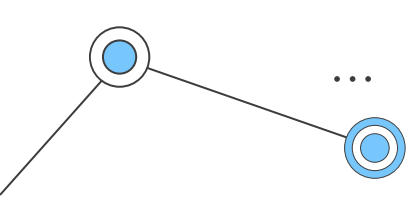
01

Garantiza encontrar  
una solución, siempre  
que exista

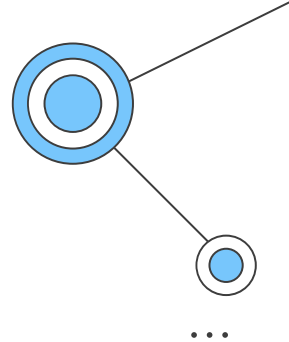
02

No queda atrapada en  
callejones sin salida





# Desventajas

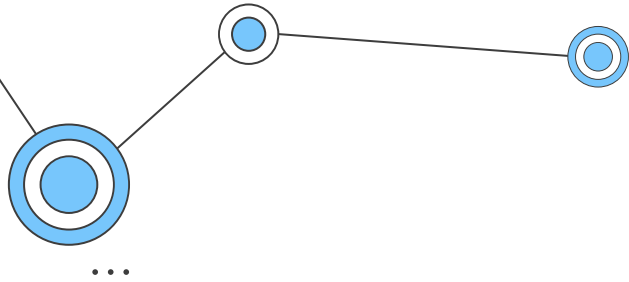


01

La complejidad  
computacional crece  
rápidamente

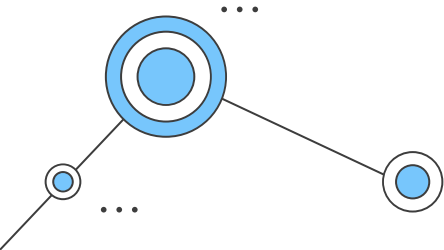
02

No tiene porque ser  
óptimo

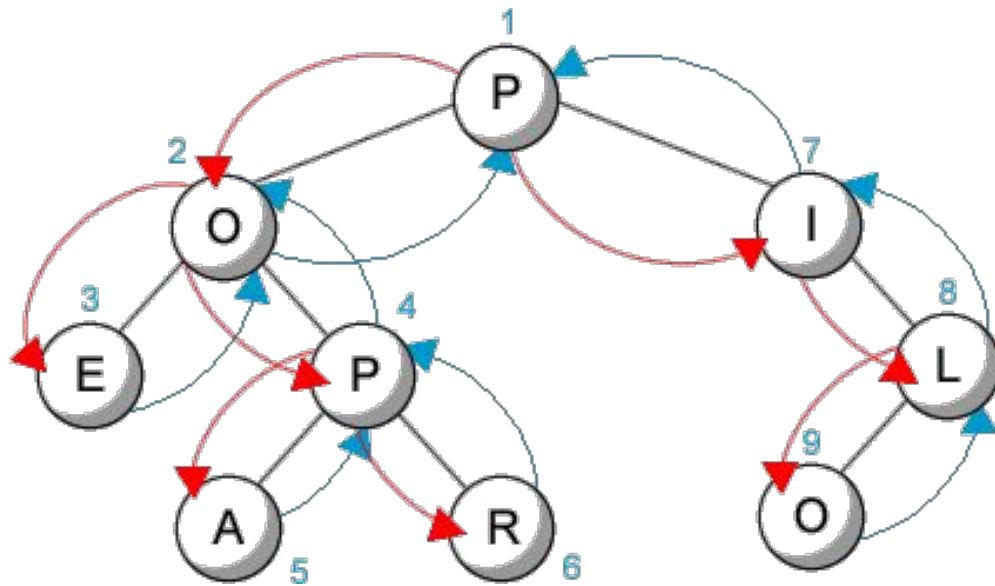


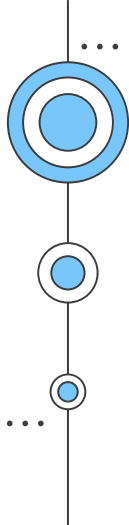
Ver pseudocódigo y código en  
github

[Búsqueda en Amplitud](#)



# PROFUNDIDAD





## Pseudocódigo algoritmo búsqueda en profundidad

función buscar\_en\_hijos(Nodo:n)

variable encontrado=boolean

inicio

si solucion(n->hijo)

retornar n->hijo

sino

n1=n->hijo

encontrado=falso

mientras no (encontrado)

n1=n1->hermano

Si solucion(n1)

retornar n1

sino

n1=null

romper ciclo

buscar\_en\_hijos(n->hijo)

n2->n->hijo

mientras(n2->hermano!=null)

n2=n2->hermano

buscar\_en\_hijos(n2)

fin si

fin mientras

fin funcion





# Ventajas

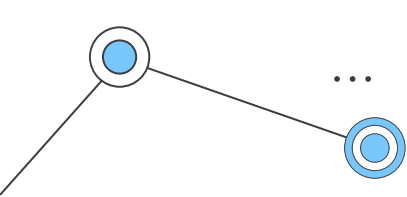


01

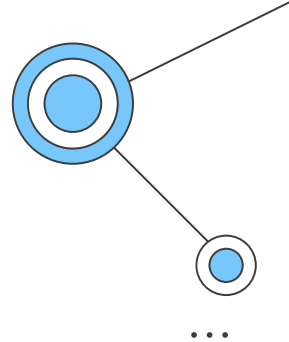
Tiene menor  
complejidad espacial  
que la búsqueda en  
amplitud

02

Expandir un rama hasta  
su máxima expresión  
puede ser útil para  
acortar la solución



# Desventajas

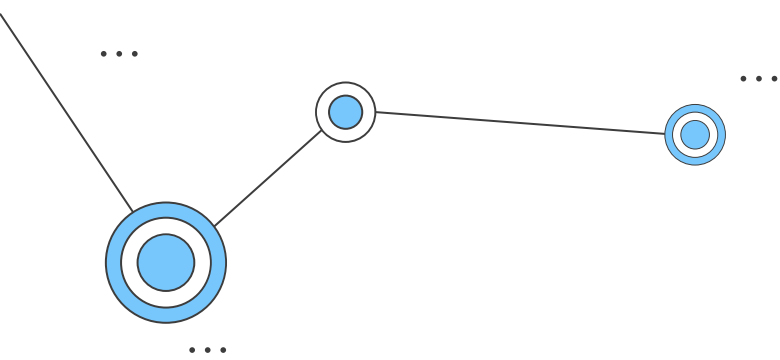


01

No es completo, si  
existe una solución  
puede no encontrarla

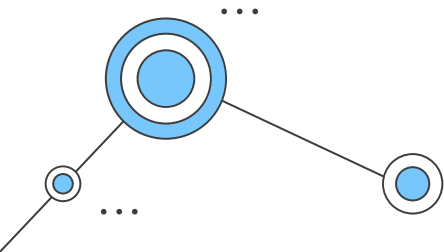
02

No tiene porque ser  
óptimo



código en github

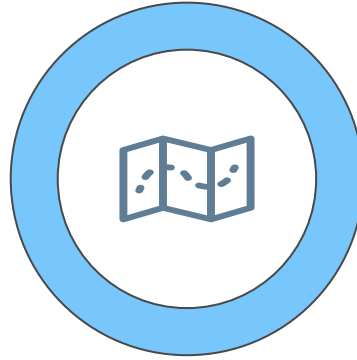
Búsqueda en profundidad



# 02

## *Búsqueda informada*





Explorar en primer lugar aquellas trayectorias que parecen más prometedoras a la hora de conducir a una solución (Heurística).

Ahora, a diferencia de la búsqueda ciega, se le asignará a cada nodo un valor que dará una idea de lo cerca que está de la meta.

...

# Métodos

Primero el mejor

...

Método de  
gradiente

...

Algoritmo A\*

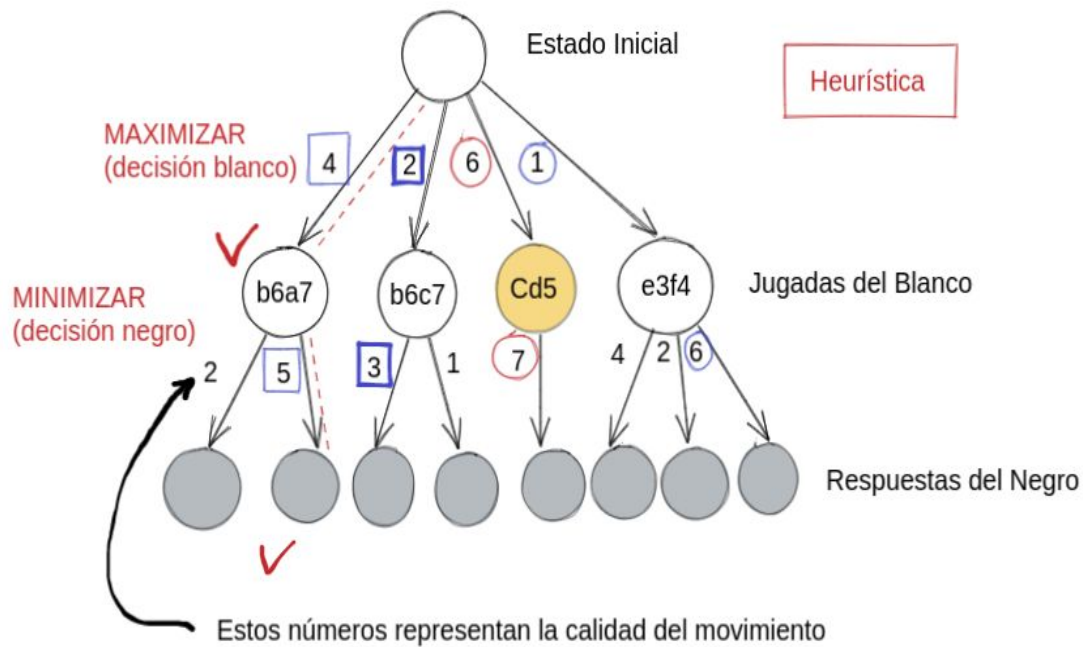
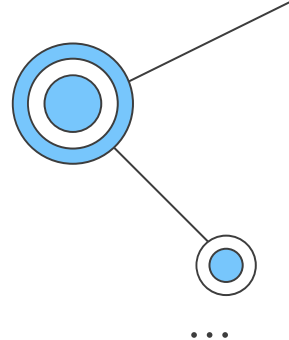
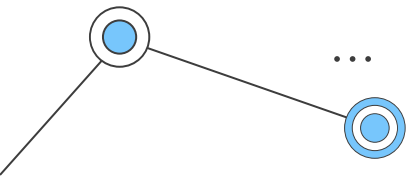
Mas utilizado

...

Búsqueda con  
adversarios

Método Minimax  
Método de poda  $\alpha - \beta$

...



**Procedimiento MINIMAX:**MINIMAX ( $m$ ,  $profundidad$ ,  $jugador$ )

1. Si  $m$  no tiene sucesores o si se considera que  $m$  ha alcanzado el límite de profundidad en  $profundidad$ , devolver  $fev(m)$  si  $m$  es un nodo MAX. En las mismas condiciones anteriores, devolver  $-fev(m)$  si  $m$  es un nodo MIN (Recordamos aquí que  $fev(m)$  representa lo prometedor que es el nodo  $m$  para MAX, independientemente de si  $m$  es un nodo MAX o MIN.).

2. Generar los sucesores de  $m$ :

2.1. Asignar a la variable  $mejor$  el valor mínimo que  $fev$  pueda tener (puede ser un valor de referencia previamente establecido).

$$mejor = \min, \{fev(j) \ \forall j\}$$

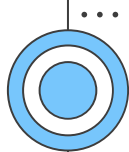
2.2. Para cada sucesor  $n$  de  $m$ :

(1)  $M(n) = \text{MINIMAX}(n, profundidad + 1, C(jugador))$   
 $C(jugador)$  es una función que cambia de jugador.

(2)  $mejor = \max(-M(n), mejor)$

3. Una vez que se han analizado recursivamente todos los sucesores de un determinado nivel (indicado por la variable  $profundidad$ ), se devuelve el valor  $mejor$ .

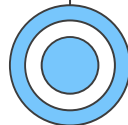
Observación: la primera llamada al procedimiento sería MINIMAX( $nodo-inicial$ , 0, MAX).



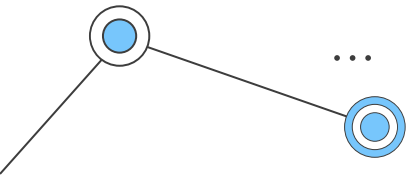
...



...



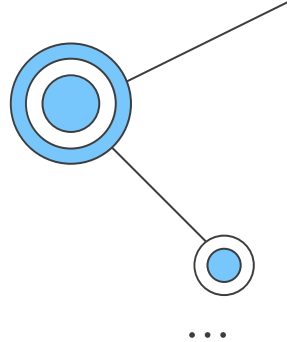
...

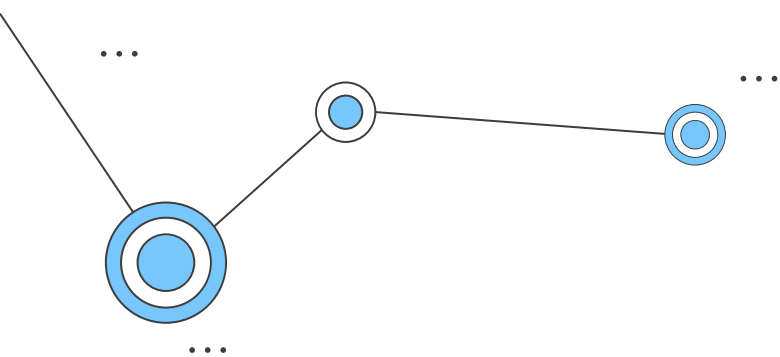


```
graph = {0: [4, 2.1, 6, 1],  
         4: [2, 5],  
         2.1: [3, 1.0],  
         6: [7],  
         1: [4.0, 2, 6]  
        }
```

```
def Minimax (m, profundidad, graph):  
    if graph.get(m) == None:  
        return m  
  
    sucesores = graph.get(m)  
    print(m)  
    mejor = min(sucesores)  
  
    for n in sucesores:  
        M = Minimax(n, profundidad + 1, graph)  
  
        mejor = max(M, mejor)  
  
    return mejor  
  
mejor = Minimax(0, 0, graph)  
print(mejor)
```

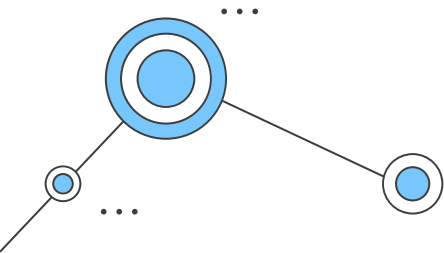
0  
4  
2





código en github

[Heuristica](#)



A decorative network diagram with blue nodes and lines. The nodes are represented by concentric circles, with some having a solid blue center and others being hollow. They are connected by thin black lines. There are three main clusters: a small cluster in the top right, a larger cluster in the bottom left, and a single node in the bottom right. Ellipses (...) are used to indicate that the network continues beyond the visible nodes.

**Gracias**