

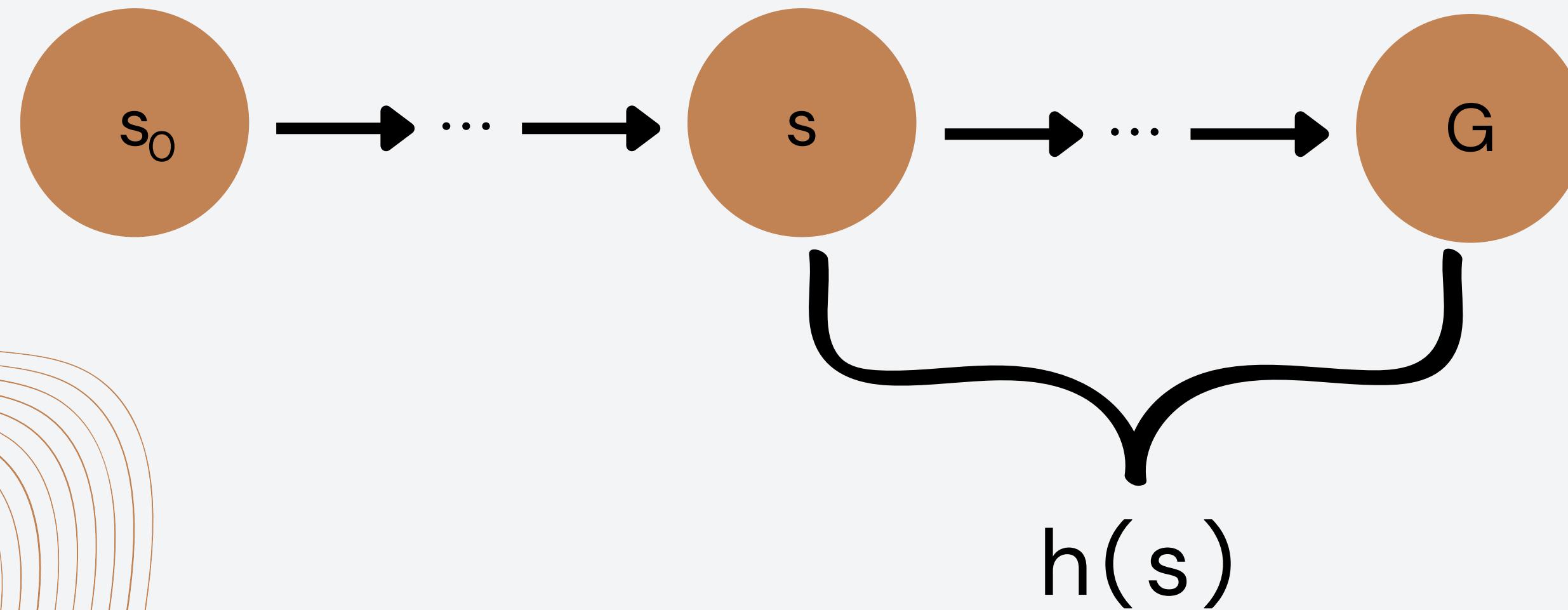
AYUDANTÍA 05

A*

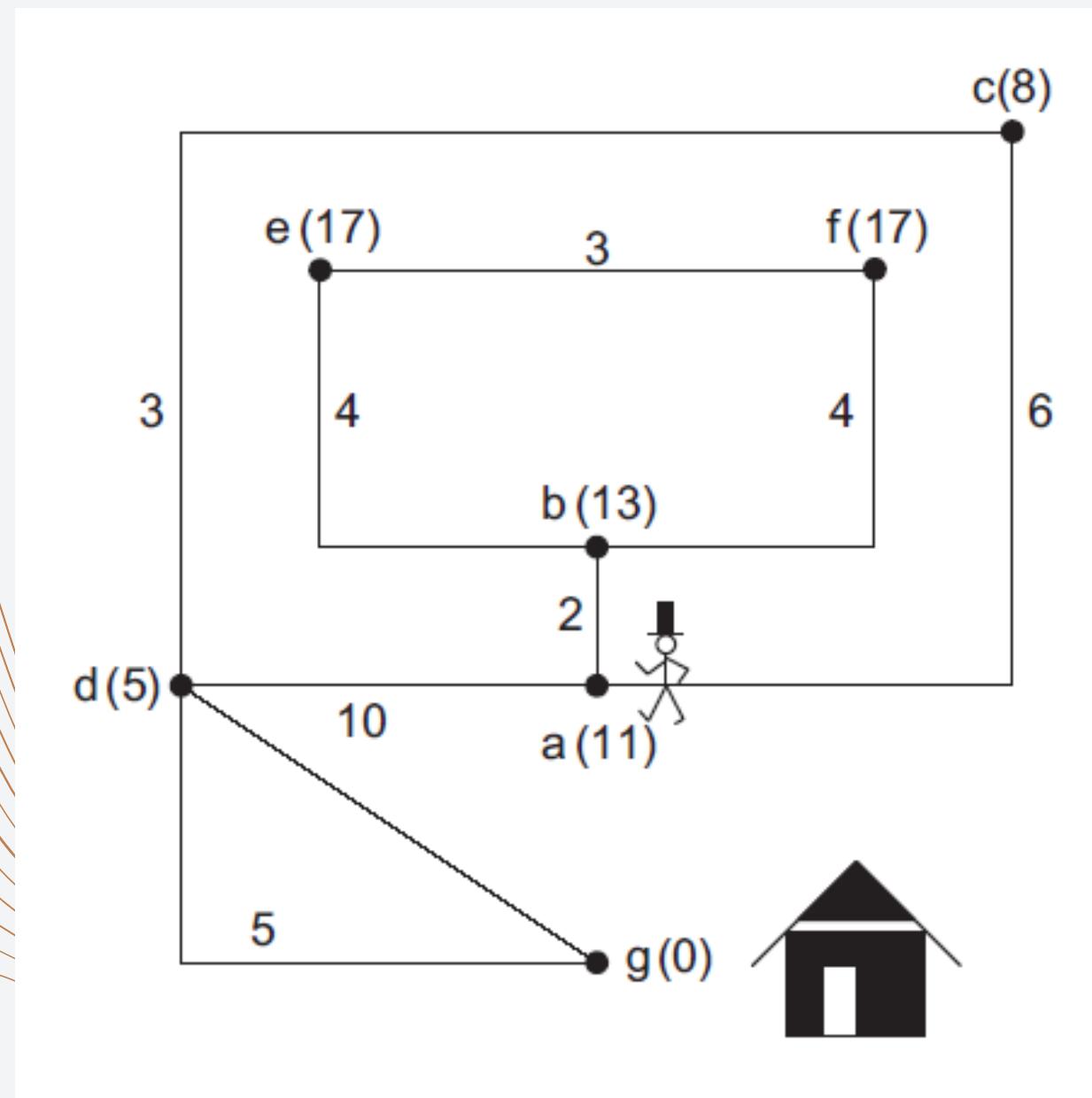
HEURÍSTICAS

HEURÍSTICAS

Es una función que **estima** qué tan cerca está un estado **s** del estado final **G**

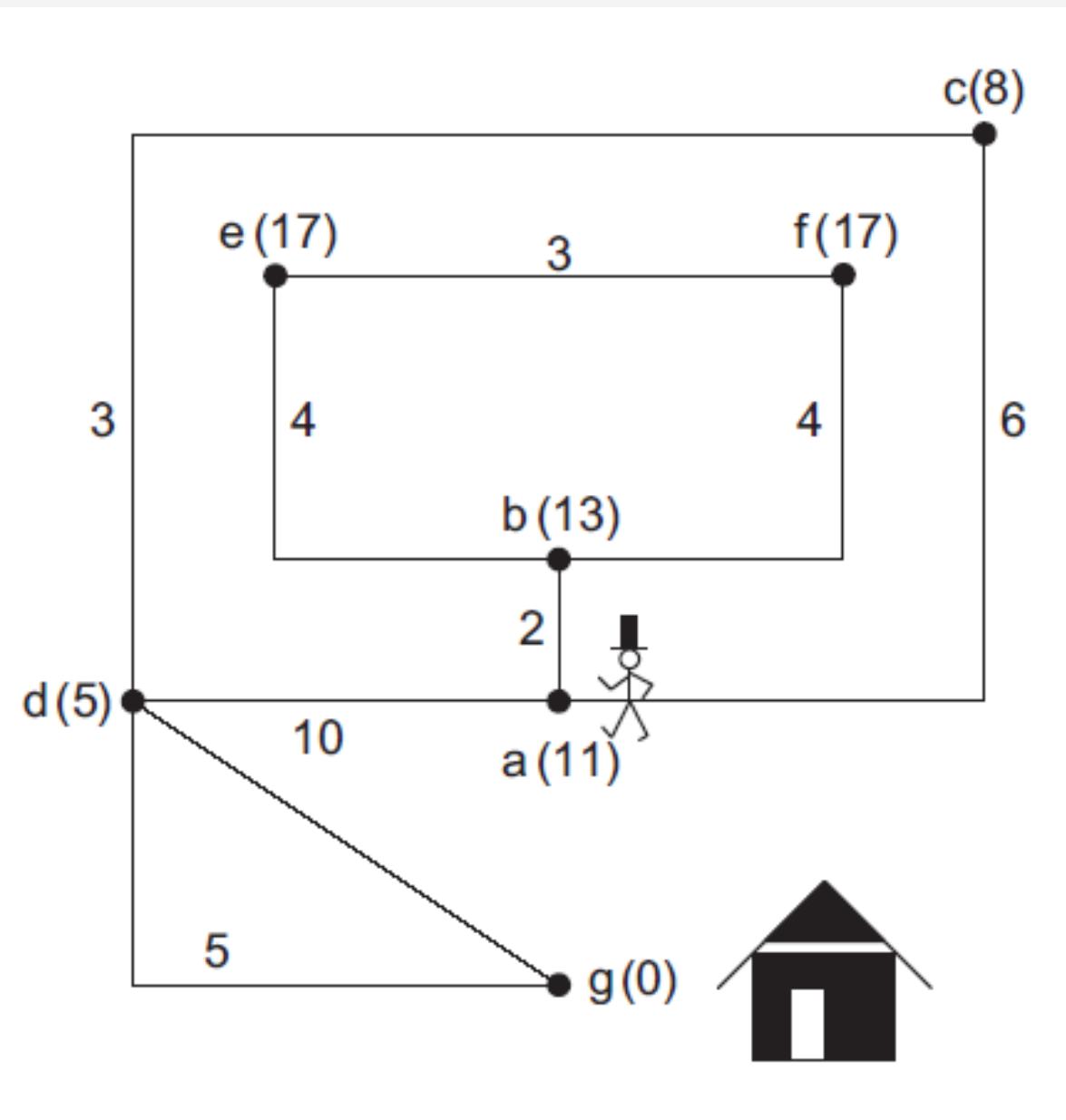


HEURÍSTICAS



$s =$	a	b	c	d	e	f	g
$h(s) =$	11	13	8	5	17	17	0

HEURÍSTICAS ADMISIBLES



$s =$	a	b	c	d	e	f	g
$h(s) =$	11	13	8	5	17	17	0
$h^*(s) =$	14	16	8	5	20	20	0

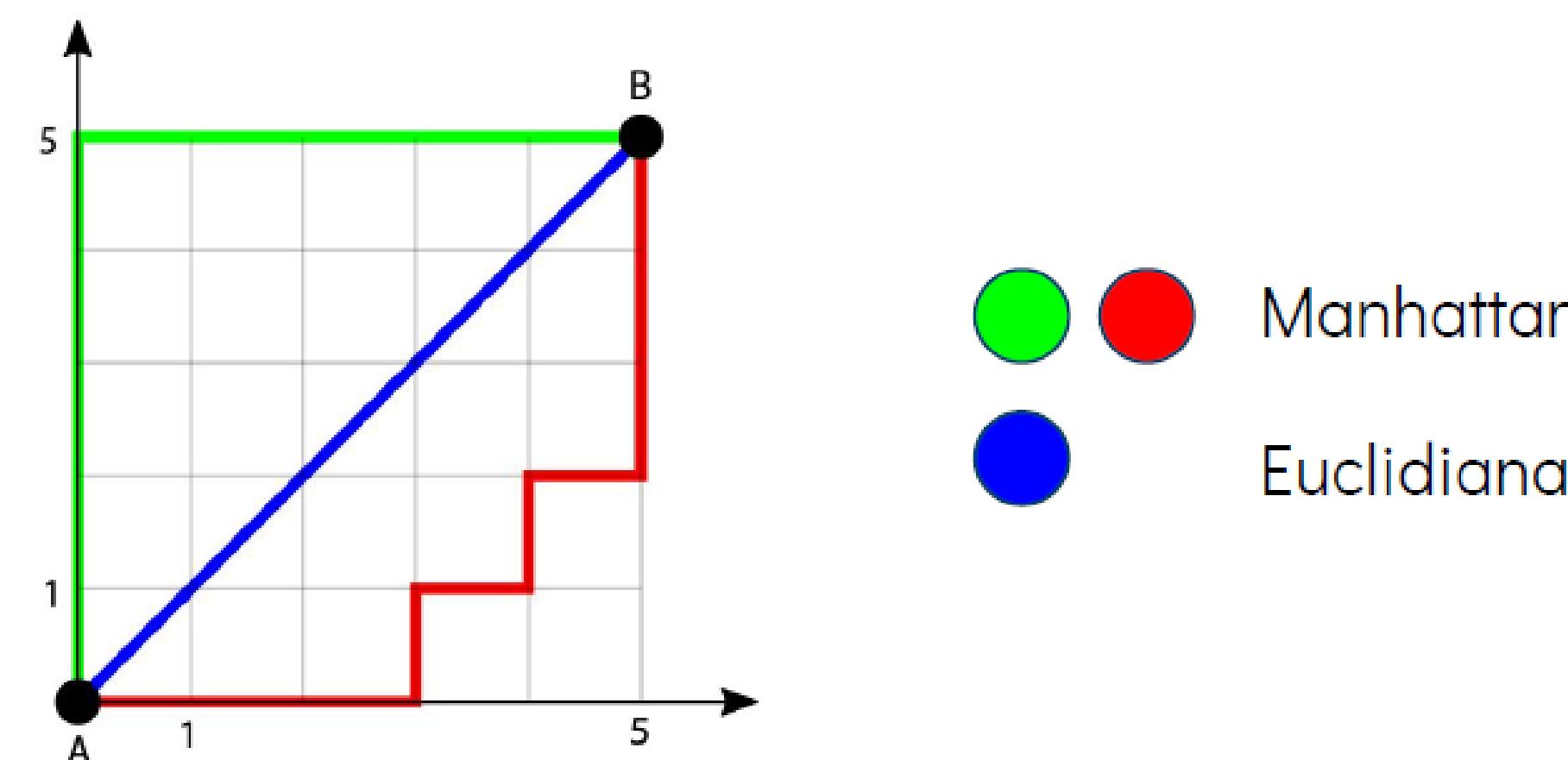
$h^*(s)$: Costo del camino óptimo desde s

DISTANCIA MANHATTAN Y EUCLIDIANA

La distancia entre dos puntos es:

- **Manhattan**: la suma de las diferencias absolutas de sus coordenadas. $D_{\text{Manhattan}} = |x_1 - x_2| + |y_1 - y_2|$
- **Euclidiana**: la distancia en línea recta entre los puntos

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



ALGORITMO A*

ALGORITMO A*

Algoritmo A*

Input: Un problema de búsqueda (S, A, s_0, G)

Output: Un nodo objetivo

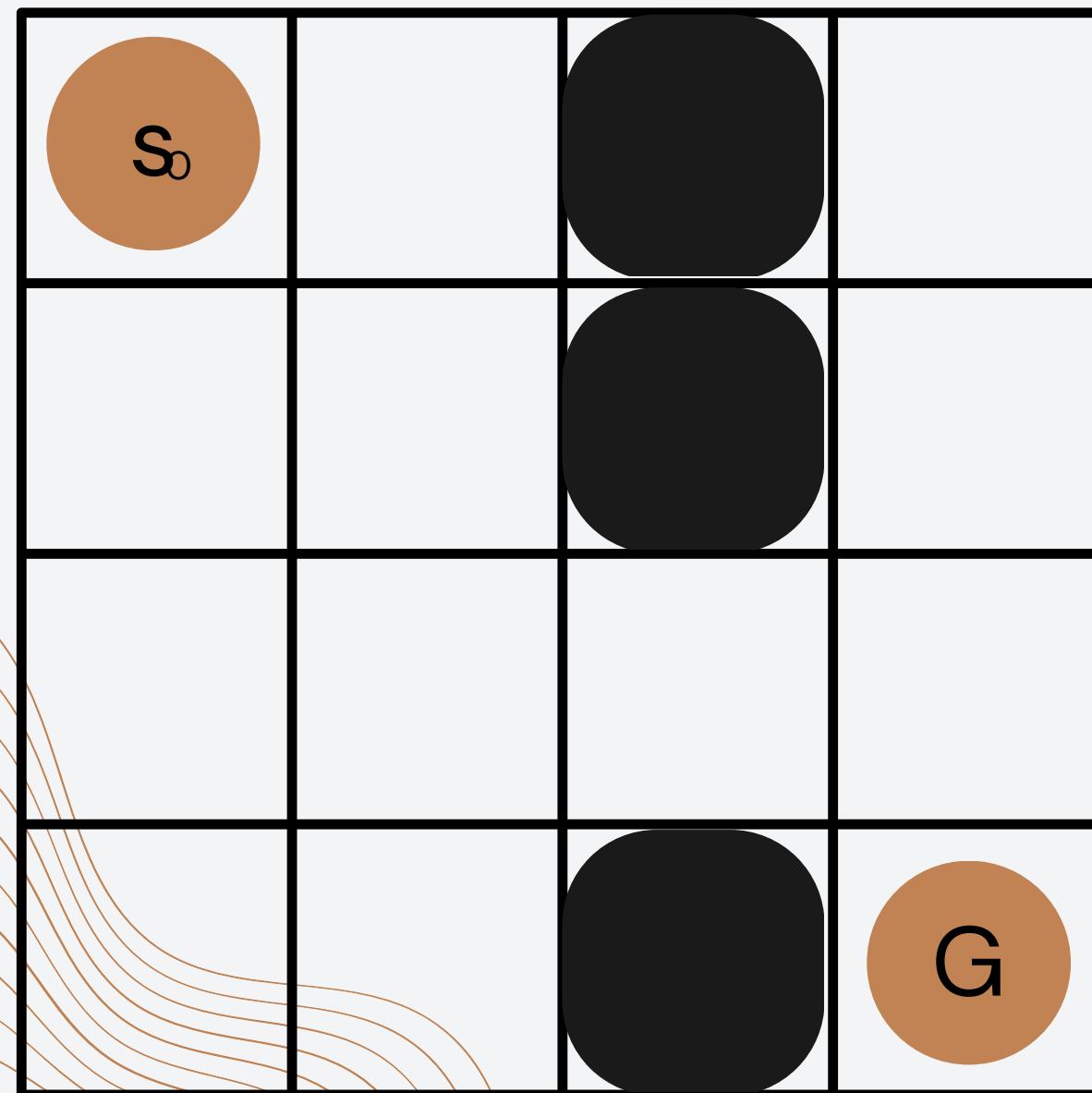
- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
 - 5 Extrae un u desde $Open$ con menor valor- f
 - 6 **if** u es objetivo **return** u
 - 7 **for each** $v \in Succ(u)$ **do**
 - 8 Insertar v

$Open$ se ordena según menor valor de f

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

Se revisa si se llega al nodo objetivo cuando se 'expande'
¿ $s=G$?

EJEMPLO



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

$g(s)$: largo del camino

$h(s)$: distancia Manhattan

$f(s)$	$g(s)$
$h(s)$	

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

s_0	inf	inf	inf
6	5	3	3
inf	inf	inf	inf
5	4	2	2
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	inf	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2 Open  $\leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
     1  $cost_v = g(u) + c(u, v)$ 
     2 if  $cost_v \geq g(v)$  return
     3  $parent(v) \leftarrow u$ 
     4  $g(v) \leftarrow cost_v$ 
     5  $f(v) \leftarrow g(v) + h(v)$ 
     6 if  $v \in Open$  then Reordenar  $Open$ 
     7 else Insertar  $v$  en  $Open$ 

```

s_0	inf	5	3
	inf	inf	inf
5	4	inf	2
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	G	0

```

1 for each  $s \in \mathcal{S}$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 

4 while  $Open \neq \emptyset$ 
5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
1  $cost_v = g(u) + c(u, v)$ 
2 if  $cost_v \geq g(v)$  return
3  $parent(v) \leftarrow u$ 
4  $g(v) \leftarrow cost_v$ 
5  $f(v) \leftarrow g(v) + h(v)$ 
6 if  $v \in Open$  then Reordenar  $Open$ 
7 else Insertar  $v$  en  $Open$ 

```



6	0	inf	inf
s_0	6	5	3
inf	inf	inf	inf
5	4	2	2
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	0	0
G			

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
    1  $cost_v = g(u) + c(u, v)$ 
    2 if  $cost_v \geq g(v)$  return
    3  $parent(v) \leftarrow u$ 
    4  $g(v) \leftarrow cost_v$ 
    5  $f(v) \leftarrow g(v) + h(v)$ 
    6 if  $v \in Open$  then Reordenar  $Open$ 
    7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates a search space for a pathfinding algorithm. The grid consists of 16 cells arranged in a 4x4 pattern. The start node s_0 is located at the top-left cell (0,0). The goal node G is located at the bottom-right cell (3,3). The following table summarizes the values assigned to each cell:

Column	Row 0	Row 1	Row 2	Row 3
0	6 (s ₀)	inf	inf	inf
1	inf	inf	inf	inf
2	5	4	inf	2
3	inf	inf	inf	inf

Note: The first column is labeled 'Column' and the first row is labeled 'Row 0'.

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    )
  
```

6 s_0 6	0	inf		inf
inf	5	inf		3
5	4	inf		inf
inf	inf	inf		2
4	3	2	inf	1
inf	inf	inf	inf	inf
3	2	G	inf	0

```
1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
```

	0	inf	
6	s_o	(0,0)	5
6			3
inf		inf	inf
5		4	2
inf		inf	inf
4		3	1
inf		inf	inf
3		2	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	1	inf
6	(0,0)	5	3
inf	inf	5	2
5	4	inf	inf
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
6	(0,0)	5			3
inf		inf			inf
5		4			2
inf		inf			inf
4		3			1
inf		inf			inf
3		2			0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the execution of the A* search algorithm on a grid-based problem. The grid has four columns and four rows. The first column is yellow (state space), the second is green (open list), the third is black (closed list), and the fourth is white (goal space).

- Row 1:** Yellow cell contains s_0 . Green cell contains $(0,0)$ with value 6. Black cell is empty. White cell contains 3.
- Row 2:** Yellow cell contains inf. Green cell contains inf. Black cell is empty. White cell contains 2.
- Row 3:** Yellow cell contains inf. Green cell contains inf. Black cell is empty. White cell contains 1.
- Row 4:** Yellow cell contains inf. Green cell contains inf. Black cell is empty. White cell contains 0.

The green cells represent the open list, showing the current f-values for each node. The black cells represent the closed list, where no further expansion is possible. The white cells represent the state space, showing the cost from the start node to reach that state. The goal node G is located in the bottom-right white cell.

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

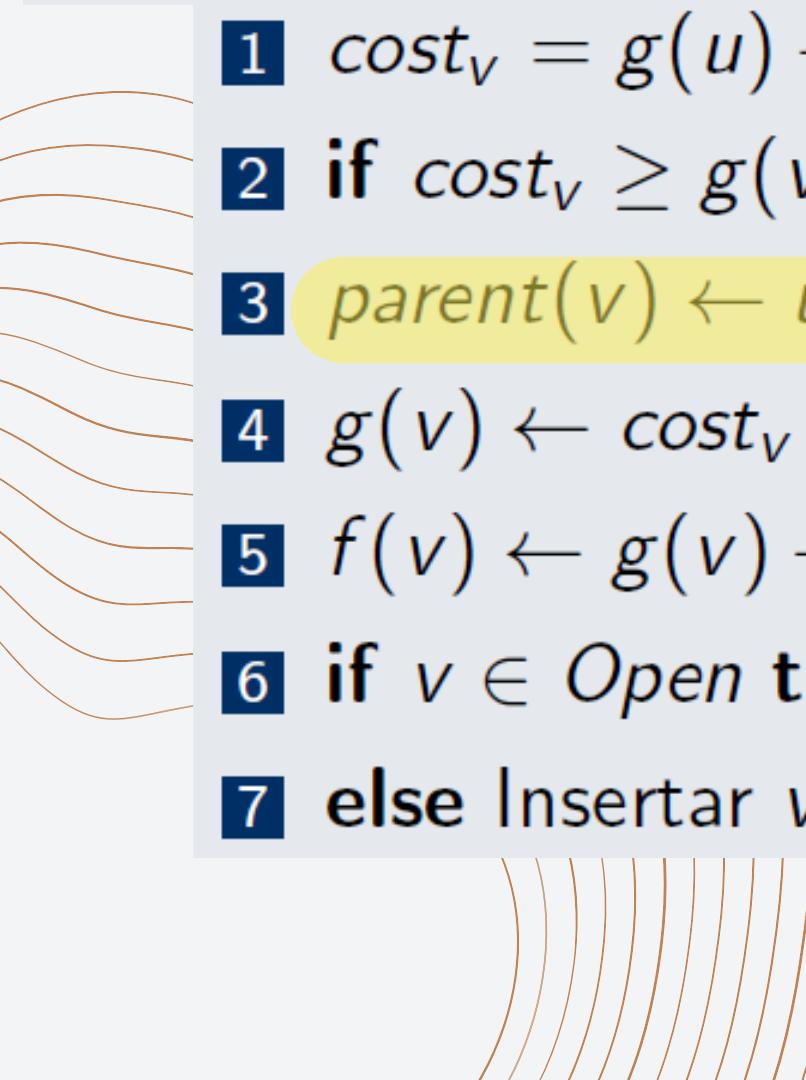
```

6	0	6	1
6	(0,0)	5	inf
inf	inf	4	2
5	4	inf	inf
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	0	inf

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

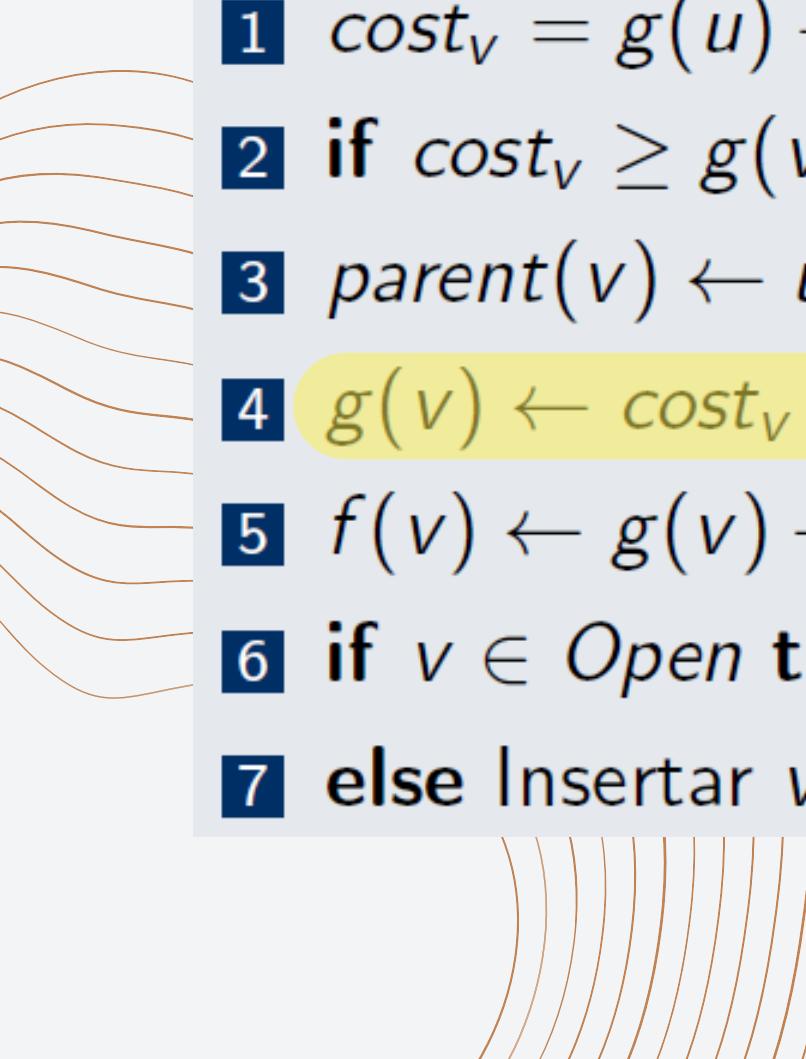


6 s_0 6	0 (0,0) 5	6 (0,0) 5	1	inf
inf (0,0) 5	inf	inf	4	2
inf	inf	inf	inf	inf
4	3	2	1	0
inf	inf	inf	inf	G

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

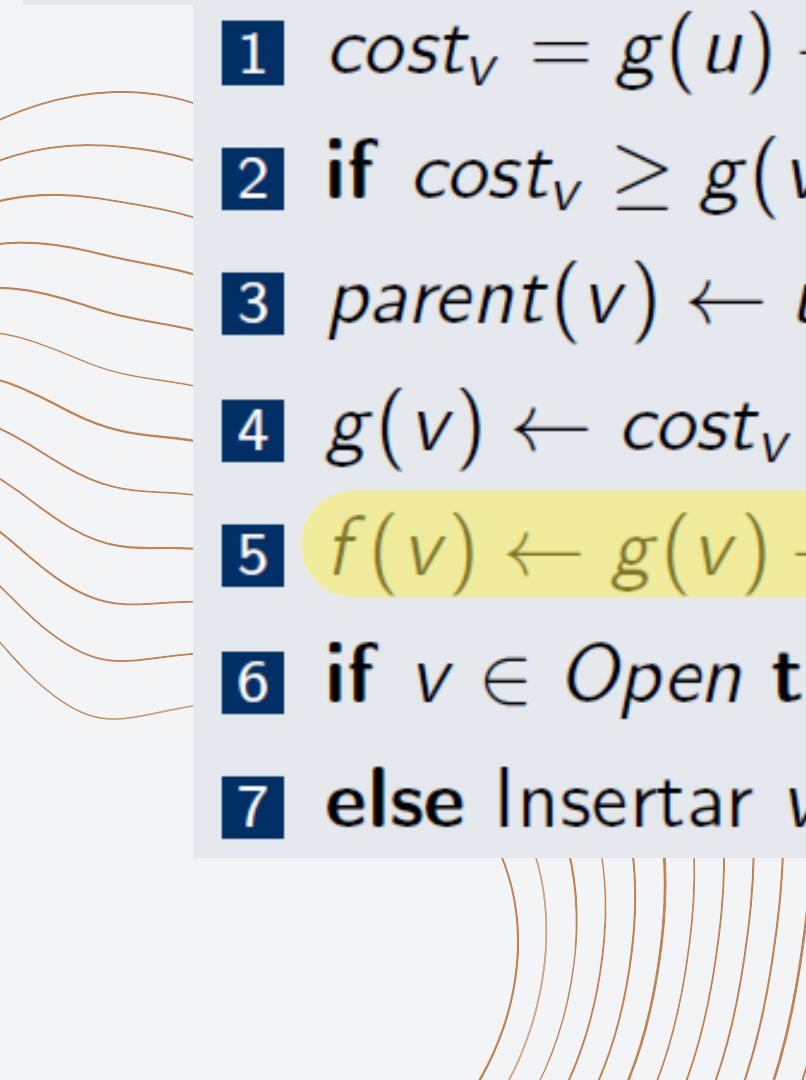


6 s_0 6	0 (0,0) 5	6 (0,0) 5	inf
1 (0,0) 5	inf	4	inf
inf	inf	inf	2
4	3	2	1
inf	inf	inf	inf
3	2	G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1	inf
s_0		(0,0)	5	3
6	1	inf		inf
(0,0)	5	4		2
inf	inf	inf	inf	inf
4	3	2	1	0
inf	inf	inf	inf	inf
		G		0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1
s_0	(0,0)	5	inf
6	1	inf	inf
(0,0)	5	4	2
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	2	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
6	(0,0)	5	3	3
6	1	inf	inf	inf
(0,0)	5	4	2	2
inf	inf	inf	inf	inf
4	3	2	1	1
inf	inf	inf	inf	inf
3	2	2	0	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1
s_0			
6	(0,0)	5	3
6	1	inf	inf
(0,0)	5	4	2
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	0	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3 parent( $v$ )  $\leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
s_0				
6	(0,0)	5		3
6	1	inf		inf
(0,0)	5	(1,0)	4	2
inf	inf	inf	inf	inf
4	3	2	1	
inf	inf	inf	inf	inf
3	2	inf	0	G

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

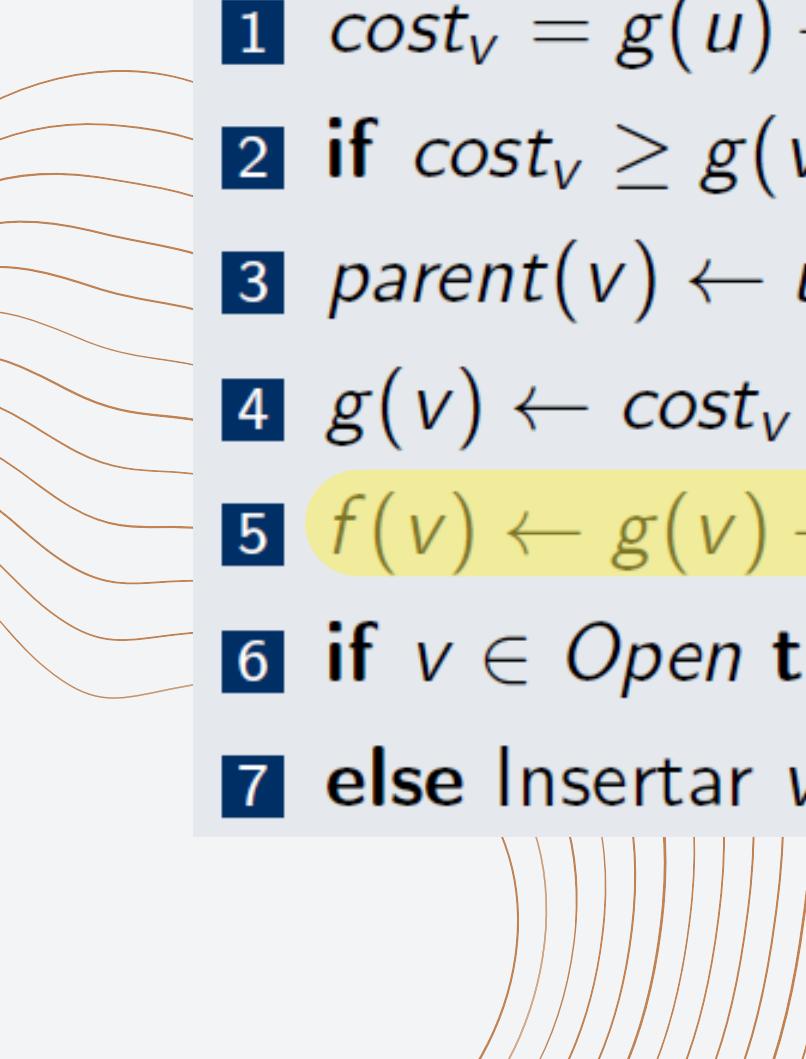
```

6	0	6	1
s_0		(0,0)	5
6	1	2	
(0,0)	5	(1,0)	4
inf	inf	inf	inf
4	3	2	1
inf	inf	inf	inf
3	2	G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1		inf
s_0		(0,0)	5		3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf		inf		inf	inf
	4		3	2	1
inf		inf		inf	inf
	3		2		0
G					

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

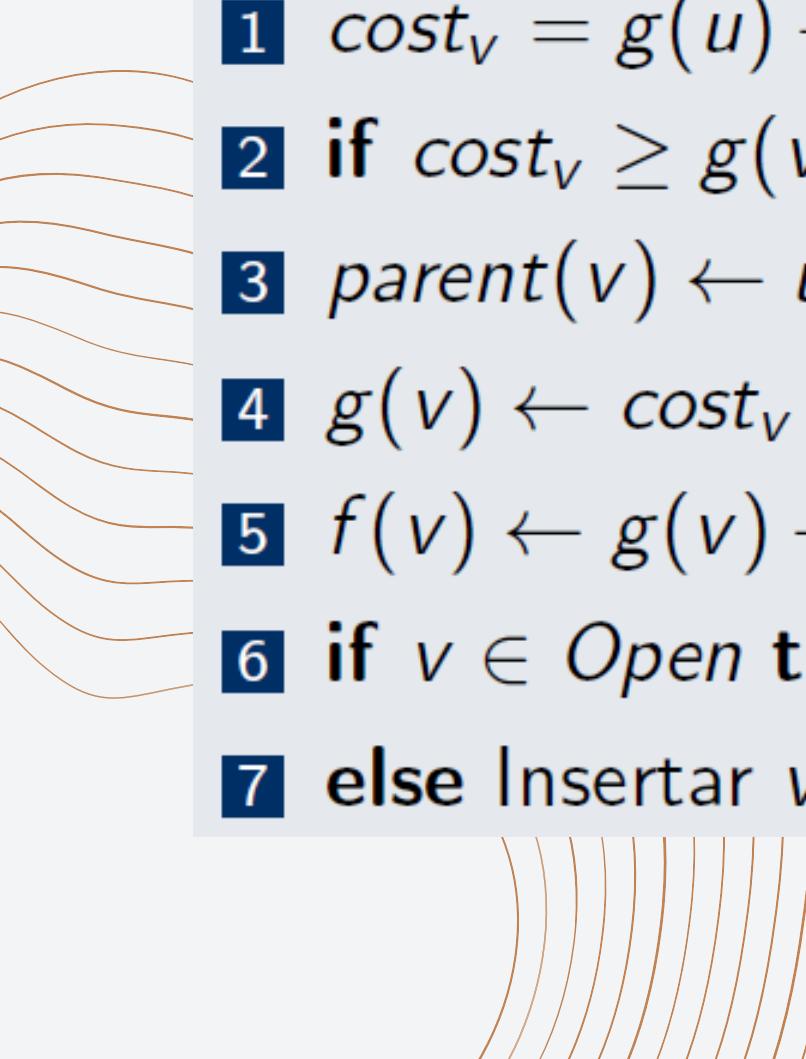
```

6	0	6	1		inf
s_0		(0,0)	5		3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	inf	inf	inf		inf
4	3	2	1		1
inf	inf	inf	inf		inf
3	2			G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf		inf		inf	inf
4		3		2	1
inf		inf		inf	inf
3		2			0
G					

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm's progress on a 4x5 grid. The grid cells are colored based on their f-values: orange for the start state s_0 , green for the current node $(0,0)$, yellow for the next node $(1,0)$, and black for nodes with f-values of infinity. The cost values are shown in the first two columns, and the f-values are shown in the last two columns. The goal state G is located at the bottom-right corner.

6	0	6	1	
s_0				inf
6	(0,0)	5		3
6	1	6	2	
(0,0)	5	(1,0)	4	inf
inf	inf	inf	inf	inf
4		3		1
inf	inf	inf		inf
3		2		0
			G	

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	inf	inf	inf	inf	inf
4	(1,1)	3	2	1	inf
inf	inf	inf	inf	inf	inf
3	2		G	0	

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

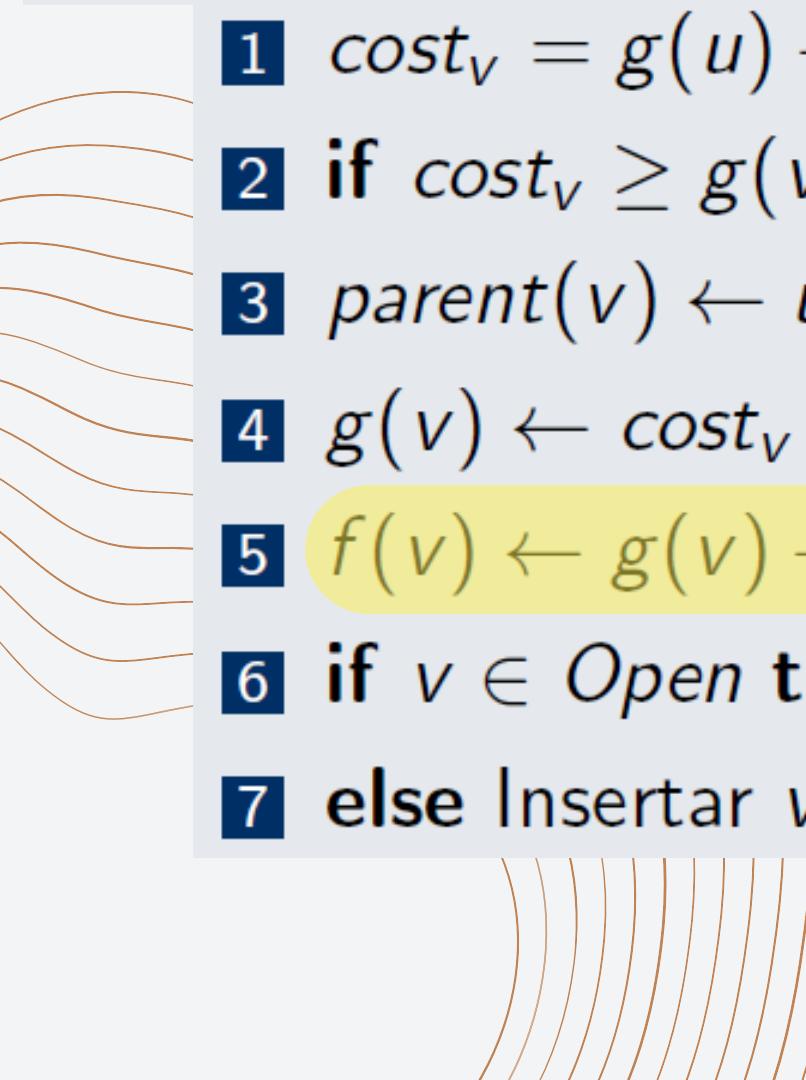
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf		3		inf	inf
4	(1,1)	3		2	1
inf		inf			inf
inf				G	0
3		2			

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	6	3		inf	inf
4	(1,1)	3		2	1
inf	inf	inf		inf	inf
3		2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	6	3		inf	inf
4	(1,1)	3		2	1
inf	inf	inf			inf
3		2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm's progress through a 4x4 grid. The grid is divided into four quadrants by thick black lines. The top-left quadrant contains the initial state s_0 with an f-value of 6. The bottom-right quadrant contains the goal state G with an f-value of 0. The other two quadrants are filled with various f-values (6, 1, 2, 3, 4, inf) and represent the open and closed sets. The open set (top-right quadrant) contains nodes $(0,0)$, $(1,0)$, and $(1,1)$. The closed set (bottom-left quadrant) contains nodes $(0,0)$, $(1,0)$, and $(1,1)$. The bottom-right quadrant shows the progression of the search, with f-values increasing from 2 to 0 as the algorithm moves towards the goal.

6	0	6	1	
s_0				inf
6	(0,0)	5		3
6	1	6	2	
(0,0)	5	(1,0)	4	2
inf	6	3		inf
4	(1,1)	3	2	1
inf	inf	inf	inf	0
3		2		G

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

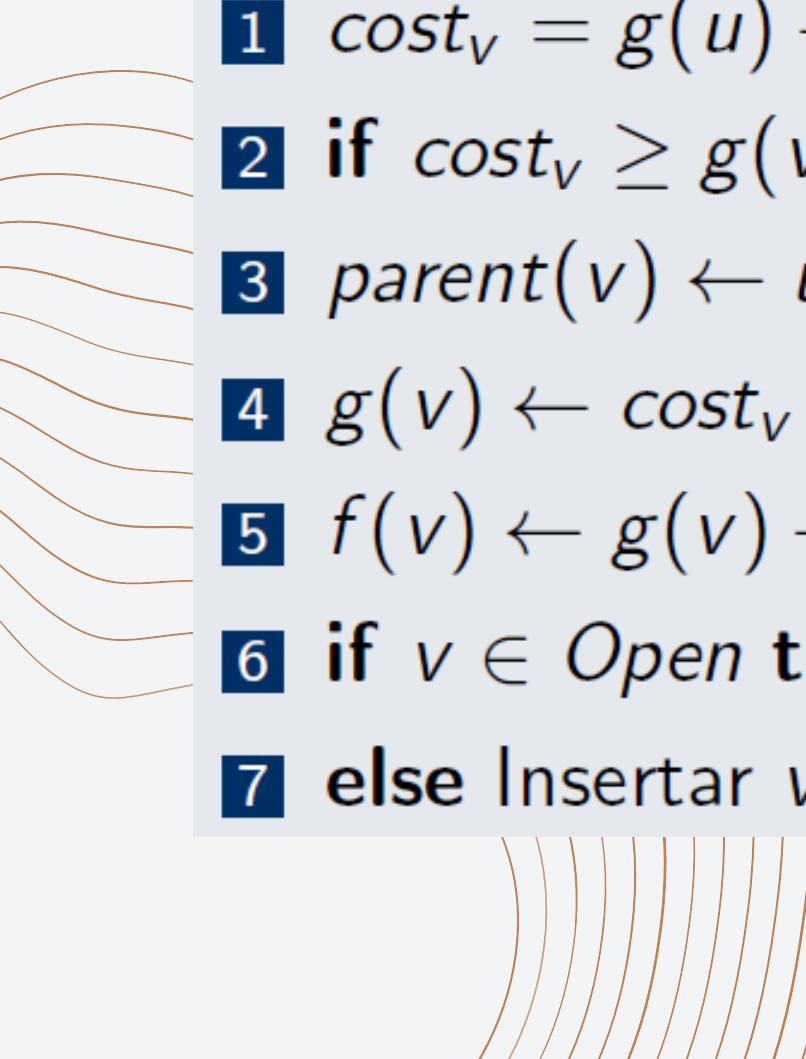
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	6	3		inf	inf
4	(1,1)	3		2	1
inf	inf	inf		inf	inf
3		2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
inf	6	3		inf	inf
4	(1,1)	3		2	1
inf	inf	inf		inf	inf
3		2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
    1  $cost_v = g(u) + c(u, v)$ 
    2 if  $cost_v \geq g(v)$  return
    3  $parent(v) \leftarrow u$ 
    4  $g(v) \leftarrow cost_v$ 
    5  $f(v) \leftarrow g(v) + h(v)$ 
    6 if  $v \in Open$  then Reordenar  $Open$ 
    7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1
s_0			
6	(0,0)	5	3
6	1	6	2
(0,0)	5	(1,0)	4
inf	6	3	inf
4	(1,1)	3	2
inf	inf	inf	inf
3	2	2	0

```
1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
```

	6	0	6	1		inf
	6	(0,0)	5			3
6	1	6	2			inf
(0,0)	5	(1,0)	4			2
inf	6	3				inf
(0,1)	4	(1,1)	3			1
inf		inf				inf
3		2				0



- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
- 5 Extrae un u desde $Open$ con menor valor
- 6 **if** u es objetivo **return** u
- 7 **for each** $v \in Succ(u)$ **do**
 - 1 $cost_v = g(u) + c(u, v)$
 - 2 **if** $cost_v \geq g(v)$ **return**
 - 3 $parent(v) \leftarrow u$
 - 4 $g(v) \leftarrow cost_v$
 - 5 $f(v) \leftarrow g(v) + h(v)$
 - 6 **if** $v \in Open$ **then** Reordenar $Open$
 - 7 **else** Insertar v en $Open$

	0	6	1		inf
6	s_o	(0,0)	5		3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
2	6	3		inf	inf
(0,1)	4	(1,1)	3		1
inf		inf		inf	
3		2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

	s_0	0	6	1		inf
6		6	(0,0)	5		3
6	1	6		2		inf
(0,0)	5	(1,0)		4		2
6	2	6	3		inf	inf
(0,1)	4	(1,1)	3		2	1
inf		inf			inf	
3		2			G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3		inf
(0,1)	4	(1,1)	3		1
inf		inf			inf
3		2			0

The diagram illustrates the A* search algorithm's execution on a grid. The grid has columns labeled 0, 1, and 2, and rows labeled 0, 1, and 2. The start node s_0 is at (0,0). The goal node G is at (1,1). The grid cells are colored based on their f-values: yellow for f=1, green for f=2, and grey for f=3. The open list $Open$ contains nodes (0,0), (1,0), (0,1), and (1,1). The closed set is shown in black. The f-values are calculated as $f(v) = g(v) + h(v)$, where $g(v)$ is the path cost from the start, and $h(v)$ is the heuristic estimate to the goal.

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3		inf
(0,1)	4	(1,1)	3		1
inf		inf			inf
3		2			0

The diagram illustrates the A* search algorithm's execution on a grid. The grid has columns labeled 0, 1, 2 and rows labeled 0, 1, 2, 3.
 - Column 0: Row 0 contains '6'. Row 1 contains 's0' in an orange circle. Row 2 contains '6'. Row 3 contains '(0,0)' and '5'.
 - Column 1: Row 0 contains '6'. Row 1 contains '1'. Row 2 contains '(0,0)' and '5'. Row 3 contains '(1,0)' and '4'.
 - Column 2: Row 0 contains '6'. Row 1 contains '2'. Row 2 contains '(1,0)' and '4'. Row 3 contains '3'.
 - Column 3: Row 0 contains '1'. Row 1 contains '2'. Row 2 contains '3'. Row 3 contains 'inf'.
 - Column 4: Row 0 contains 'inf'. Row 1 contains 'inf'. Row 2 contains '2'. Row 3 contains '0'.
 - Column 5: Row 0 contains 'inf'. Row 1 contains 'inf'. Row 2 contains '1'. Row 3 contains '0'.
 - Colored regions highlight specific states:
 - A green region covers the area from (0,0) to (1,1).
 - A yellow region covers the area from (1,0) to (1,1).
 - An orange circle labeled 'G' is at the bottom right corner (3,3).

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
s_0				
6	(0,0)	5		3
6	1	6	2	inf
(0,0)	5	(1,0)	4	2
6	2	6	3	inf
(0,1)	4	(1,1)	3	1
inf	inf	inf	inf	inf
3	2	2	1	0

- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
- 5 Extrae un u desde $Open$ con menor valor
- 6 **if** u es objetivo **return** u
- 7 **for each** $v \in Succ(u)$ **do**
 - 1 $cost_v = g(u) + c(u, v)$
 - 2 **if** $cost_v \geq g(v)$ **return**
 - 3 $parent(v) \leftarrow u$
 - 4 $g(v) \leftarrow cost_v$
 - 5 $f(v) \leftarrow g(v) + h(v)$
 - 6 **if** $v \in Open$ **then** Reordenar $Open$
 - 7 **else** Insertar v en $Open$

	6	0	6	1		inf
	6	(0,0)	5			3
6	1	6	2			inf
(0,0)	5	(1,0)	4			2
6	2	6	3		inf	inf
(0,1)	4	(1,1)	3	(1,2)	2	1
inf		inf			inf	
3		2			G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

	s_0	0	6	1		inf
6		6	(0,0)	5		3
6	1	6		2		inf
(0,0)	5	(1,0)		4		2
6	2	6	3		4	inf
(0,1)	4	(1,1)	3	(1,2)	2	1
inf		inf			inf	
3		2			G	0

- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
- 5 Extrae un u desde $Open$ con menor valor
- 6 **if** u es objetivo **return** u
- 7 **for each** $v \in Succ(u)$ **do**
 - 1 $cost_v = g(u) + c(u, v)$
 - 2 **if** $cost_v \geq g(v)$ **return**
 - 3 $parent(v) \leftarrow u$
 - 4 $g(v) \leftarrow cost_v$
 - 5 $f(v) \leftarrow g(v) + h(v)$
 - 6 **if** $v \in Open$ **then** Reordenar $Open$
 - 7 **else** Insertar v en $Open$

	s_0	0	6	1		inf
6		6	(0,0)	5		3
6	1	6		2		inf
(0,0)	5	(1,0)		4		2
6	2	6	3	6	4	inf
(0,1)	4	(1,1)	3	(1,2)	2	1
inf		inf			inf	
3		2			G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
8     1  $cost_v = g(u) + c(u, v)$ 
9     2 if  $cost_v \geq g(v)$  return
10    3  $parent(v) \leftarrow u$ 
11    4  $g(v) \leftarrow cost_v$ 
12    5  $f(v) \leftarrow g(v) + h(v)$ 
13    6 if  $v \in Open$  then Reordenar  $Open$ 
14    7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
inf		inf			1
3		2			inf
G					0

```
1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
8     1  $cost_v = g(u) + c(u, v)$ 
9     2 if  $cost_v \geq g(v)$  return
10    3  $parent(v) \leftarrow u$ 
11    4  $g(v) \leftarrow cost_v$ 
12    5  $f(v) \leftarrow g(v) + h(v)$ 
13    6 if  $v \in Open$  then Reordenar  $Open$ 
14    7 else Insertar  $v$  en  $Open$ 
```

6	0	6	1		inf
	s_o				
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3		inf
(0,1)	4	(1,1)	3		1
inf		inf			inf
3		2			0
				G	

```
1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
```

	6	0	6	1		inf
	6	(0,0)	5			3
	6	1	6	2		inf
	(0,0)	5	(1,0)	4		2
	6	2	6	3	6	inf
	(0,1)	4	(1,1)	3	(1,2)	2
	inf		inf			1
	3	(1,2)	2		G	inf
	0					0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
inf			4		1
3		(1,2)	2		inf
				G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

	6	0	6	1		inf
	6	(0,0)	5			3
6	1	6	2			inf
(0,0)	5	(1,0)	4			2
6	2	6	3	6	4	inf
(0,1)	4	(1,1)	3	(1,2)	2	1
inf	6	4				inf
3	(1,2)	2			G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
s_0				
6	(0,0)	5		3
6	1	6	2	inf
(0,0)	5	(1,0)	4	2
6	2	6	3	inf
(0,1)	4	(1,1)	3	1
inf	6	4		inf
3	(1,2)	2		0
			G	

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor de  $f(u)$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

	6	0	6	1		inf
	6	(0,0)	5			3
	6	1	6	2		inf
	(0,0)	5	(1,0)	4		2
	6	2	6	3	6	inf
	(0,1)	4	(1,1)	3	(1,2)	2
	inf	6	4			1
	3	(1,2)	2			inf
	G				0	

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
inf		6	4		1
3		(1,2)	2		inf
				G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
     1  $cost_v = g(u) + c(u, v)$ 
     2 if  $cost_v \geq g(v)$  return
     3  $parent(v) \leftarrow u$ 
     4  $g(v) \leftarrow cost_v$ 
     5  $f(v) \leftarrow g(v) + h(v)$ 
     6 if  $v \in Open$  then Reordenar  $Open$ 
     7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
inf		6	4		1
3		(1,2)	2		inf
				G	0

- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
- 5 Extrae un u desde $Open$ con menor valor
- 6 **if** u es objetivo **return** u
- 7 **for each** $v \in Succ(u)$ **do**
 - 1 $cost_v = g(u) + c(u, v)$
 - 2 **if** $cost_v \geq g(v)$ **return**
- 3 $parent(v) \leftarrow u$
- 4 $g(v) \leftarrow cost_v$
- 5 $f(v) \leftarrow g(v) + h(v)$
- 6 **if** $v \in Open$ **then** Reordenar $Open$
- 7 **else** Insertar v en $Open$

6	0	6	1		inf
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	inf
(0,1)	4	(1,1)	3	(1,2)	1
inf		6	4		inf
3		(1,2)	2		0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4		(2,2)	1
3	(1,2)	2			inf
				G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

<img alt="Diagram showing the search space with nodes s0 and G. Nodes are colored based on their f-values: orange for s0, green for (0,1), yellow for (1,2), cyan for (2,2), and black for G. Edges are labeled with costs: (s0, 0) = 6, (0, 1) = 5, (1, 2) = 4, (2, 2) = 1, (0, 2) = 6, (1, 3) = 3, (2, 4) = 2, (0, 3) = 6, (1, 4) = 4, (2, 5) = 1, (0, 4) = 6, (1, 5) = 4, (2, 6) = 2, (0, 5) = 6, (1, 6) = 4, (2, 7) = 1, (0, 6) = 6, (1, 7) = 4, (2, 8) = 2, (0, 7) = 6, (1, 8) = 4, (2, 9) = 1, (0, 8) = 6, (1, 9) = 4, (2, 10) = 2, (0, 9) = 6, (1, 10) = 4, (2, 11) = 1, (0, 10) = 6, (1, 11) = 4, (2, 12) = 2, (0, 11) = 6, (1, 12) = 4, (2, 13) = 1, (0, 12) = 6, (1, 13) = 4, (2, 14) = 2, (0, 13) = 6, (1, 14) = 4, (2, 15) = 1, (0, 14) = 6, (1, 15) = 4, (2, 16) = 2, (0, 15) = 6, (1, 16) = 4, (2, 17) = 1, (0, 16) = 6, (1, 17) = 4, (2, 18) = 2, (0, 17) = 6, (1, 18) = 4, (2, 19) = 1, (0, 18) = 6, (1, 19) = 4, (2, 20) = 2, (0, 19) = 6, (1, 20) = 4, (2, 21) = 1, (0, 20) = 6, (1, 21) = 4, (2, 22) = 2, (0, 21) = 6, (1, 22) = 4, (2, 23) = 1, (0, 22) = 6, (1, 23) = 4, (2, 24) = 2, (0, 23) = 6, (1, 24) = 4, (2, 25) = 1, (0, 24) = 6, (1, 25) = 4, (2, 26) = 2, (0, 25) = 6, (1, 26) = 4, (2, 27) = 1, (0, 26) = 6, (1, 27) = 4, (2, 28) = 2, (0, 27) = 6, (1, 28) = 4, (2, 29) = 1, (0, 28) = 6, (1, 29) = 4, (2, 30) = 2, (0, 29) = 6, (1, 30) = 4, (2, 31) = 1, (0, 30) = 6, (1, 31) = 4, (2, 32) = 2, (0, 31) = 6, (1, 32) = 4, (2, 33) = 1, (0, 32) = 6, (1, 33) = 4, (2, 34) = 2, (0, 33) = 6, (1, 34) = 4, (2, 35) = 1, (0, 34) = 6, (1, 35) = 4, (2, 36) = 2, (0, 35) = 6, (1, 36) = 4, (2, 37) = 1, (0, 36) = 6, (1, 37) = 4, (2, 38) = 2, (0, 37) = 6, (1, 38) = 4, (2, 39) = 1, (0, 38) = 6, (1, 39) = 4, (2, 40) = 2, (0, 39) = 6, (1, 40) = 4, (2, 41) = 1, (0, 40) = 6, (1, 41) = 4, (2, 42) = 2, (0, 41) = 6, (1, 42) = 4, (2, 43) = 1, (0, 42) = 6, (1, 43) = 4, (2, 44) = 2, (0, 43) = 6, (1, 44) = 4, (2, 45) = 1, (0, 44) = 6, (1, 45) = 4, (2, 46) = 2, (0, 45) = 6, (1, 46) = 4, (2, 47) = 1, (0, 46) = 6, (1, 47) = 4, (2, 48) = 2, (0, 47) = 6, (1, 48) = 4, (2, 49) = 1, (0, 48) = 6, (1, 49) = 4, (2, 50) = 2, (0, 49) = 6, (1, 50) = 4, (2, 51) = 1, (0, 50) = 6, (1, 51) = 4, (2, 52) = 2, (0, 51) = 6, (1, 52) = 4, (2, 53) = 1, (0, 52) = 6, (1, 53) = 4, (2, 54) = 2, (0, 53) = 6, (1, 54) = 4, (2, 55) = 1, (0, 54) = 6, (1, 55) = 4, (2, 56) = 2, (0, 55) = 6, (1, 56) = 4, (2, 57) = 1, (0, 56) = 6, (1, 57) = 4, (2, 58) = 2, (0, 57) = 6, (1, 58) = 4, (2, 59) = 1, (0, 58) = 6, (1, 59) = 4, (2, 60) = 2, (0, 59) = 6, (1, 60) = 4, (2, 61) = 1, (0, 60) = 6, (1, 61) = 4, (2, 62) = 2, (0, 61) = 6, (1, 62) = 4, (2, 63) = 1, (0, 62) = 6, (1, 63) = 4, (2, 64) = 2, (0, 63) = 6, (1, 64) = 4, (2, 65) = 1, (0, 64) = 6, (1, 65) = 4, (2, 66) = 2, (0, 65) = 6, (1, 66) = 4, (2, 67) = 1, (0, 66) = 6, (1, 67) = 4, (2, 68) = 2, (0, 67) = 6, (1, 68) = 4, (2, 69) = 1, (0, 68) = 6, (1, 69) = 4, (2, 70) = 2, (0, 69) = 6, (1, 70) = 4, (2, 71) = 1, (0, 70) = 6, (1, 71) = 4, (2, 72) = 2, (0, 71) = 6, (1, 72) = 4, (2, 73) = 1, (0, 72) = 6, (1, 73) = 4, (2, 74) = 2, (0, 73) = 6, (1, 74) = 4, (2, 75) = 1, (0, 74) = 6, (1, 75) = 4, (2, 76) = 2, (0, 75) = 6, (1, 76) = 4, (2, 77) = 1, (0, 76) = 6, (1, 77) = 4, (2, 78) = 2, (0, 77) = 6, (1, 78) = 4, (2, 79) = 1, (0, 78) = 6, (1, 79) = 4, (2, 80) = 2, (0, 79) = 6, (1, 80) = 4, (2, 81) = 1, (0, 80) = 6, (1, 81) = 4, (2, 82) = 2, (0, 81) = 6, (1, 82) = 4, (2, 83) = 1, (0, 82) = 6, (1, 83) = 4, (2, 84) = 2, (0, 83) = 6, (1, 84) = 4, (2, 85) = 1, (0, 84) = 6, (1, 85) = 4, (2, 86) = 2, (0, 85) = 6, (1, 86) = 4, (2, 87) = 1, (0, 86) = 6, (1, 87) = 4, (2, 88) = 2, (0, 87) = 6, (1, 88) = 4, (2, 89) = 1, (0, 88) = 6, (1, 89) = 4, (2, 90) = 2, (0, 89) = 6, (1, 90) = 4, (2, 91) = 1, (0, 90) = 6, (1, 91) = 4, (2, 92) = 2, (0, 91) = 6, (1, 92) = 4, (2, 93) = 1, (0, 92) = 6, (1, 93) = 4, (2, 94) = 2, (0, 93) = 6, (1, 94) = 4, (2, 95) = 1, (0, 94) = 6, (1, 95) = 4, (2, 96) = 2, (0, 95) = 6, (1, 96) = 4, (2, 97) = 1, (0, 96) = 6, (1, 97) = 4, (2, 98) = 2, (0, 97) = 6, (1, 98) = 4, (2, 99) = 1, (0, 98) = 6, (1, 99) = 4, (2, 100) = 2, (0, 99) = 6, (1, 100) = 4, (2, 101) = 1, (0, 100) = 6, (1, 101) = 4, (2, 102) = 2, (0, 101) = 6, (1, 102) = 4, (2, 103) = 1, (0, 102) = 6, (1, 103) = 4, (2, 104) = 2, (0, 103) = 6, (1, 104) = 4, (2, 105) = 1, (0, 104) = 6, (1, 105) = 4, (2, 106) = 2, (0, 105) = 6, (1, 106) = 4, (2, 107) = 1, (0, 106) = 6, (1, 107) = 4, (2, 108) = 2, (0, 107) = 6, (1, 108) = 4, (2, 109) = 1, (0, 108) = 6, (1, 109) = 4, (2, 110) = 2, (0, 109) = 6, (1, 110) = 4, (2, 111) = 1, (0, 110) = 6, (1, 111) = 4, (2, 112) = 2, (0, 111) = 6, (1, 112) = 4, (2, 113) = 1, (0, 112) = 6, (1, 113) = 4, (2, 114) = 2, (0, 113) = 6, (1, 114) = 4, (2, 115) = 1, (0, 114) = 6, (1, 115) = 4, (2, 116) = 2, (0, 115) = 6, (1, 116) = 4, (2, 117) = 1, (0, 116) = 6, (1, 117) = 4, (2, 118) = 2, (0, 117) = 6, (1, 118) = 4, (2, 119) = 1, (0, 118) = 6, (1, 119) = 4, (2, 120) = 2, (0, 119) = 6, (1, 120) = 4, (2, 121) = 1, (0, 120) = 6, (1, 121) = 4, (2, 122) = 2, (0, 121) = 6, (1, 122) = 4, (2, 123) = 1, (0, 122) = 6, (1, 123) = 4, (2, 124) = 2, (0, 123) = 6, (1, 124) = 4, (2, 125) = 1, (0, 124) = 6, (1, 125) = 4, (2, 126) = 2, (0, 125) = 6, (1, 126) = 4, (2, 127) = 1, (0, 126) = 6, (1, 127) = 4, (2, 128) = 2, (0, 127) = 6, (1, 128) = 4, (2, 129) = 1, (0, 128) = 6, (1, 129) = 4, (2, 130) = 2, (0, 129) = 6, (1, 130) = 4, (2, 131) = 1, (0, 130) = 6, (1, 131) = 4, (2, 132) = 2, (0, 131) = 6, (1, 132) = 4, (2, 133) = 1, (0, 132) = 6, (1, 133) = 4, (2, 134) = 2, (0, 133) = 6, (1, 134) = 4, (2, 135) = 1, (0, 134) = 6, (1, 135) = 4, (2, 136) = 2, (0, 135) = 6, (1, 136) = 4, (2, 137) = 1, (0, 136) = 6, (1, 137) = 4, (2, 138) = 2, (0, 137) = 6, (1, 138) = 4, (2, 139) = 1, (0, 138) = 6, (1, 139) = 4, (2, 140) = 2, (0, 139) = 6, (1, 140) = 4, (2, 141) = 1, (0, 140) = 6, (1, 141) = 4, (2, 142) = 2, (0, 141) = 6, (1, 142) = 4, (2, 143) = 1, (0, 142) = 6, (1, 143) = 4, (2, 144) = 2, (0, 143) = 6, (1, 144) = 4, (2, 145) = 1, (0, 144) = 6, (1, 145) = 4, (2, 146) = 2, (0, 145) = 6, (1, 146) = 4, (2, 147) = 1, (0, 146) = 6, (1, 147) = 4, (2, 148) = 2, (0, 147) = 6, (1, 148) = 4, (2, 149) = 1, (0, 148) = 6, (1, 149) = 4, (2, 150) = 2, (0, 149) = 6, (1, 150) = 4, (2, 151) = 1, (0, 150) = 6, (1, 151) = 4, (2, 152) = 2, (0, 151) = 6, (1, 152) = 4, (2, 153) = 1, (0, 152) = 6, (1, 153) = 4, (2, 154) = 2, (0, 153) = 6, (1, 154) = 4, (2, 155) = 1, (0, 154) = 6, (1, 155) = 4, (2, 156) = 2, (0, 155) = 6, (1, 156) = 4, (2, 157) = 1, (0, 156) = 6, (1, 157) = 4, (2, 158) = 2, (0, 157) = 6, (1, 158) = 4, (2, 159) = 1, (0, 158) = 6, (1, 159) = 4, (2, 160) = 2, (0, 159) = 6, (1, 160) = 4, (2, 161) = 1, (0, 160) = 6, (1, 161) = 4, (2, 162) = 2, (0, 161) = 6, (1, 162) = 4, (2, 163) = 1, (0, 162) = 6, (1, 163) = 4, (2, 164) = 2, (0, 163) = 6, (1, 164) = 4, (2, 165) = 1, (0, 164) = 6, (1, 165) = 4, (2, 166) = 2, (0, 165) = 6, (1, 166) = 4, (2, 167) = 1, (0, 166) = 6, (1, 167) = 4, (2, 168) = 2, (0, 167) = 6, (1, 168) = 4, (2, 169) = 1, (0, 168) = 6, (1, 169) = 4, (2, 170) = 2, (0, 169) = 6, (1, 170) = 4, (2, 171) = 1, (0, 170) = 6, (1, 171) = 4, (2, 172) = 2, (0, 171) = 6, (1, 172) = 4, (2, 173) = 1, (0, 172) = 6, (1, 173) = 4, (2, 174) = 2, (0, 173) = 6, (1, 174) = 4, (2, 175) = 1, (0, 174) = 6, (1, 175) = 4, (2, 176) = 2, (0, 175) = 6, (1, 176) = 4, (2, 177) = 1, (0, 176) = 6, (1, 177) = 4, (2, 178) = 2, (0, 177) = 6, (1, 178) = 4, (2, 179) = 1, (0, 178) = 6, (1, 179) = 4, (2, 180) = 2, (0, 179) = 6, (1, 180) = 4, (2, 181) = 1, (0, 180) = 6, (1, 181) = 4, (2, 182) = 2, (0, 181) = 6, (1, 182) = 4, (2, 183) = 1, (0, 182) = 6, (1, 183) = 4, (2, 184) = 2, (0, 183) = 6, (1, 184) = 4, (2, 185) = 1, (0, 184) = 6, (1, 185) = 4, (2, 186) = 2, (0, 185) = 6, (1, 186) = 4, (2, 187) = 1, (0, 186) = 6, (1, 187) = 4, (2, 188) = 2, (0, 187) = 6, (1, 188) = 4, (2, 189) = 1, (0, 188) = 6, (1, 189) = 4, (2, 190) = 2, (0, 189) = 6, (1, 190) = 4, (2, 191) = 1, (0, 190) = 6, (1, 191) = 4, (2, 192) = 2, (0, 191) = 6, (1, 192) = 4, (2, 193) = 1, (0, 192) = 6, (1, 193) = 4, (2, 194) = 2, (0, 193) = 6, (1, 194) = 4, (2, 195) = 1, (0, 194) = 6, (1, 195) = 4, (2, 196) = 2, (0, 195) = 6, (1, 196) = 4, (2, 197) = 1, (0, 196) = 6, (1, 197) = 4, (2, 198) = 2, (0, 197) = 6, (1, 198) = 4, (2, 199) = 1, (0, 198) = 6, (1, 199) = 4, (2, 200) = 2, (0, 199) = 6, (1, 200) = 4, (2, 201) = 1, (0, 200) = 6, (1, 201) = 4, (2, 202) = 2, (0, 201) = 6, (1, 202) = 4, (2, 203) = 1, (0, 202) = 6, (1, 203) = 4, (2, 204) = 2, (0, 203) = 6, (1, 204) = 4, (2, 205) = 1, (0, 204) = 6, (1, 205) = 4, (2, 206) = 2, (0, 205) = 6, (1, 206) = 4, (2, 207) = 1, (0, 206) = 6, (1, 207) = 4, (2, 208) = 2, (0, 207) = 6, (1, 208) = 4, (2, 209) = 1, (0, 208) = 6, (1, 209) = 4, (2, 210) = 2, (0, 209) = 6, (1, 210) = 4, (2, 211) = 1, (0, 210) = 6, (1, 211) = 4, (2, 212) = 2, (0, 211) = 6, (1, 212) = 4, (2, 213) = 1, (0, 212) = 6, (1, 213) =

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			inf
3	(1,2)	2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm on a grid. The grid has 6 columns and 6 rows. The start node s_0 is at (0,0). The goal node G is at (2,2). The grid shows the following values:

	0	1	2	3	4	5	6	
0	$g(s_0) = 0$	$g(1,0) = 5$	$g(2,0) = 4$	$g(3,0) = 3$	$g(4,0) = 2$	$g(5,0) = 1$	$g(6,0) = \infty$	$g(7,0) = \infty$
1	$g(0,1) = 4$	$g(1,1) = 3$	$g(2,1) = 2$	$g(3,1) = 1$	$g(4,1) = \infty$	$g(5,1) = \infty$	$g(6,1) = \infty$	$g(7,1) = \infty$
2	$g(0,2) = 5$	$g(1,2) = 4$	$g(2,2) = 2$	$g(3,2) = \infty$	$g(4,2) = \infty$	$g(5,2) = \infty$	$g(6,2) = \infty$	$g(7,2) = \infty$
3	$g(0,3) = 6$	$g(1,3) = 5$	$g(2,3) = 4$	$g(3,3) = 3$	$g(4,3) = 2$	$g(5,3) = 1$	$g(6,3) = \infty$	$g(7,3) = \infty$
4	$g(0,4) = 7$	$g(1,4) = 6$	$g(2,4) = 5$	$g(3,4) = 4$	$g(4,4) = 3$	$g(5,4) = 2$	$g(6,4) = 1$	$g(7,4) = \infty$
5	$g(0,5) = 8$	$g(1,5) = 7$	$g(2,5) = 6$	$g(3,5) = 5$	$g(4,5) = 4$	$g(5,5) = 3$	$g(6,5) = 2$	$g(7,5) = 1$
6	$g(0,6) = 9$	$g(1,6) = 8$	$g(2,6) = 7$	$g(3,6) = 6$	$g(4,6) = 5$	$g(5,6) = 4$	$g(6,6) = 3$	$g(7,6) = \infty$
7	$g(0,7) = 10$	$g(1,7) = 9$	$g(2,7) = 8$	$g(3,7) = 7$	$g(4,7) = 6$	$g(5,7) = 5$	$g(6,7) = 4$	$g(7,7) = \infty$

The open list (green cells) contains nodes (0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1), (4,0), (4,1), (5,0), (5,1), (6,0), (6,1), (7,0), (7,1), (0,2), (1,2), (2,2), (3,2), (4,2), (5,2), (6,2), (7,2), (0,3), (1,3), (2,3), (3,3), (4,3), (5,3), (6,3), (7,3), (0,4), (1,4), (2,4), (3,4), (4,4), (5,4), (6,4), (7,4), (0,5), (1,5), (2,5), (3,5), (4,5), (5,5), (6,5), (7,5), (0,6), (1,6), (2,6), (3,6), (4,6), (5,6), (6,6), (7,6), (0,7), (1,7), (2,7), (3,7), (4,7), (5,7), (6,7), (7,7).

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		inf
(0,0)	5	(1,0)	4		2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4		(2,2)	1
3	(1,2)	2			inf
				G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm's progress through a 4x4 grid. The grid is organized into columns:

- Column 1:** Contains states and values. The first row is labeled s_0 . The second row is labeled $(0,0)$. The third row is labeled $(0,1)$. The fourth row is labeled inf . The fifth row is labeled G .
- Column 2:** Contains values from the function g . The first row is labeled 0. The second row is labeled 5. The third row is labeled 4. The fourth row is labeled 6. The fifth row is labeled 2.
- Column 3:** Contains values from the function f . The first row is labeled 6. The second row is labeled 2. The third row is labeled 3. The fourth row is labeled 4. The fifth row is labeled 2.
- Column 4:** Contains values from the function h . The first row is labeled 1. The second row is labeled 5. The third row is labeled 4. The fourth row is labeled 3. The fifth row is labeled 1.
- Column 5:** Contains values from the function g . The first row is labeled 6. The second row is labeled 4. The third row is labeled 3. The fourth row is labeled 2. The fifth row is labeled 0.
- Column 6:** Contains values from the function h . The first row is labeled 6. The second row is labeled 5. The third row is labeled 4. The fourth row is labeled 3. The fifth row is labeled 2.
- Column 7:** Contains values from the function h . The first row is labeled inf . The second row is labeled 3. The third row is labeled inf . The fourth row is labeled 2. The fifth row is labeled 0.

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3 parent(v)  $\leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1			inf
s_0						
6	(0,0)	5				3
6	1	6	2			inf
(0,0)	5	(1,0)	4			(3,2) 2
6	2	6	3	6	4	6 5
(0,1)	4	(1,1)	3	(1,2)	2	(2,2) 1
inf	6	4				inf
3	(1,2)	2				G 0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm's progress on a 4x4 grid. The grid is divided into four quadrants by color: top-left (grey), top-right (black), bottom-left (green), and bottom-right (yellow). The start node s_0 is located at index 0 in the top-left quadrant. The goal node G is located at index 0 in the bottom-right quadrant. The grid contains numerical values representing f-values: 6, 0, 6, 1; 6, (0,0), 5; (0,0), 5, 6, 2; (0,0), 5, (1,0), 4; (1,0), 3, 6, 3; 6, 2, (0,1), 4; (0,1), 4, (1,1), 3; (1,1), 3, (1,2), 2; (1,2), 2, (2,2), 1; inf, 3, (1,2), 2; (1,2), 2, inf, G; inf, 0. The black quadrant represents nodes yet to be expanded, while the other quadrants represent nodes already processed or expanded.

6	0	6	1					inf
s_0								
6	(0,0)	5						3
6	1	6	2					6
(0,0)	5	(1,0)	4					(3,2) 2
6	2	6	3	6	4	6	5	
(0,1)	4	(1,1)	3	(1,2)	2	(2,2)	1	
inf	3	(1,2)	2	inf				0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
8     1  $cost_v = g(u) + c(u, v)$ 
9     2 if  $cost_v \geq g(v)$  return
10    3  $parent(v) \leftarrow u$ 
11    4  $g(v) \leftarrow cost_v$ 
12    5  $f(v) \leftarrow g(v) + h(v)$ 
13    6 if  $v \in Open$  then Reordenar  $Open$ 
14    7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		8
(0,0)	5	(1,0)	4		6
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			inf
3	(1,2)	2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
s_0				
6	(0,0)	5		3
6	1	6	2	8
(0,0)	5	(1,0)	4	6
6	2	6	3	5
(0,1)	4	(1,1)	3	2
inf	6	4		inf
3	(1,2)	2		0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
     1  $cost_v = g(u) + c(u, v)$ 
     2 if  $cost_v \geq g(v)$  return
     3  $parent(v) \leftarrow u$ 
     4  $g(v) \leftarrow cost_v$ 
     5  $f(v) \leftarrow g(v) + h(v)$ 
     6 if  $v \in Open$  then Reordenar  $Open$ 
     7 else Insertar  $v$  en  $Open$ 

```

The diagram illustrates the A* search algorithm's progress through a grid. The grid is 4 columns wide and 8 rows high. The first column is labeled with index 0 and contains the start node s_0 . The last column is labeled with index 0 and contains the goal node G . The first row is labeled with index 6 and contains values 0, 6, 1. The second row is labeled with index 6 and contains values (0,0), 5, 3. The third row is labeled with index 6 and contains values 1, 6, 2. The fourth row is labeled with index (0,0) and contains values 5, (1,0), 4, (3,2), 2. The fifth row is labeled with index 6 and contains values 2, 6, 3, 6, 4. The sixth row is labeled with index (0,1) and contains values 4, (1,1), 3, (1,2), 2, (2,2), 1. The seventh row is labeled with index inf and contains values 6, 4. The eighth row is labeled with index 3 and contains values (1,2), 2, inf, 0.

6	0	6	1				inf
s_0							
6	(0,0)	5	3				
6	1	6	2				
(0,0)	5	(1,0)	4				
6	2	6	3	6	4	6	5
(0,1)	4	(1,1)	3	(1,2)	2	(2,2)	1
inf	6	4					
3	(1,2)	2	inf				0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		8
(0,0)	5	(1,0)	4	(3,2)	2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			inf
3	(1,2)	2		G	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

The diagram illustrates the search space and the state of the open list at different stages. It consists of a grid of cells, some of which are shaded in different colors (grey, black, green, yellow, cyan) and contain numerical values or labels. A large orange circle labeled 's0' is positioned in the top-left corner of the first cell. A large orange circle labeled 'G' is positioned in the bottom-right corner of the last cell. The grid is organized into four columns and four rows.

6	0	6	1		inf
6	(0,0)	5			3
6	1	6	2		8
					6
(0,0)	5	(1,0)	4		(3,2)
					2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
					(2,2)
inf	6	4			6
3	(1,2)	2			0

6	0	6	1		inf
6	(0,0)	5			3
6	1	6	2		8
					6
(0,0)	5	(1,0)	4		(3,2)
					2
6	2	6	3	6	4
(0,1)	4	(1,1)	3	(1,2)	2
					(2,2)
inf	6	4			6
3	(1,2)	2			0



- 1 **for each** $s \in S$ **do** $g(s) \leftarrow \infty$
- 2 $Open \leftarrow \{s_0\}$
- 3 $g(s_0) \leftarrow 0; f(s_0) \leftarrow h(s_0)$
- 4 **while** $Open \neq \emptyset$
- 5 Extrae un u desde $Open$ con menor valor
- 6 **if** u es objetivo **return** u
- 7 **for each** $v \in Succ(u)$ **do**
 - 1 $cost_v = g(u) + c(u, v)$
 - 2 **if** $cost_v \geq g(v)$ **return**
 - 3 $parent(v) \leftarrow u$
 - 4 $g(v) \leftarrow cost_v$
 - 5 $f(v) \leftarrow g(v) + h(v)$
 - 6 **if** $v \in Open$ **then** Reordenar $Open$
 - 7 **else** Insertar v en $Open$

	0	6	1		inf
6	s_o	(0,0)	5		3
6	1	6	2		6
(0,0)	5	(1,0)	4	(3,2)	2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4		6	6
3	(1,2)	2		(3,2)	0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1	inf
s_0				
6	(0,0)	5		3
6	1	6	2	8
(0,0)	5	(1,0)	4	6
6	2	6	3	5
(0,1)	4	(1,1)	3	(2,2)
inf	6	4	2	1
3	(1,2)	2		
6	6	G	0	
(3,2)				

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
8     1  $cost_v = g(u) + c(u, v)$ 
9     2 if  $cost_v \geq g(v)$  return
10    3  $parent(v) \leftarrow u$ 
11    4  $g(v) \leftarrow cost_v$ 
12    5  $f(v) \leftarrow g(v) + h(v)$ 
13    6 if  $v \in Open$  then Reordenar  $Open$ 
14    7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		8
(0,0)	5	(1,0)	4		6
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			1
3	(1,2)	2			
6	6			G	0
(3,2)					

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```

6	0	6	1		inf
s_0					
6	(0,0)	5			3
6	1	6	2		8
(0,0)	5	(1,0)	4		6
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			1
3	(1,2)	2			
6	6			G	0
(3,2)					

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
5   Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
6   if  $u$  es objetivo return  $u$ 
7   for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 
    
```

The diagram shows a 4x4 grid representing a search space. The grid cells are colored based on their f-values:

- Black Cells:** (0,0), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2), (3,0), (3,1), (3,2).
- Grey Cells:** (0,1), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2), (3,0), (3,1), (3,2).
- Green Cells:** (0,0), (0,1), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2).
- White Cell:** (3,2).
- Orange Circle:** Located at (0,0), labeled s_0 .
- Orange Circle:** Located at (3,2), labeled G .

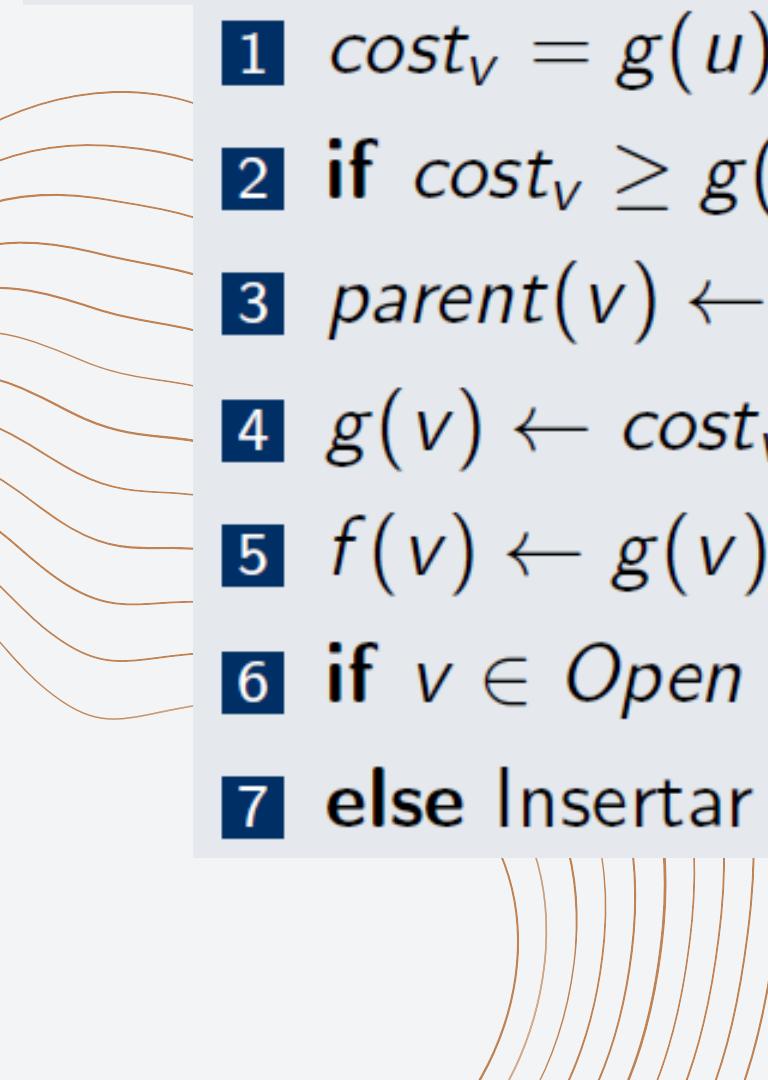
A red arrow points from the cell (2,2) up to the cell (3,2), indicating the direction of the search path.

6	0	6	1		inf
(0,0)	5				3
6	1	6	2		6
(0,0)	5	(1,0)	4		2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4			6
3	(1,2)	2			0

```

1 for each  $s \in S$  do  $g(s) \leftarrow \infty$ 
2  $Open \leftarrow \{s_0\}$ 
3  $g(s_0) \leftarrow 0$ ;  $f(s_0) \leftarrow h(s_0)$ 
4 while  $Open \neq \emptyset$ 
    5 Extrae un  $u$  desde  $Open$  con menor valor- $f$ 
    6 if  $u$  es objetivo return  $u$ 
    7 for each  $v \in Succ(u)$  do
        1  $cost_v = g(u) + c(u, v)$ 
        2 if  $cost_v \geq g(v)$  return
        3  $parent(v) \leftarrow u$ 
        4  $g(v) \leftarrow cost_v$ 
        5  $f(v) \leftarrow g(v) + h(v)$ 
        6 if  $v \in Open$  then Reordenar  $Open$ 
        7 else Insertar  $v$  en  $Open$ 

```



6	0	6	1		inf
6	(0,0)	5			3
6	1	6	2		6
(0,0)	5	(1,0)	4		(3,2) 2
6	2	6	3	6	5
(0,1)	4	(1,1)	3	(1,2)	2
inf	6	4		(2,2)	1
3	(1,2)	2		6	6
				G	0



OPTIMALIDAD

ADMISIBILIDAD

Una heurística $h(s)$ es **admissible si no **sobreestima** el costo real para llegar al objetivo, es decir**

$$h(s) \leq h^*(s)$$

ADMISIBILIDAD

I	G=1		
F			

	G=1	H=7	
	H=3		

	G=1	H=8	
	H=9		

CONSISTENCIA

Una heurística es *consistente* si el valor de la heurística en un nodo *s* debe ser menor o igual que la *suma* del valor de la heurística de sus vecinos y el costo de llegar a cada uno de ellos.

$$h(s) = 0, \forall s \text{ en } G$$

$$h(s) \leq c(s,s') + h(s'), \text{ para todo vecinos de } s.$$

¿Por qué A* es óptimo? - YouTube

https://www.youtube.com/watch?v=_41v4l5GTNc