

## Depth-first tree search

Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

## Depth-first tree search

Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

Estrategia: expandir el **nodo más profundo**

Esto se consigue con  **$f(n) = -\text{level}(n)$**

# Depth-first tree search

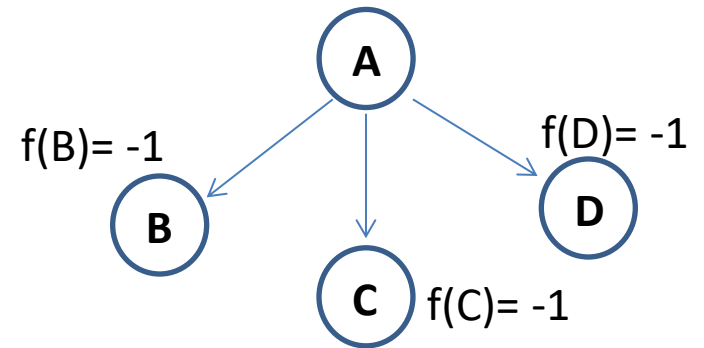
Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

Estrategia: expandir el **nodo más profundo**

Esto se consigue con  $f(n) = -\text{level}(n)$



# Depth-first tree search

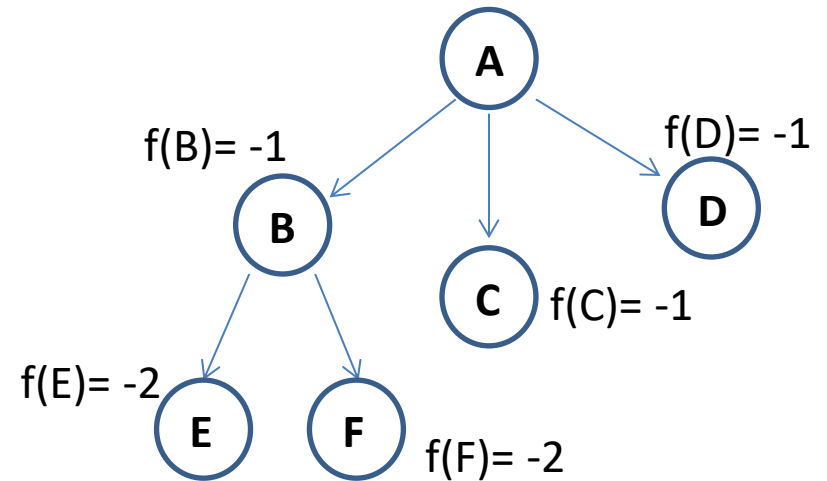
Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

Estrategia: expandir el **nodo más profundo**

Esto se consigue con  $f(n) = -\text{level}(n)$



# Depth-first tree search

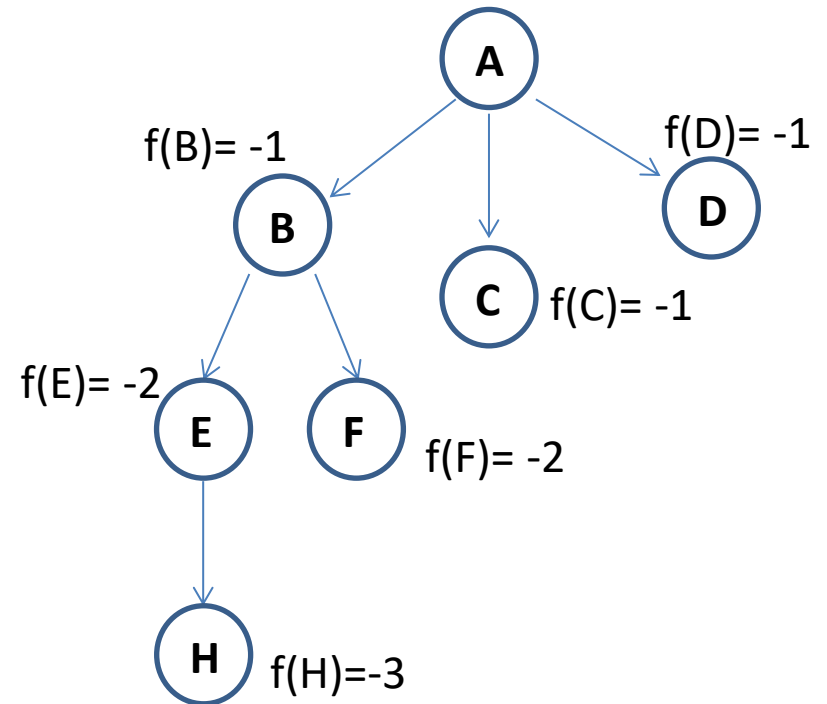
Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

Estrategia: expandir el **nodo más profundo**

Esto se consigue con  $f(n) = -\text{level}(n)$



# Depth-first tree search

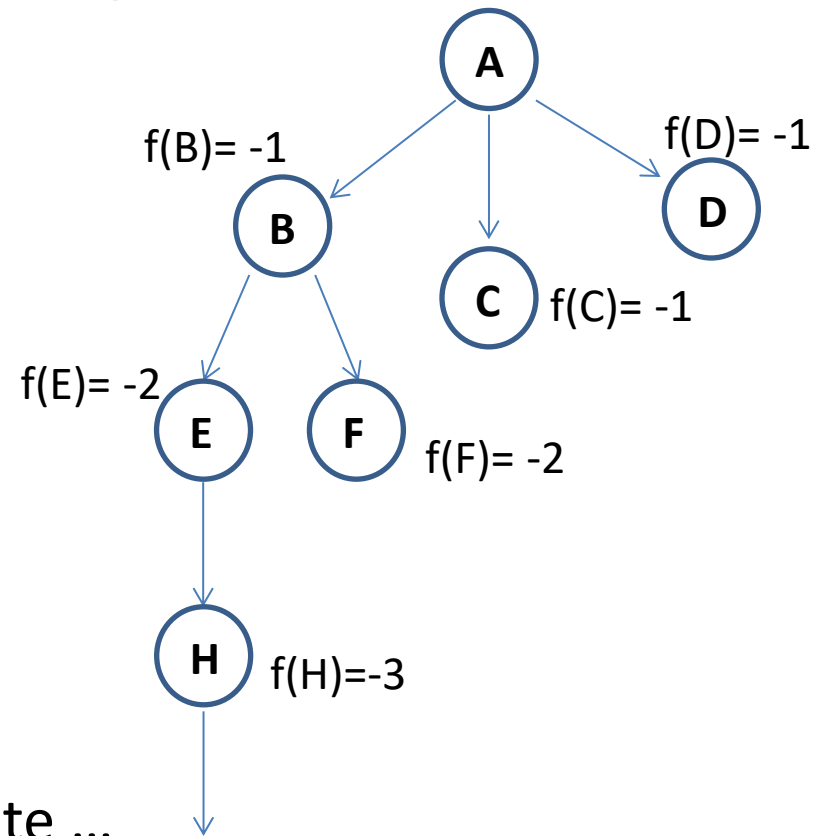
Es un poco diferente ...

No hay **CLOSED list** ... esa es la razón de que sea un **tree search**

Se mantiene en su lugar una **PATH list** que almacena solo los nodos del camino actual y se eliminan cuando se hace Backtracking

Estrategia: expandir el **nodo más profundo**

Esto se consigue con  $f(n) = -\text{level}(n)$

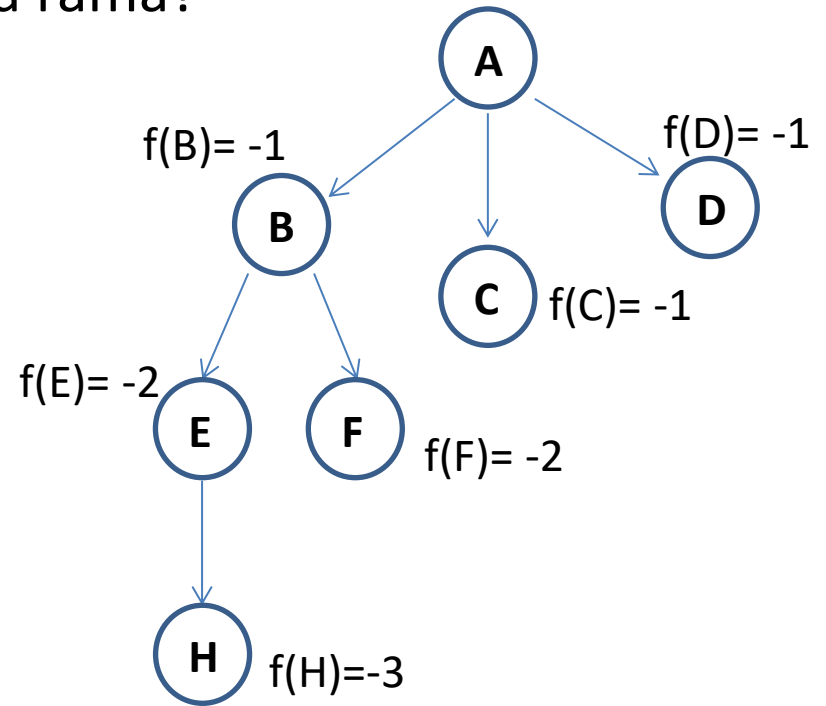


Y podemos continuar indefinidamente ...

## Depth-first tree search

¿Cuándo parar la expansión de una rama?

Cuando ...

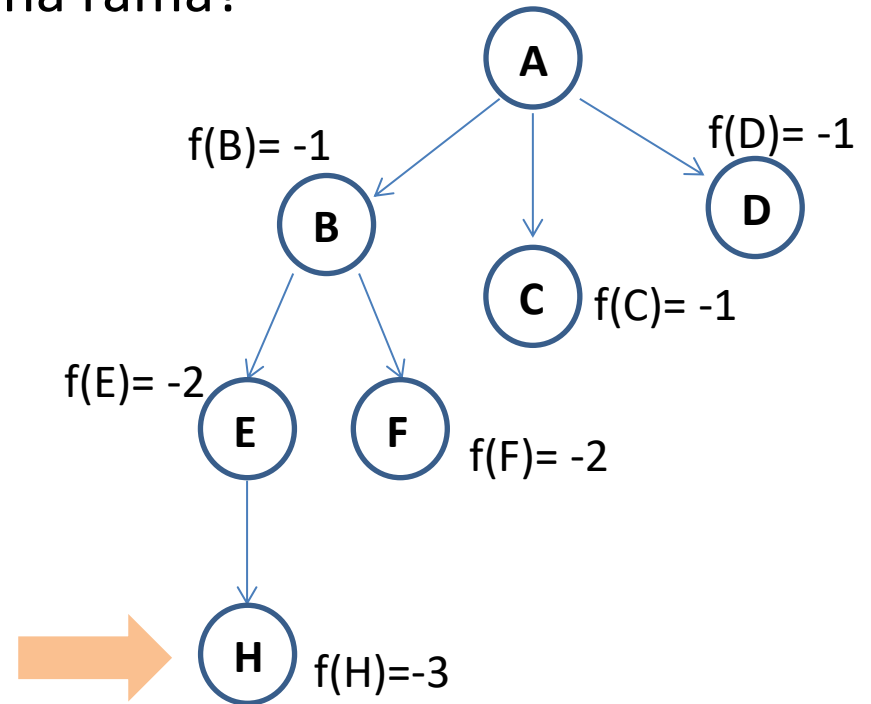


## Depth-first tree search

¿Cuándo parar la expansión de una rama?

Cuando ...

1) El nodo H no tiene hijos  
(no hay acciones aplicables)





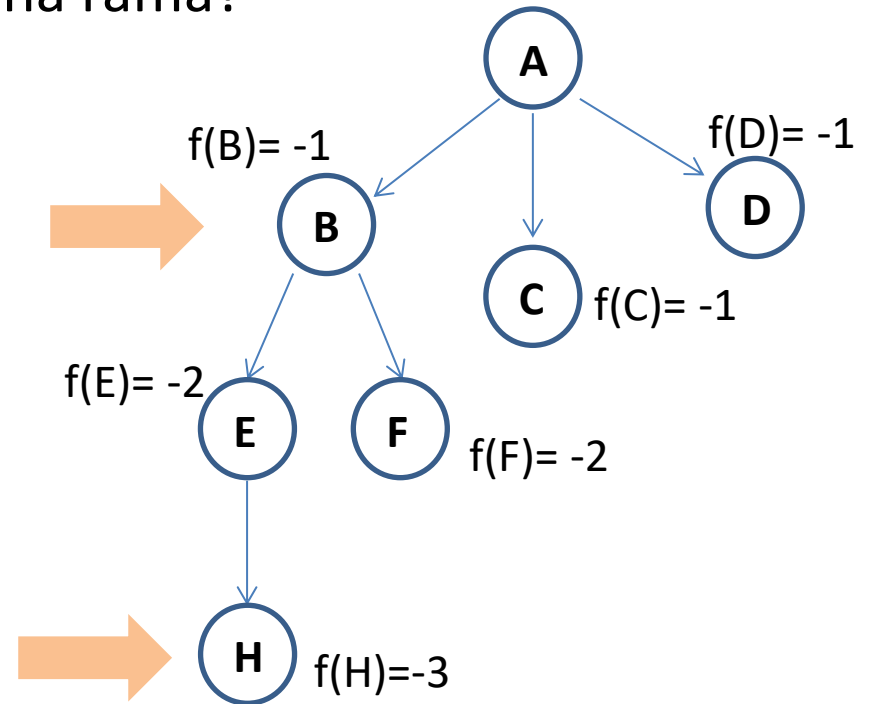
## Depth-first tree search

¿Cuándo parar la expansión de una rama?

Por ejemplo: si  $H=B$  entonces  $H$  es un nodo repetido

Cuando ...

- 1) El nodo  $H$  no tiene hijos  
(no hay acciones aplicables)
- 2) El nodo  $H$  es un estado repetido  
con un nodo de OPEN o PATH



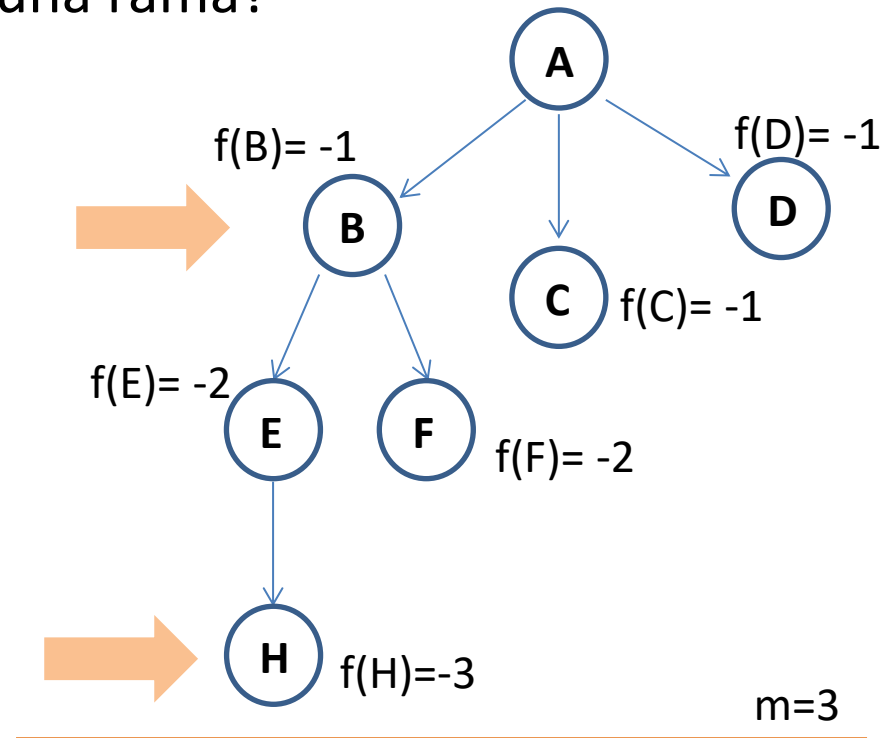
# Depth-first tree search

¿Cuándo parar la expansión de una rama?

Por ejemplo: si  $H=B$  entonces H es un nodo repetido

Cuando ...

- 1) El nodo H no tiene hijos  
(no hay acciones aplicables)
- 2) El nodo H es un estado repetido  
con un nodo de OPEN o PATH



- 3) Establecemos un máximo límite  $m$  de profundidad para la expansión del árbol

# Depth-first tree search

¿Cuándo parar la expansión de una rama?

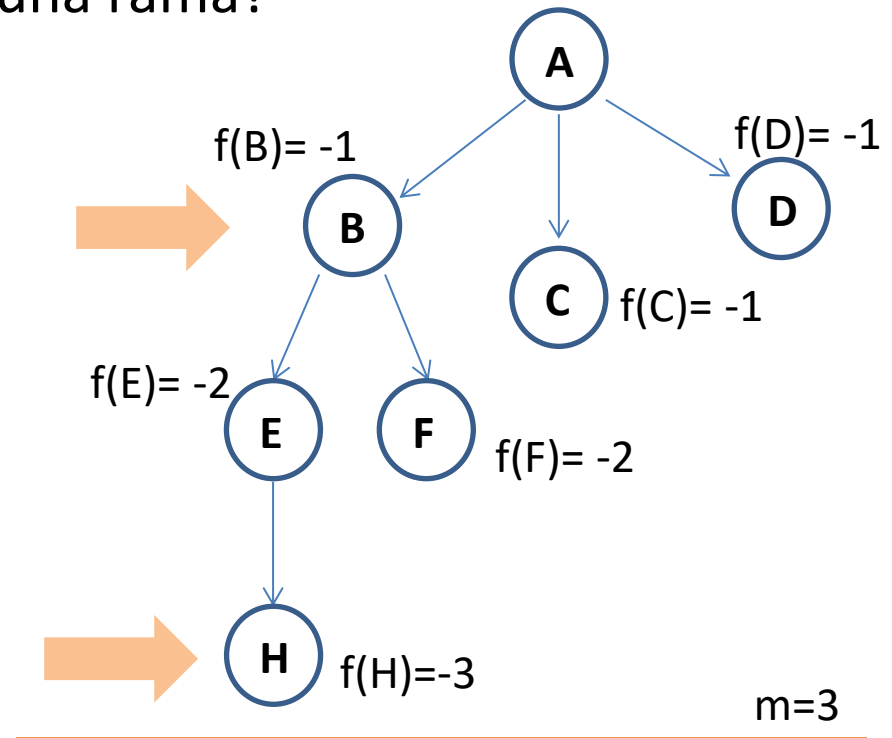
Por ejemplo: si  $H=B$  entonces  $H$  es un nodo repetido

Cuando ...

- 1) El nodo  $H$  no tiene hijos (no hay acciones aplicables)
- 2) El nodo  $H$  es un estado repetido con un nodo de OPEN o PATH

- 3) Establecemos un máximo límite  $m$  de profundidad para la expansión del árbol

En cualquiera de estos tres casos aplicamos **BACKTRACKING CRONOLÓGICO**



## 5. Depth-first tree search

Mantenemos una **PATH list** que *almacena los nodos del camino actual*

Asignamos **m=3**

## 5. Depth-first tree search

Mantenemos una **PATH list** que *almacena los nodos del camino actual*

Asignamos **m=3**

OPEN list = {A}

PATH list = {}



## 5. Depth-first tree search

Mantenemos una **PATH list** que *almacena los nodos del camino actual*

Asignamos **m=3**

OPEN list = {A}



PATH list = {}

Expandimos el nodo **A**

## 5. Depth-first tree search

OPEN list = {}

PATH list = {A}



Expandimos **A**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {}

PATH list = {A}



Expandimos **A**  
(lo ponemos en PATH)

Si **A** es objetivo => STOP



## 5. Depth-first tree search

OPEN list = {}

PATH list = {A}



Expandimos **A**  
(lo ponemos en PATH)

Si **A** es objetivo => STOP

Si **nivel(A) = m** => BACKTRACKING

## 5. Depth-first tree search

OPEN list = {}

PATH list = {A}



Expandimos **A**  
(lo ponemos en PATH)

Si **A** es objetivo => STOP

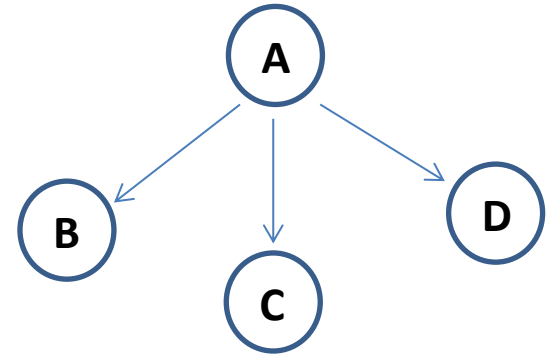
Si **nivel(A) = m** => BACKTRACKING

En cualquier otro caso, generamos los hijos

## 5. Depth-first tree search

OPEN list = {}

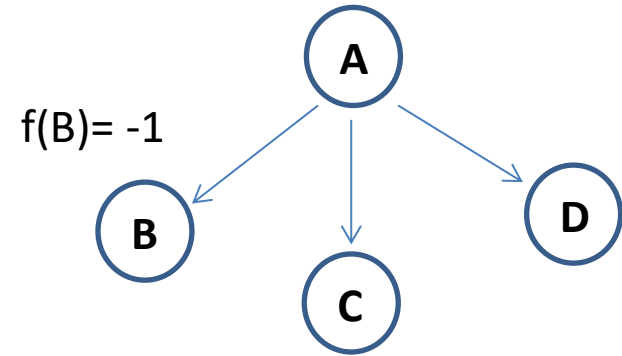
PATH list = {A}



## 5. Depth-first tree search

OPEN list = {}

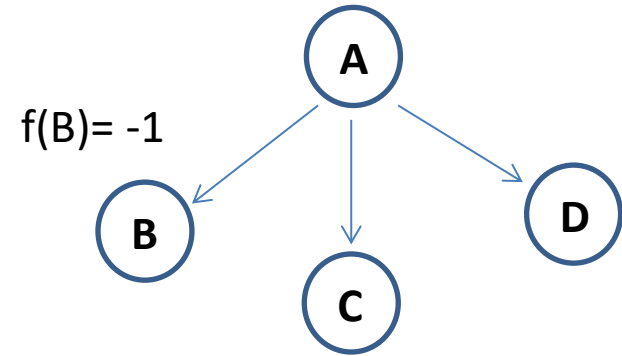
PATH list = {A}



## 5. Depth-first tree search

OPEN list = {B}

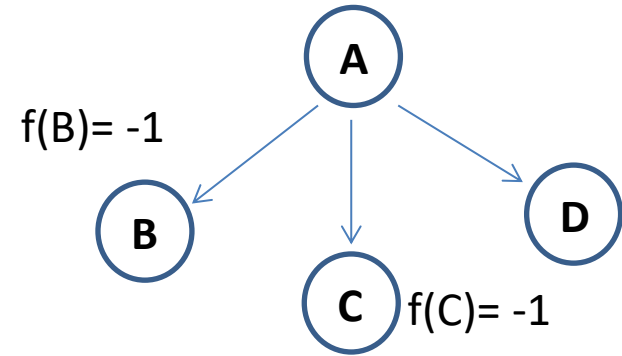
PATH list = {A}



## 5. Depth-first tree search

OPEN list = {B}

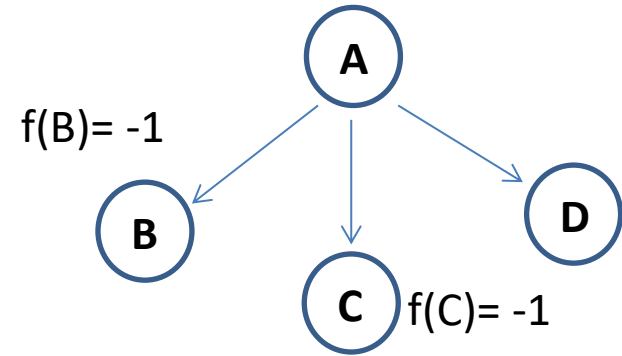
PATH list = {A}



## 5. Depth-first tree search

OPEN list = {B,C}

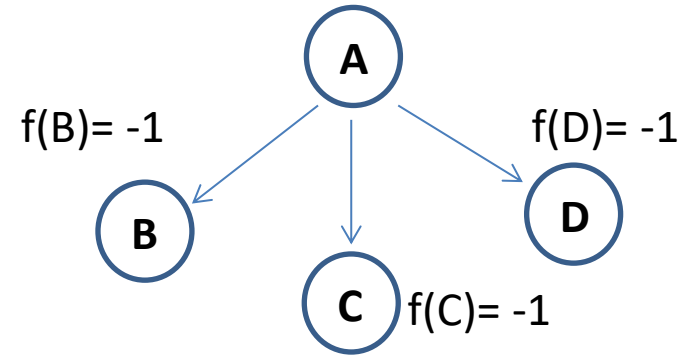
PATH list = {A}



## 5. Depth-first tree search

OPEN list = {B,C,D}

PATH list = {A}

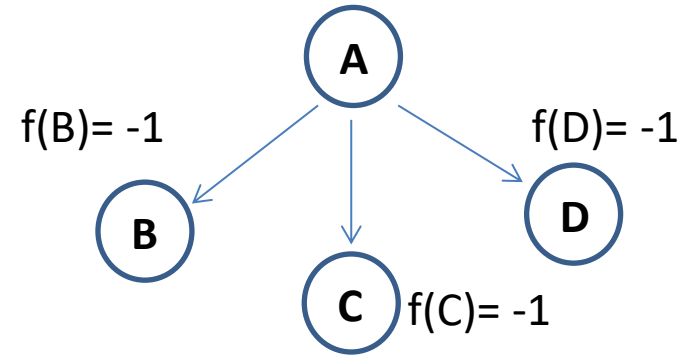




## 5. Depth-first tree search

OPEN list = {C,D}

PATH list = {A,B}

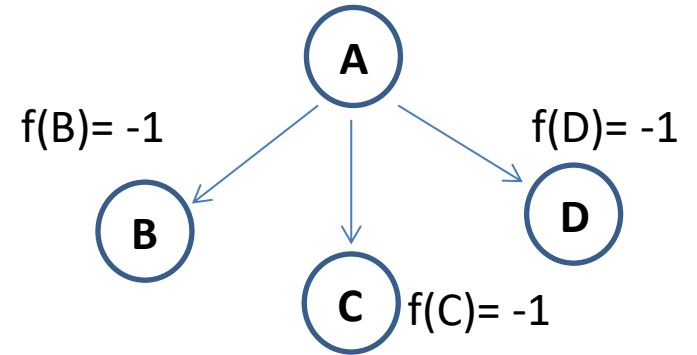


Expandimos **B**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {C,D}

PATH list = {A,B}



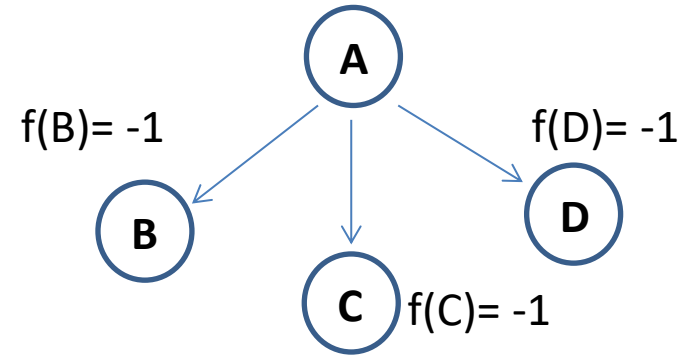
Expandimos **B**  
(lo ponemos en PATH)

Si **B** es objetivo => STOP

## 5. Depth-first tree search

OPEN list = {C,D}

PATH list = {A,B}



Expandimos **B**  
(lo ponemos en PATH)

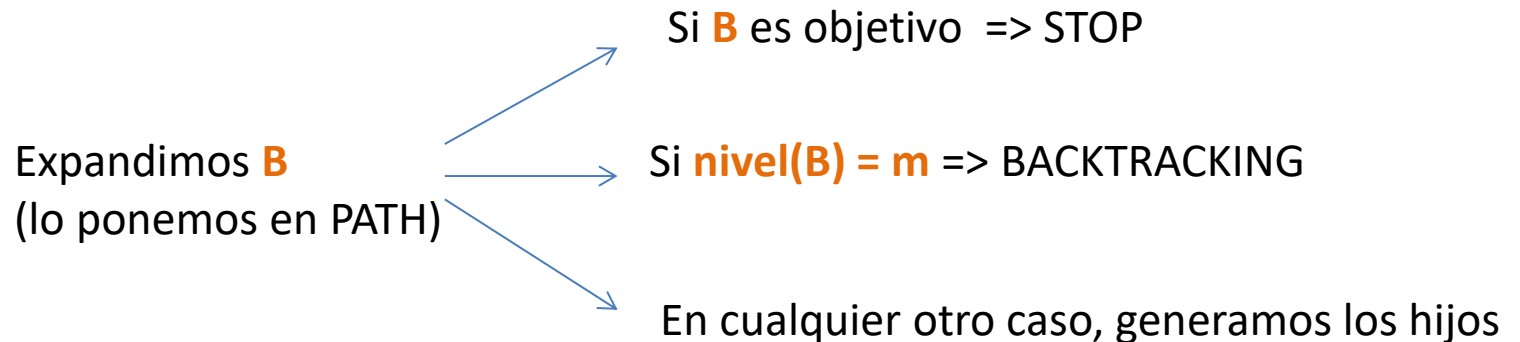
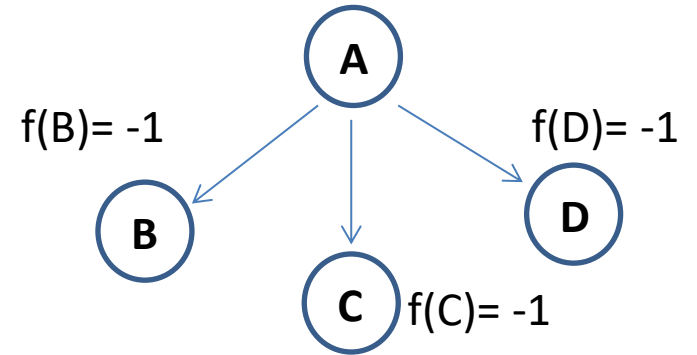
Si **B** es objetivo  $\Rightarrow$  STOP

Si **nivel(B) = m**  $\Rightarrow$  BACKTRACKING

## 5. Depth-first tree search

OPEN list = {C,D}

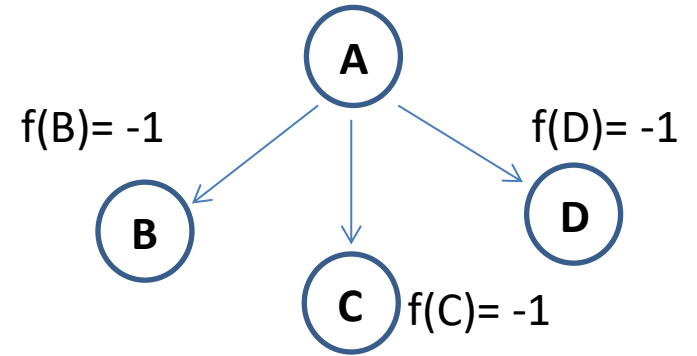
PATH list = {A,B}



## 5. Depth-first tree search

OPEN list = {C,D}

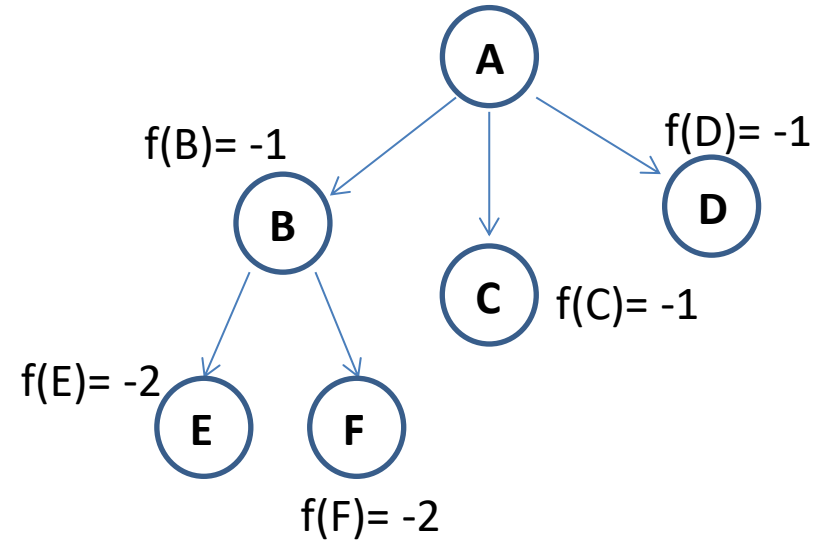
PATH list = {A,B}



## 5. Depth-first tree search

OPEN list = {E, F, C, D}

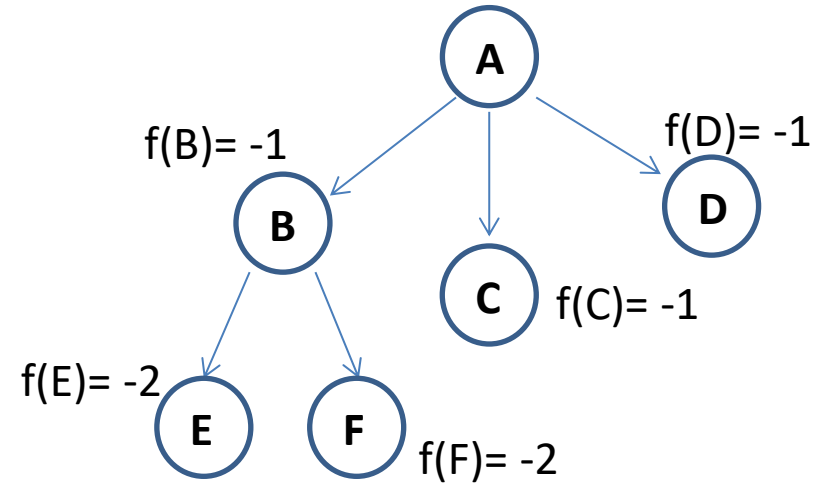
PATH list = {A, B}



## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}

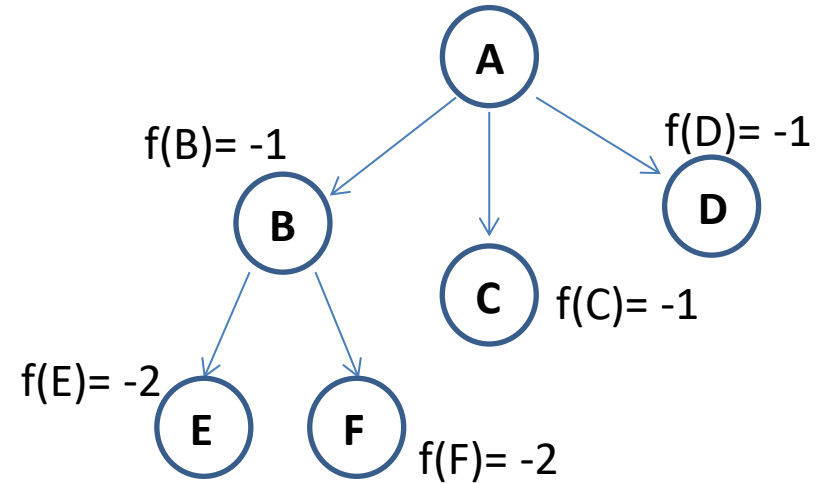


Expandimos **E**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}



Si **E** es objetivo  $\Rightarrow$  STOP

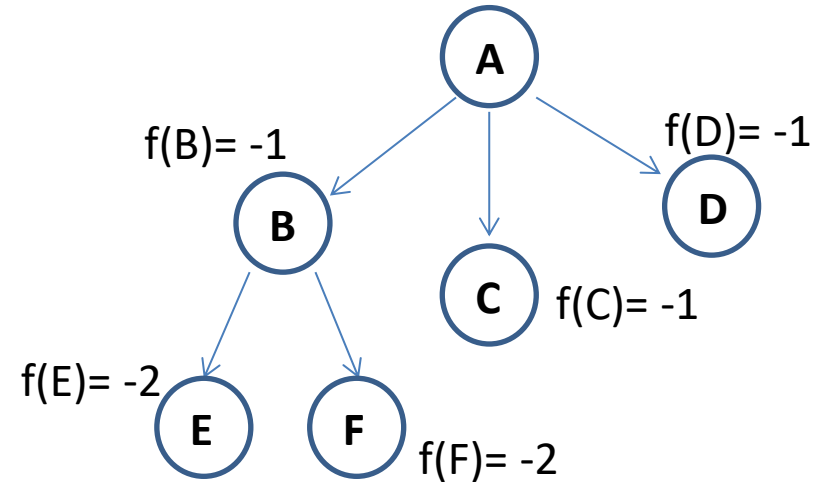
Expandimos **E**  
(lo ponemos en PATH)



## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}



Expandimos **E**  
(lo ponemos en PATH)

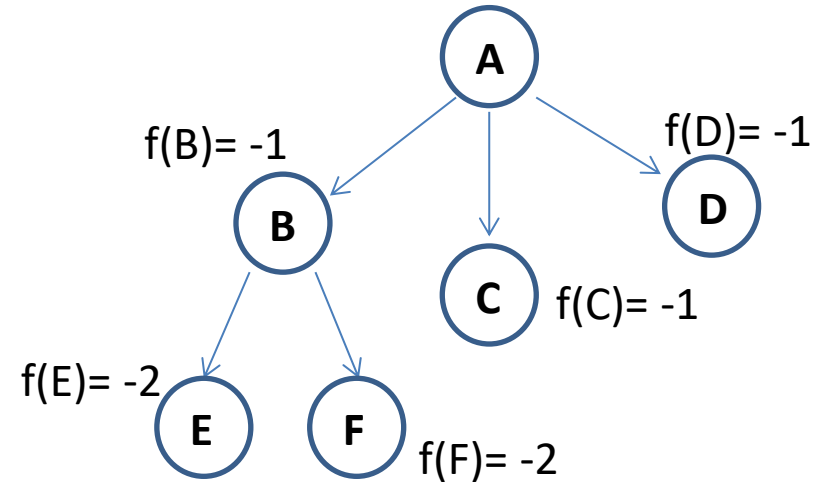
Si **E** es objetivo  $\Rightarrow$  STOP

Si **nivel(E) = m**  $\Rightarrow$  BACKTRACKING

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}



Expandimos **E**  
(lo ponemos en PATH)

Si **E** es objetivo  $\Rightarrow$  STOP

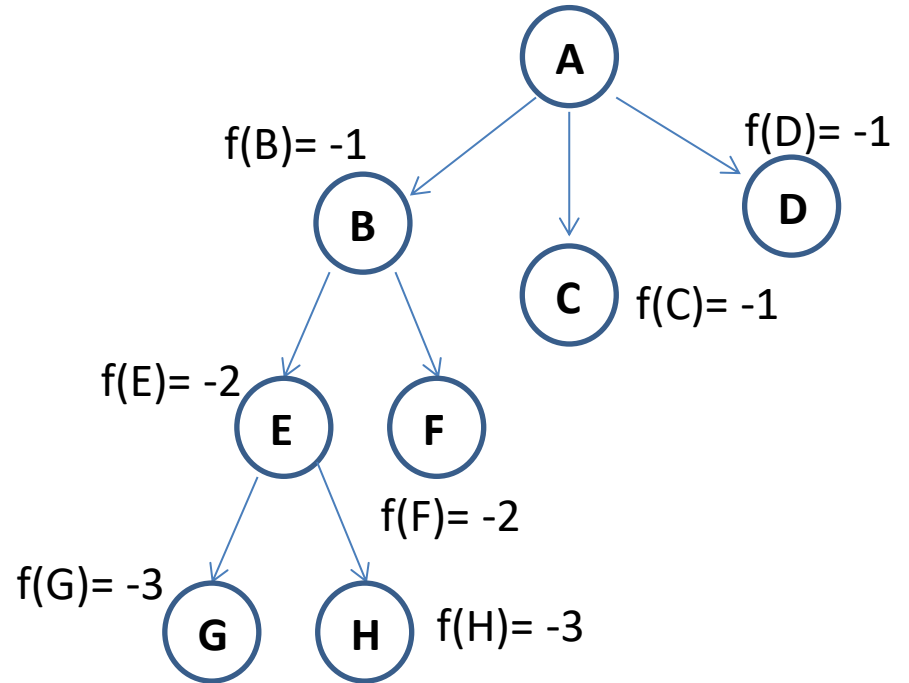
Si **nivel(E) = m**  $\Rightarrow$  BACKTRACKING

En cualquier otro caso, generamos los hijos

## 5. Depth-first tree search

OPEN list = {G, H, F, C, D}

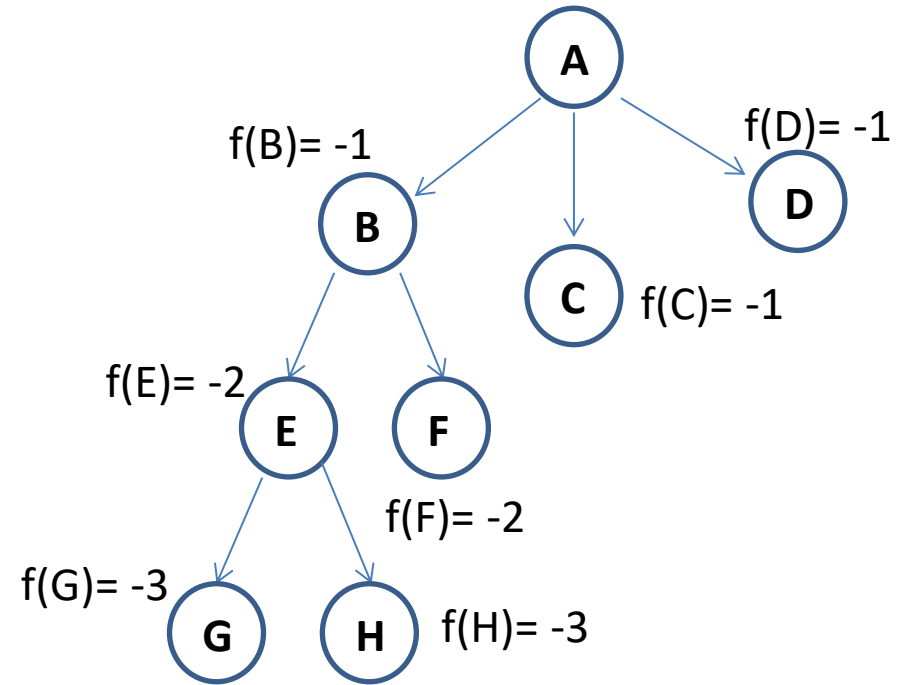
PATH list = {A, B, E}



## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E, G}

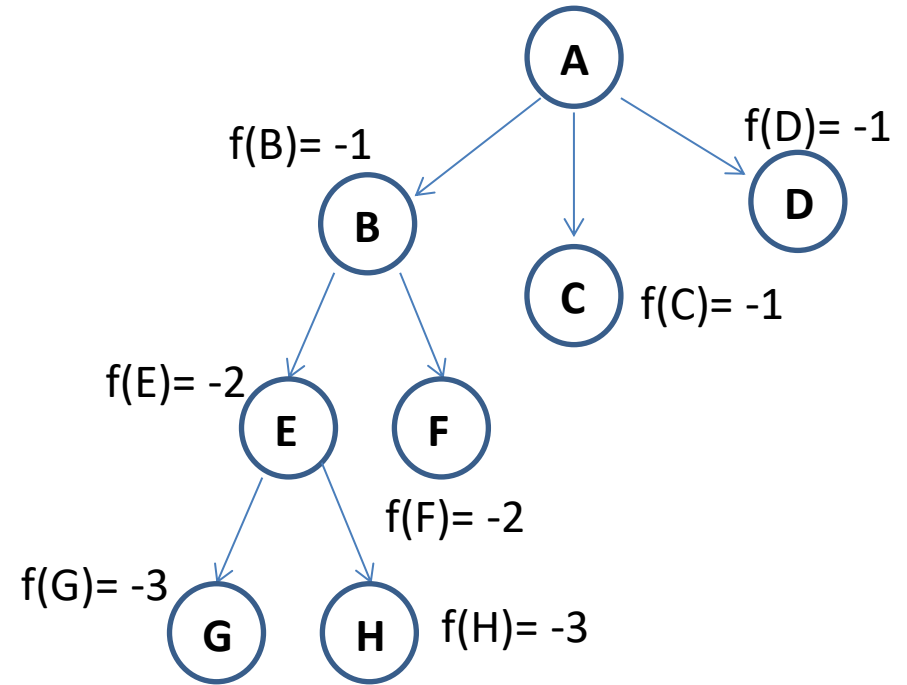


Expandimos **G**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E, G}



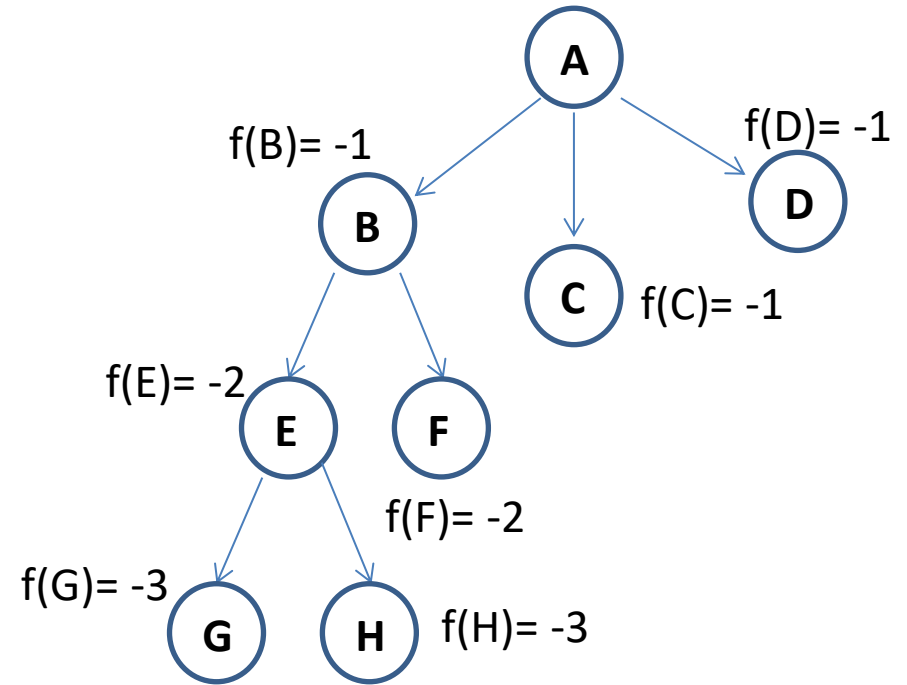
Si **G** es objetivo  $\Rightarrow$  STOP

Expandimos **G**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E, G}



Si **G** es objetivo  $\Rightarrow$  STOP

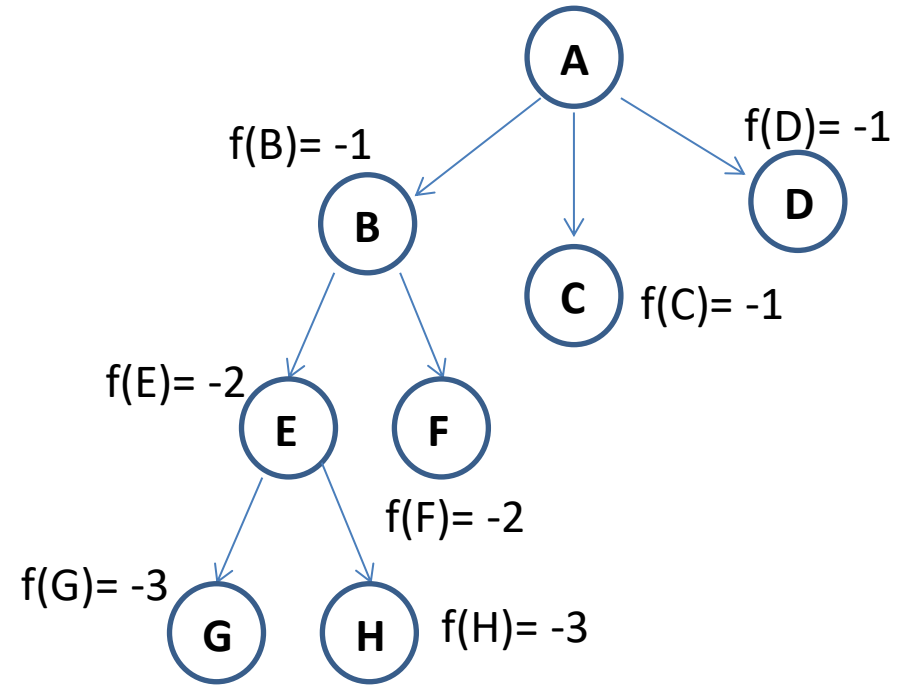
Expandimos **G**  
(lo ponemos en PATH)

Si **nivel(G) = m**  $\Rightarrow$  BACKTRACKING

## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E, G}



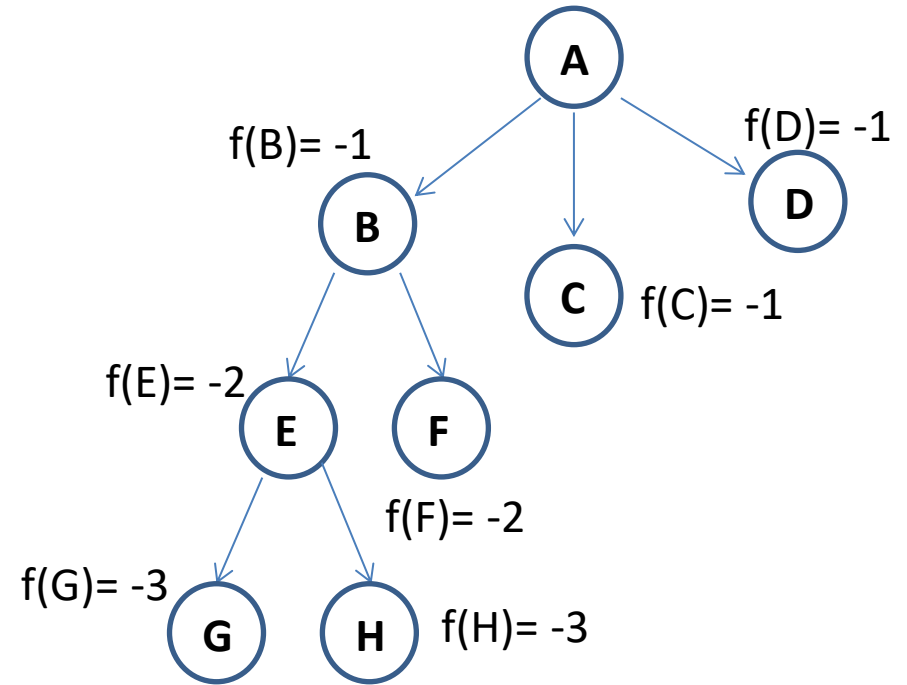
### BACKTRACKING (n):

1. Eliminar  $n$  de la lista PATH
2. Si  $\text{parent}(n)$  tiene más hijos en OPEN  $\Rightarrow$  seleccionar el siguiente hijo de OPEN list
3. Si  $\text{parent}(n)$  no tiene más hijos en OPEN  $\Rightarrow$  BACKTRACKING ( $\text{parent}(n)$ )

## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E, G}



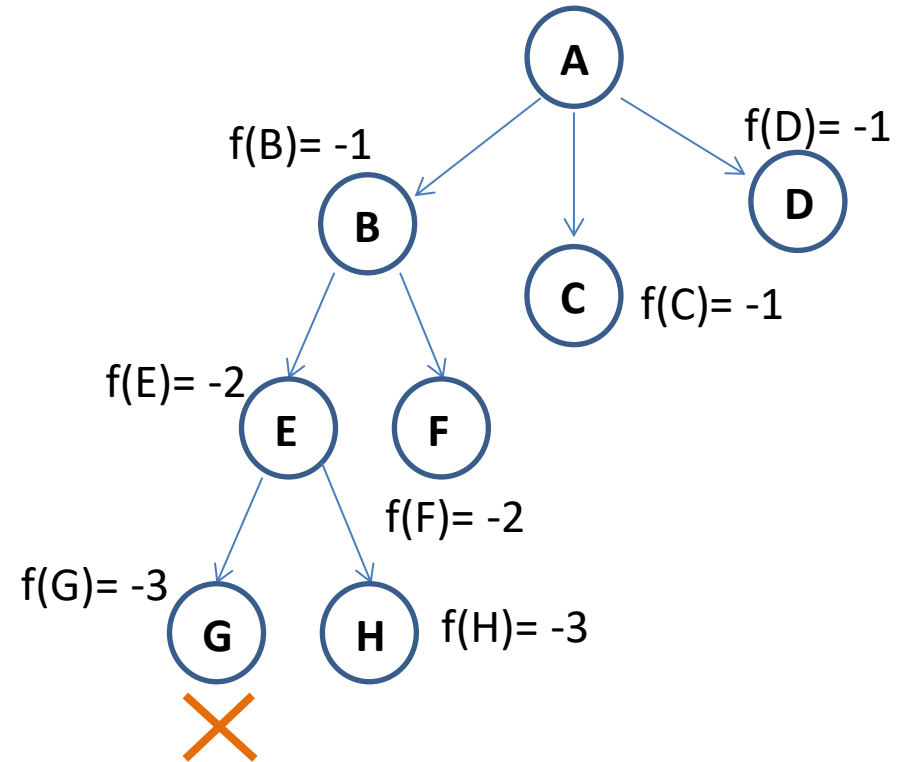
**Backtracking (G)**



## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E}



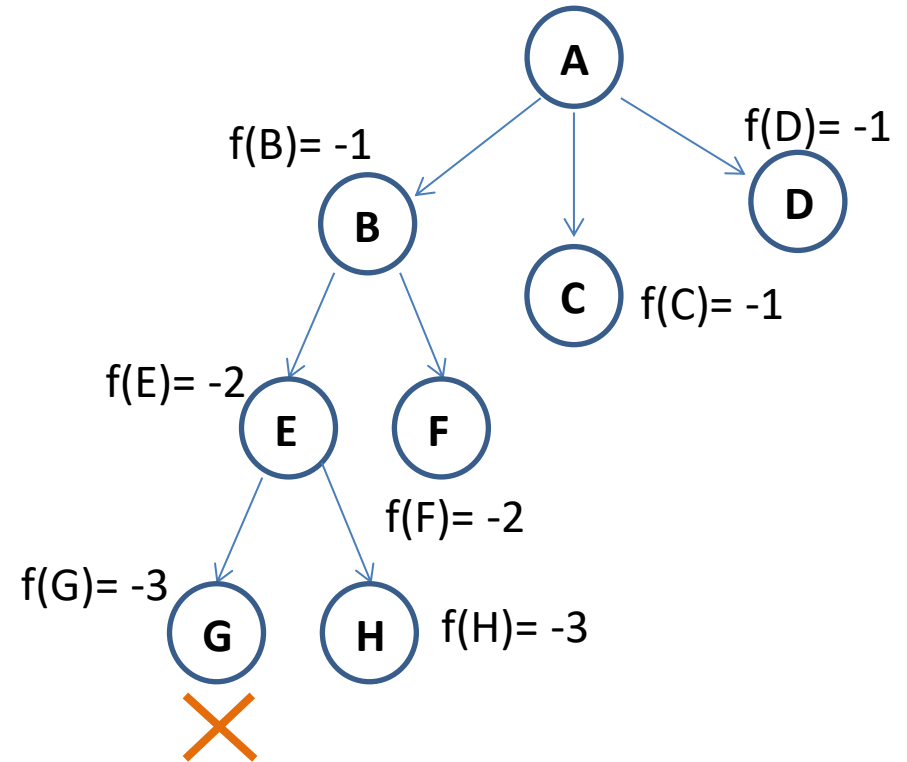
**Backtracking (G)**

Eliminar G de PATH

## 5. Depth-first tree search

OPEN list = {H, F, C, D}

PATH list = {A, B, E}



**Backtracking (G)**

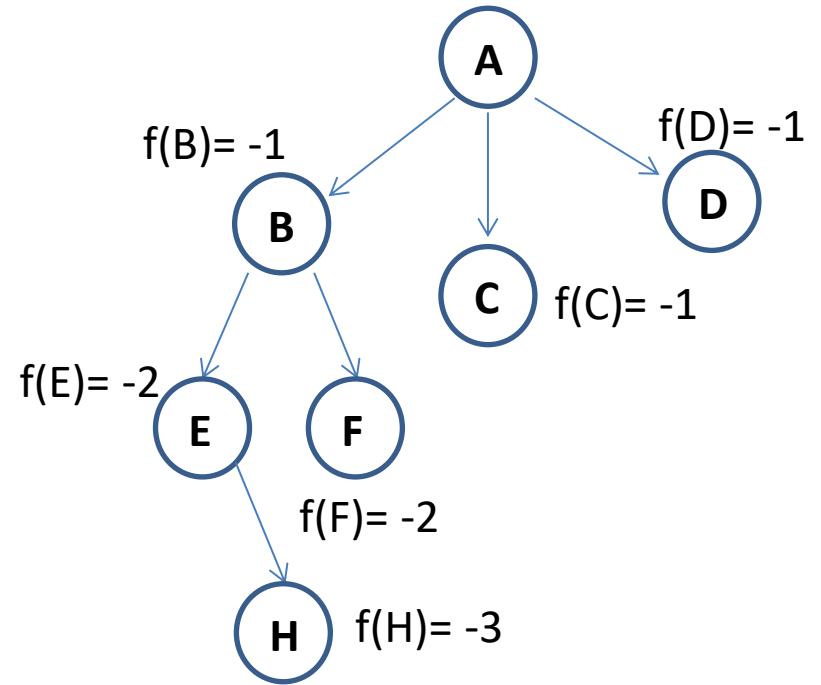
Eliminar G de PATH

Seleccionar el siguiente nodo de OPEN

## Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E, H}

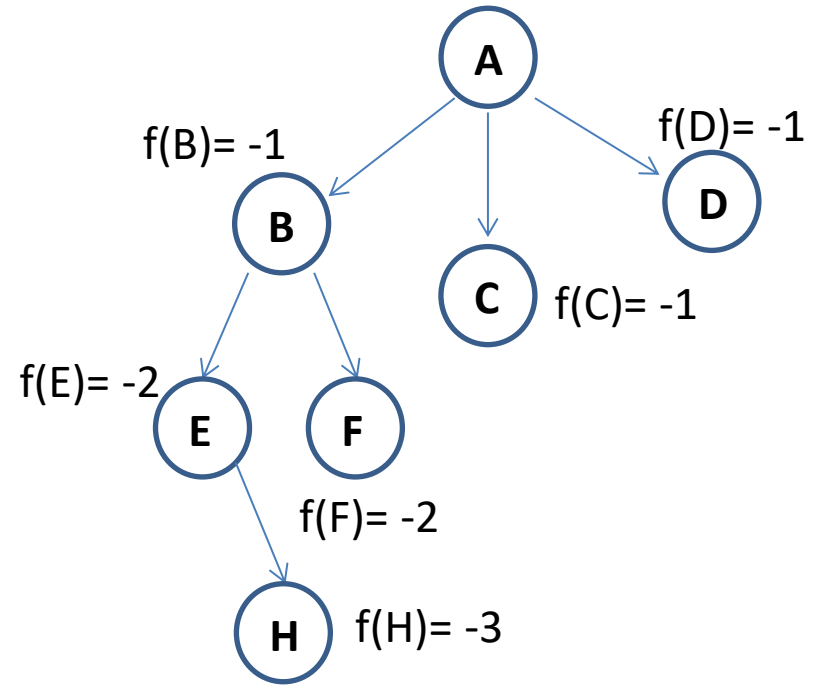


Expandimos el nodo **H**  
(lo ponemos en PATH)

# Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E, H}



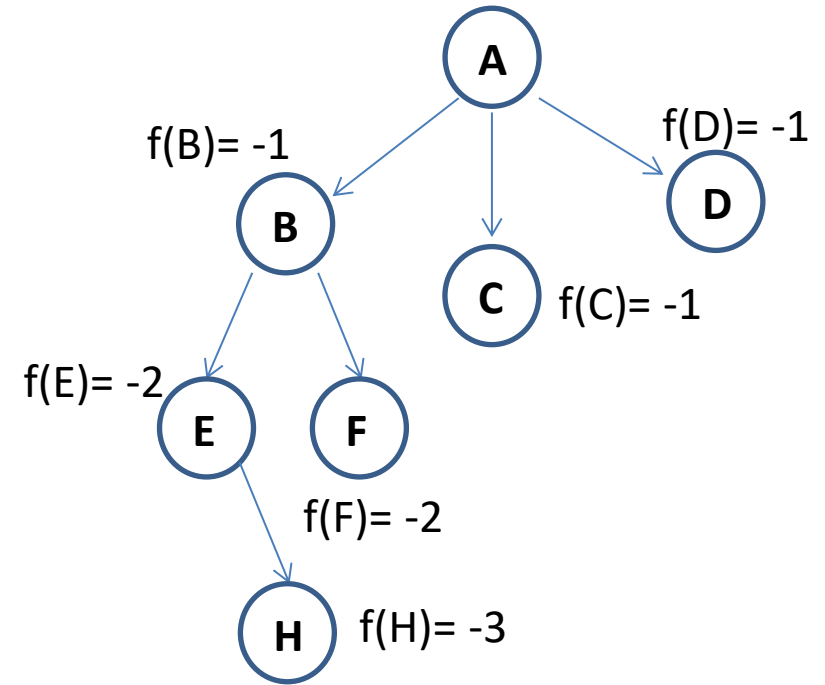
Expandimos el nodo **H**  
(lo ponemos en PATH)

Si **H** es objetivo  $\Rightarrow$  STOP

# Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E, H}



Expandimos el nodo **H**  
(lo ponemos en PATH)

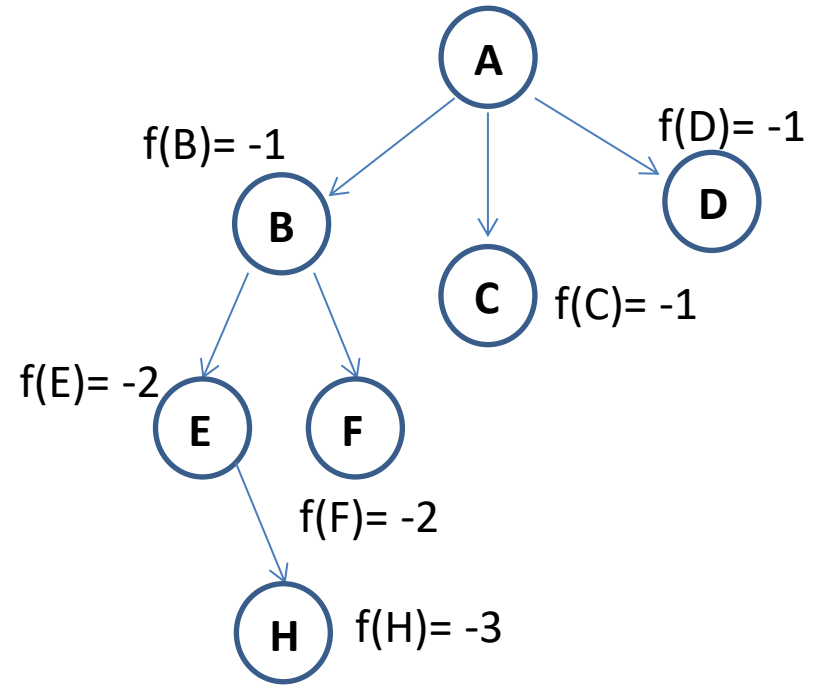
Si **H** es objetivo  $\Rightarrow$  STOP

Si **nivel(H) = m**  $\Rightarrow$  **BACKTRACKING**

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E, H}

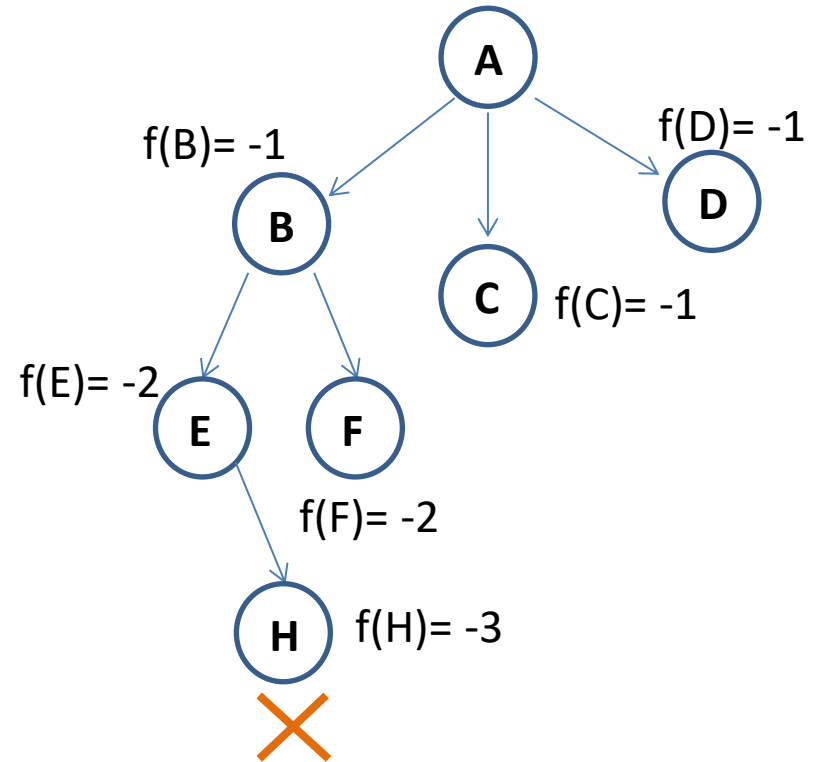


**Backtracking (H)**

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}



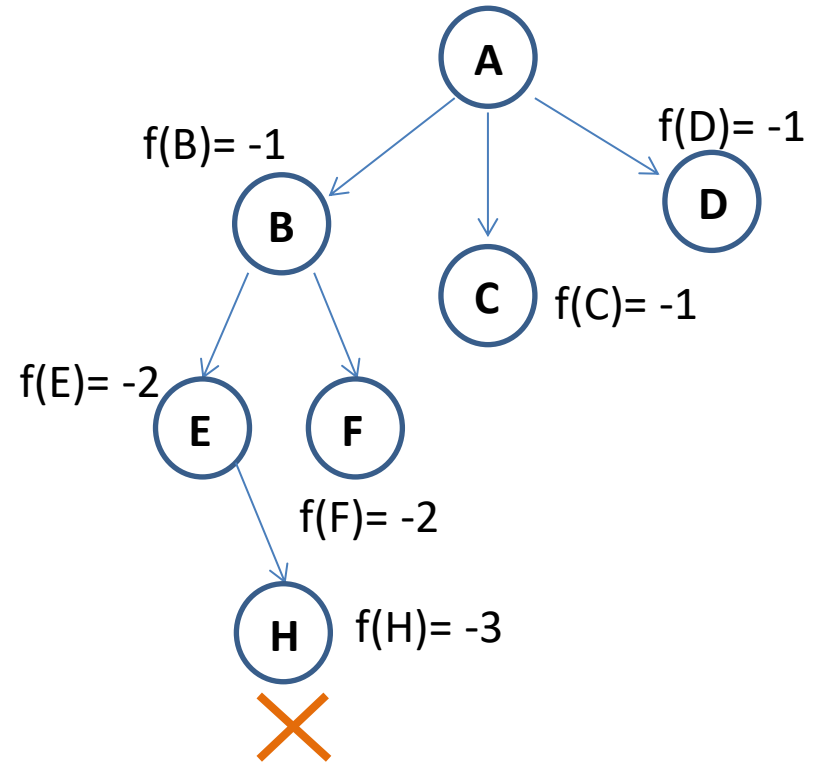
**Backtracking (H)**

Eliminamos H de PATH

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}



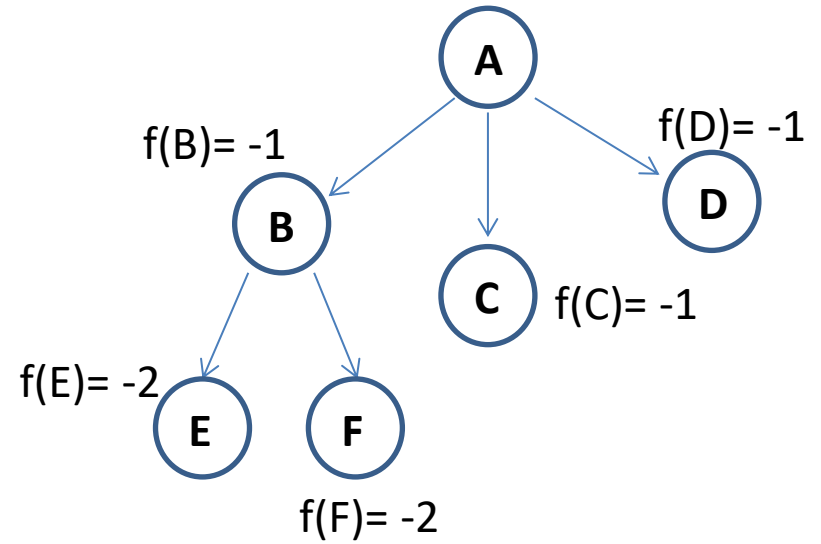
Backtracking (H) → Eliminamos H de PATH  
Backtracking (H) → Backtracking (E)



## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B, E}

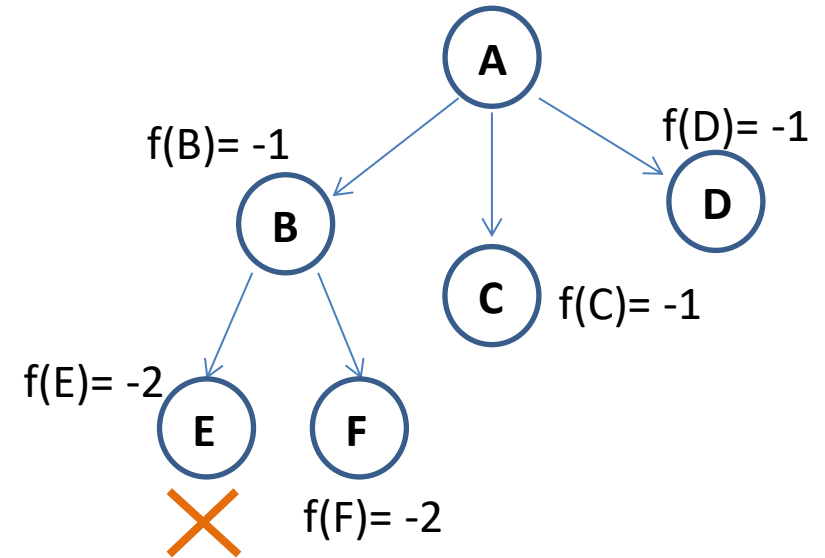


**Backtracking (E)**

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B}



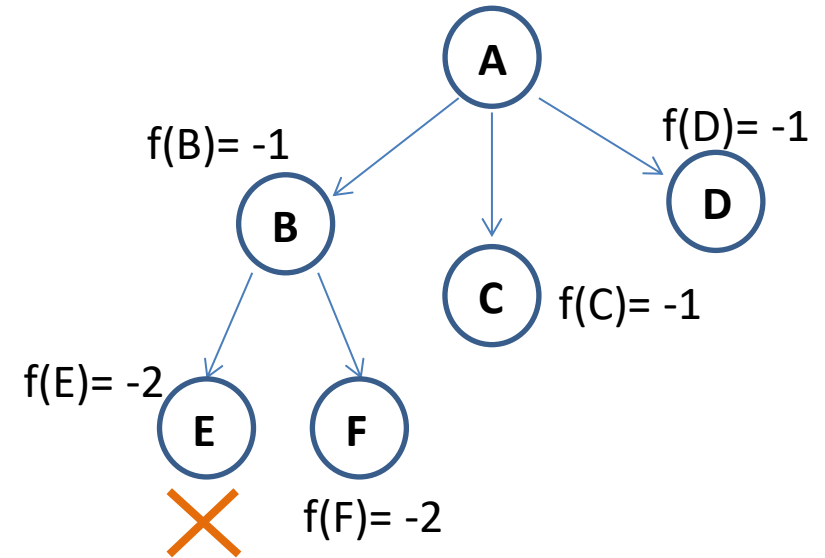
**Backtracking (E)**

Eliminamos E de PATH

## 5. Depth-first tree search

OPEN list = {F, C, D}

PATH list = {A, B}



**Backtracking (E)**

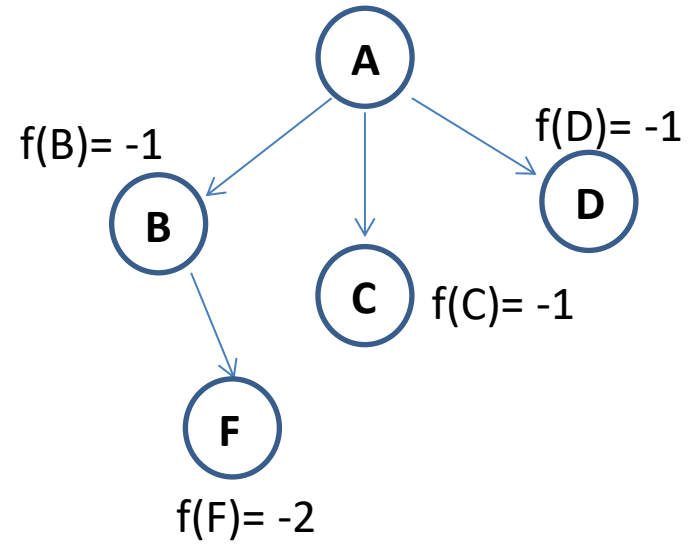
Eliminamos E de PATH

Seleccionamos el siguiente nodo de OPEN → nodo F

## 5. Depth-first tree search

OPEN list = {C, D}

PATH list = {A, B, F}

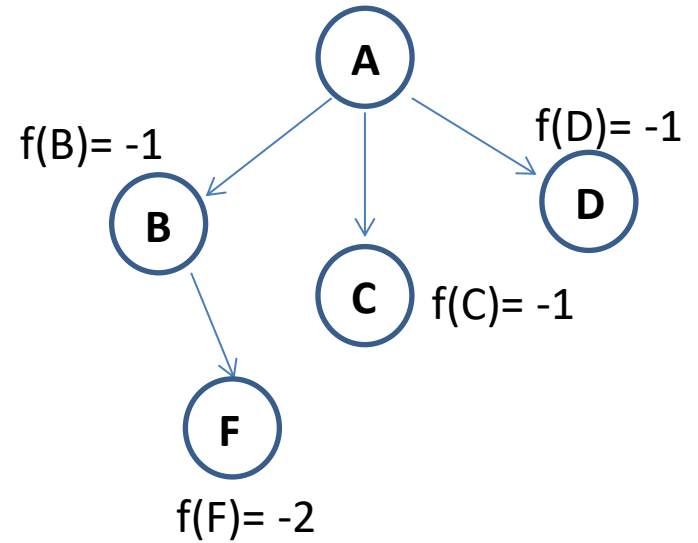


Expandimos **F**  
(lo ponemos en PATH)

## 5. Depth-first tree search

OPEN list = {C, D}

PATH list = {A, B, F}



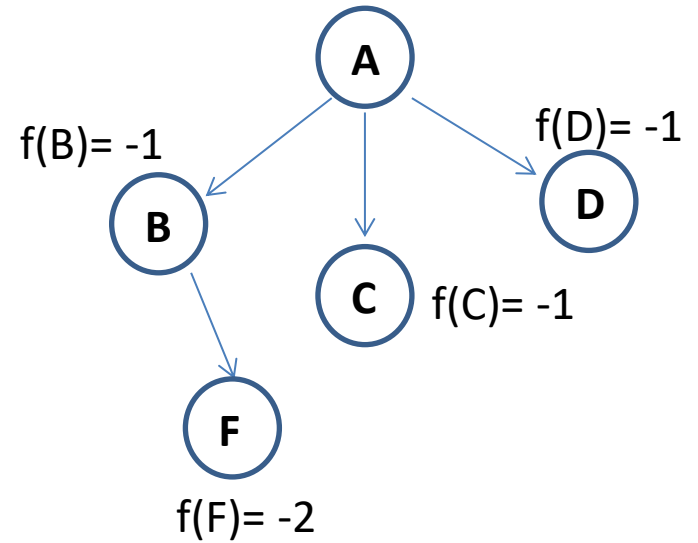
Expandimos **F**  
(lo ponemos en PATH)

Si **F** es objetivo => STOP

## 5. Depth-first tree search

OPEN list = {C, D}

PATH list = {A, B, F}



Expandimos **F**  
(lo ponemos en PATH)

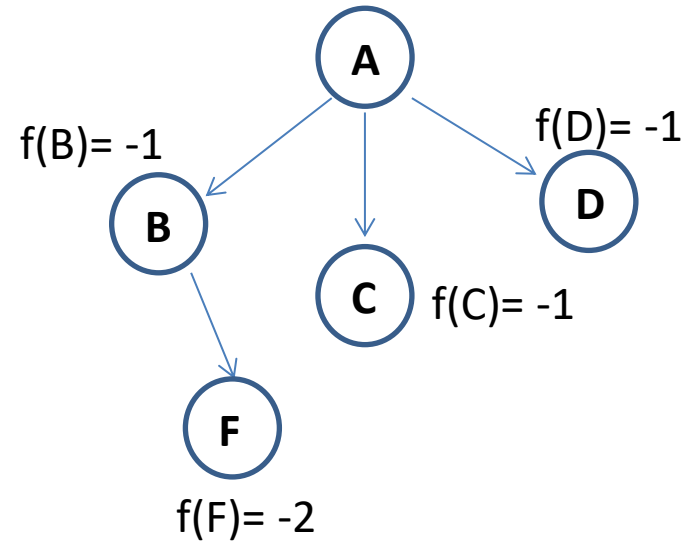
Si **F** es objetivo => STOP

Si **nivel(F) = m** => BACKTRACKING

## 5. Depth-first tree search

OPEN list = {C, D}

PATH list = {A, B, F}



Expandimos **F**

(lo ponemos en PATH)

Si **F** es objetivo  $\Rightarrow$  STOP

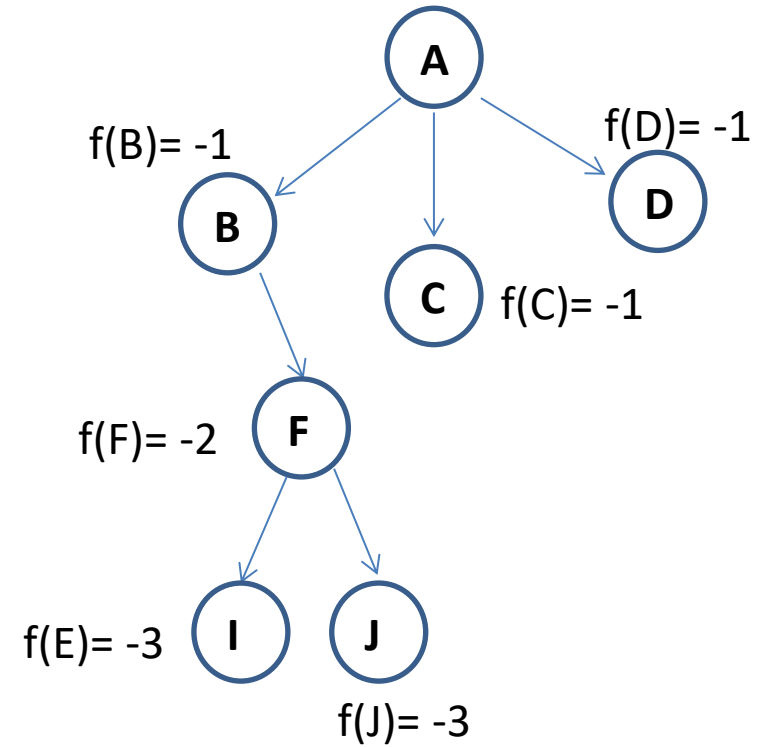
Si **nivel(F) = m**  $\Rightarrow$  BACKTRACKING

En cualquier otro caso, generamos los hijos

## Depth-first tree search

OPEN list = {I, J, C, D}

PATH list = {A, B, F}





# Depth-first tree search

**Espacio lineal!!**

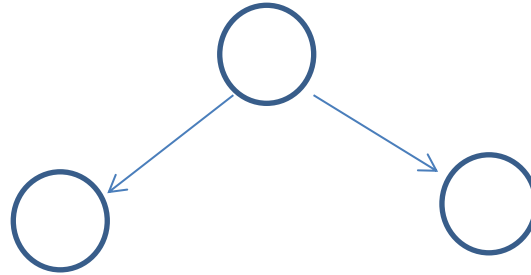
# Depth-first tree search

**Espacio lineal!!**



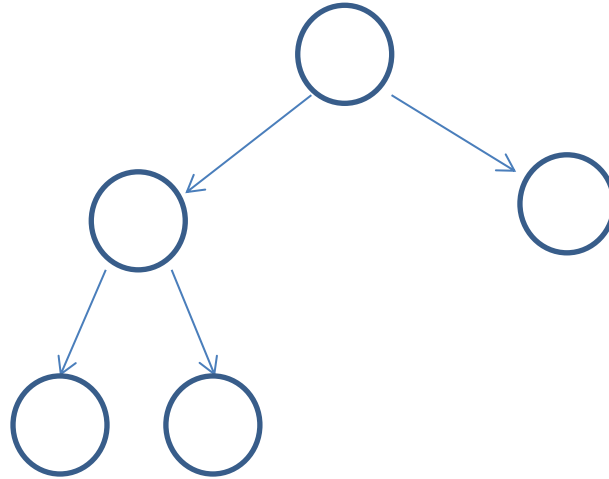
# Depth-first tree search

**Espacio lineal!!**



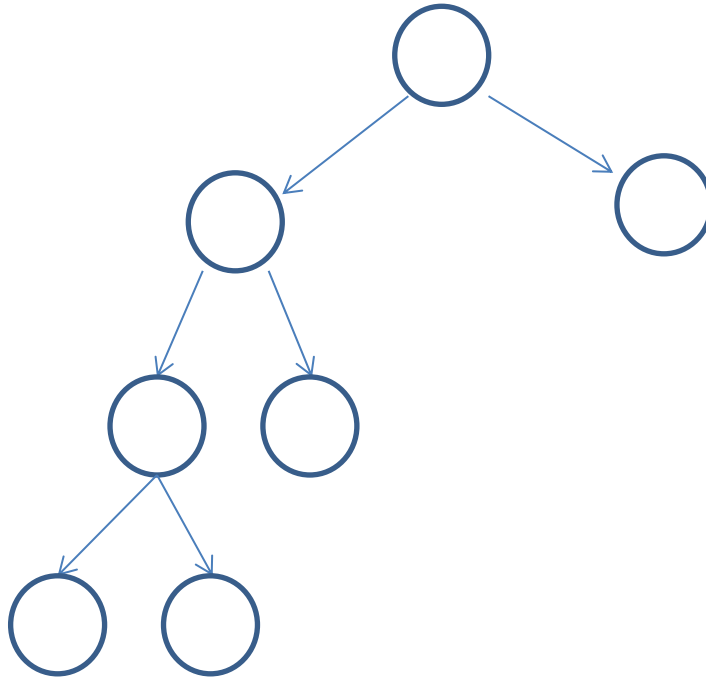
# Depth-first tree search

**Espacio lineal!!**



# Depth-first tree search

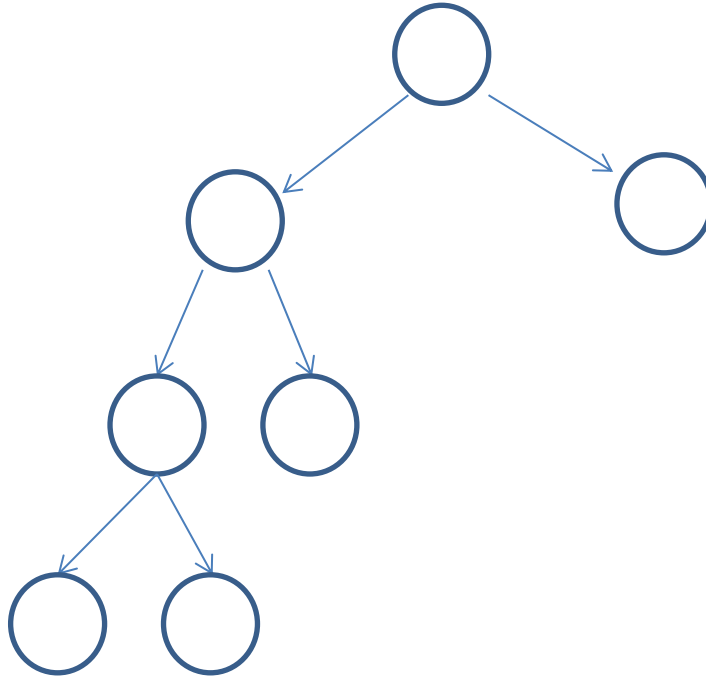
**Espacio lineal!!**



# Depth-first tree search

Espacio lineal!!

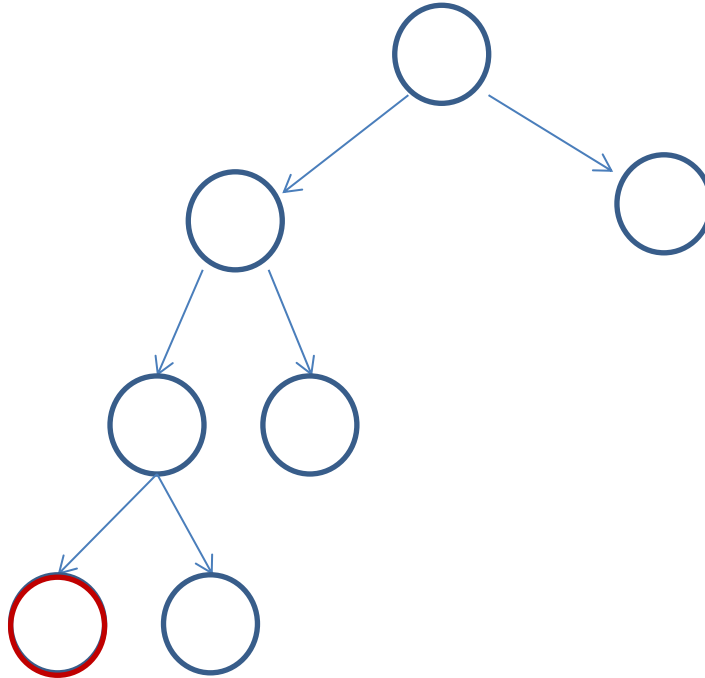
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

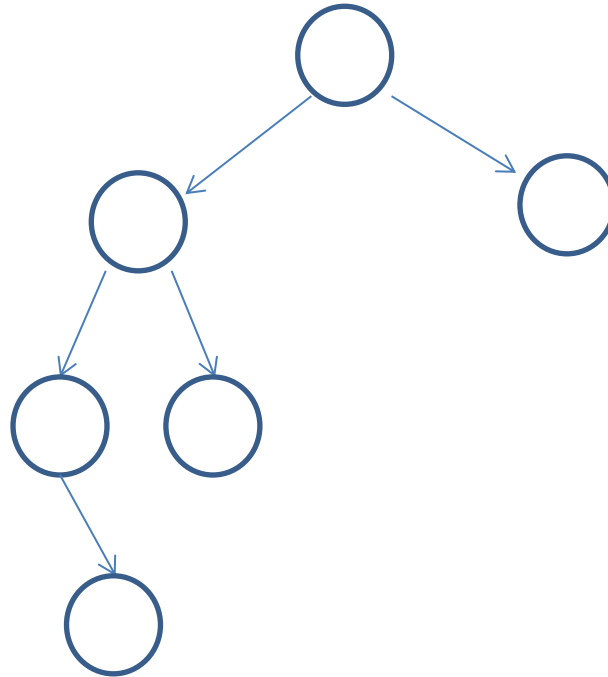
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

$O(b \cdot m)$

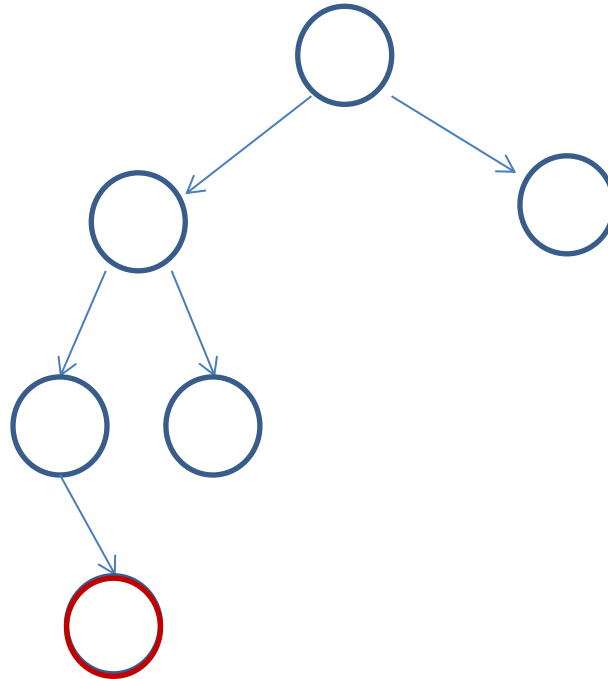




# Depth-first tree search

Espacio lineal!!

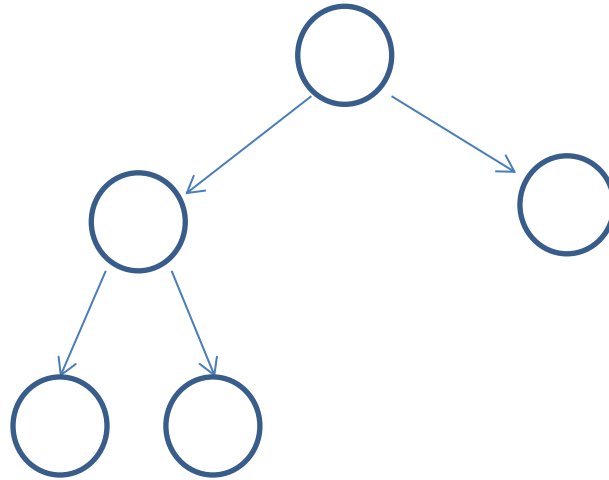
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

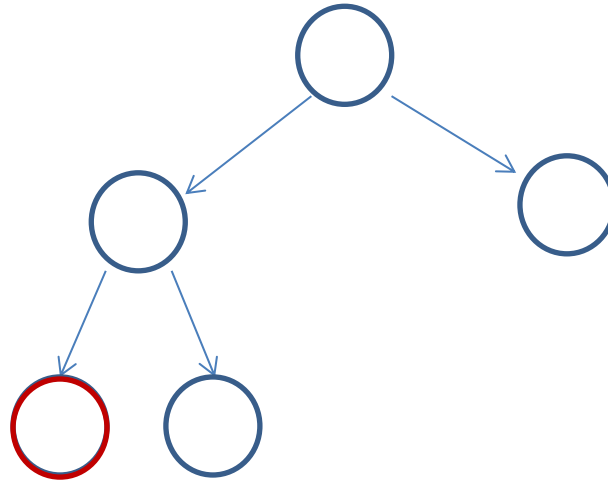
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

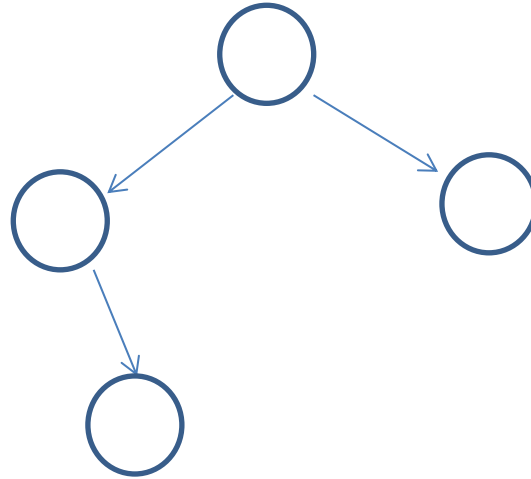
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

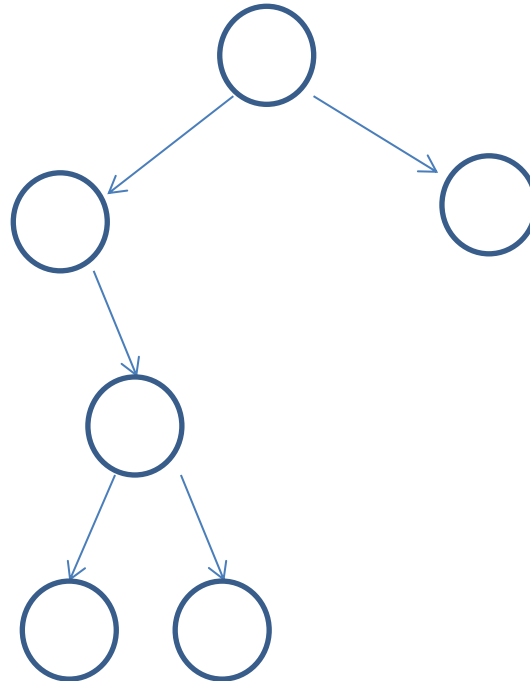
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

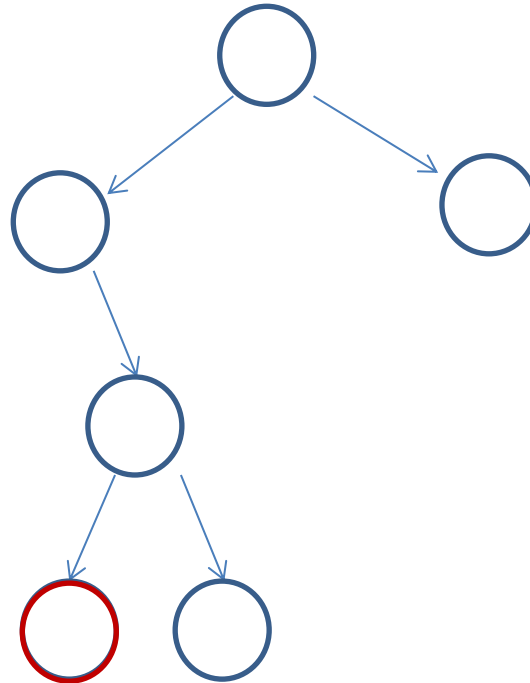
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

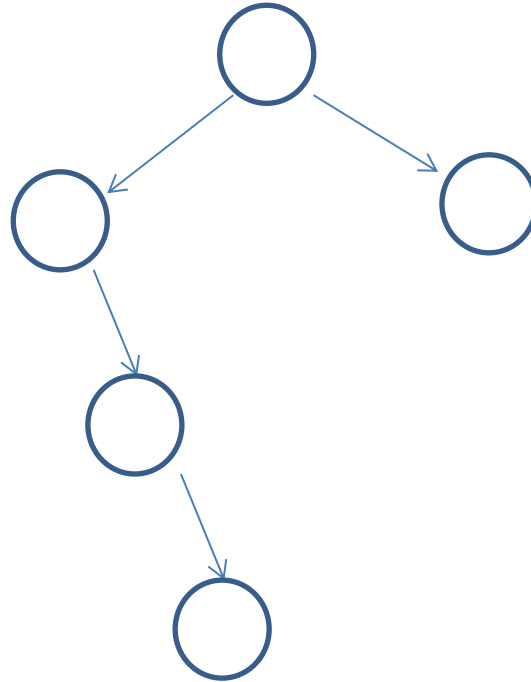
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

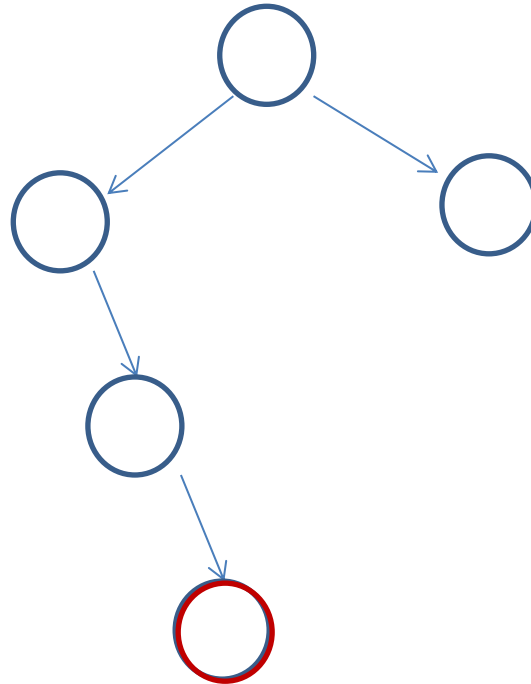
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

$O(b \cdot m)$

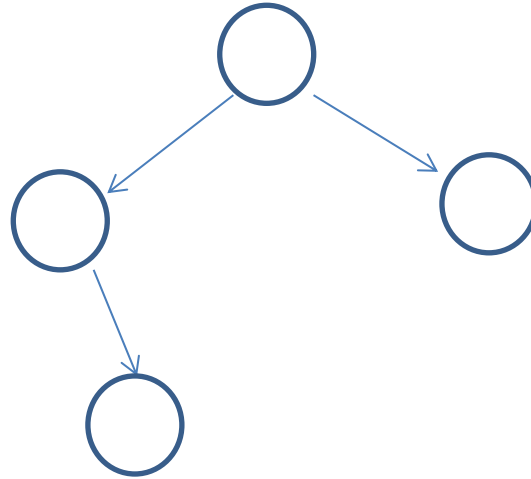




# Depth-first tree search

Espacio lineal!!

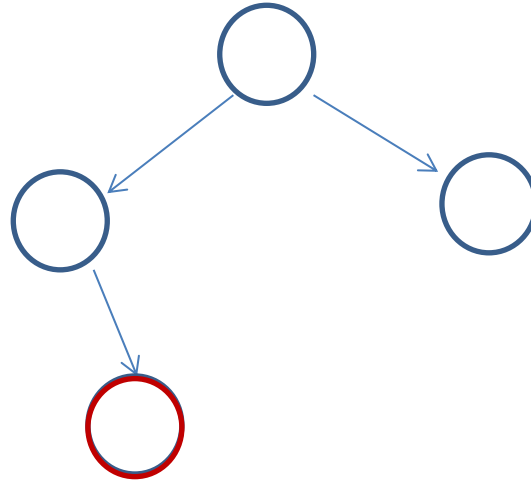
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

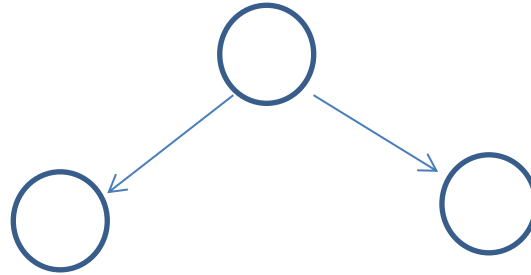
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

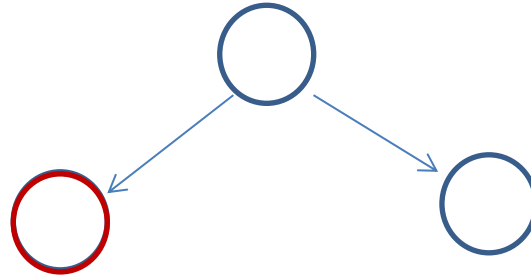
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

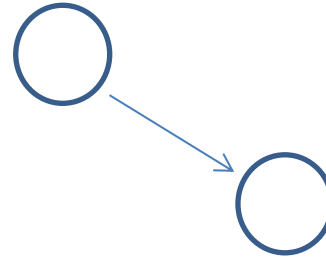
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

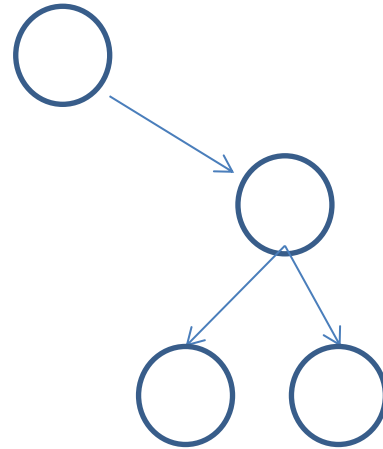
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

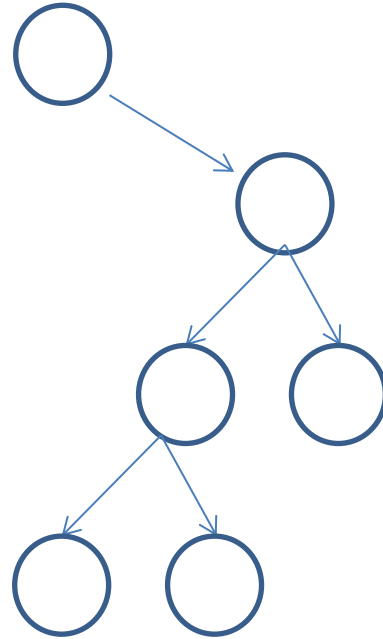
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

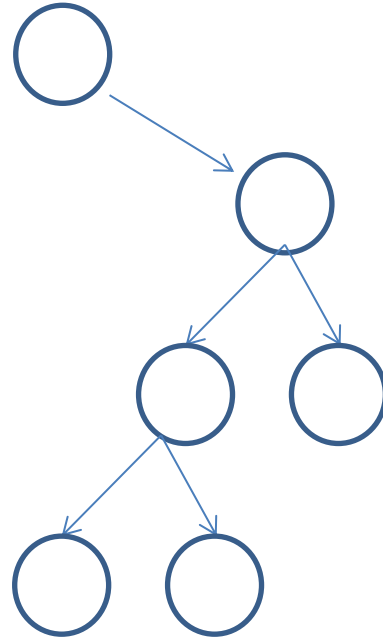
$O(b \cdot m)$



# Depth-first tree search

Espacio lineal!!

$$O(b \cdot m)$$





# Depth-first tree search

Espacio lineal!!

$O(b \cdot m)$

