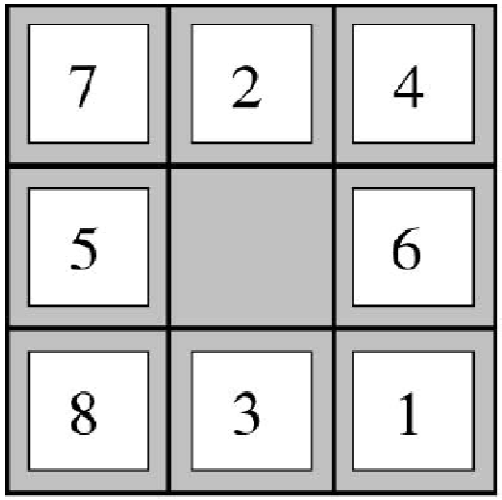
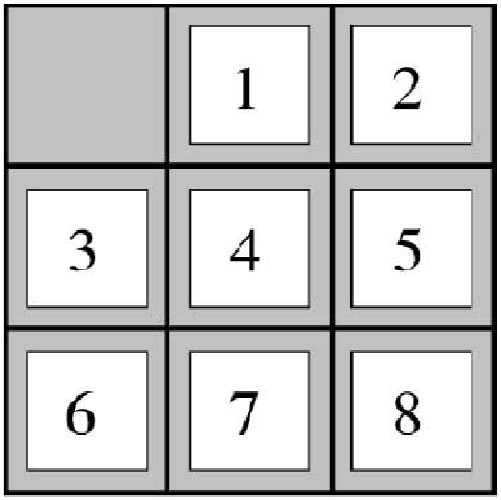
Ejemplo: el 8-puzle http://www.8puzzle.com/



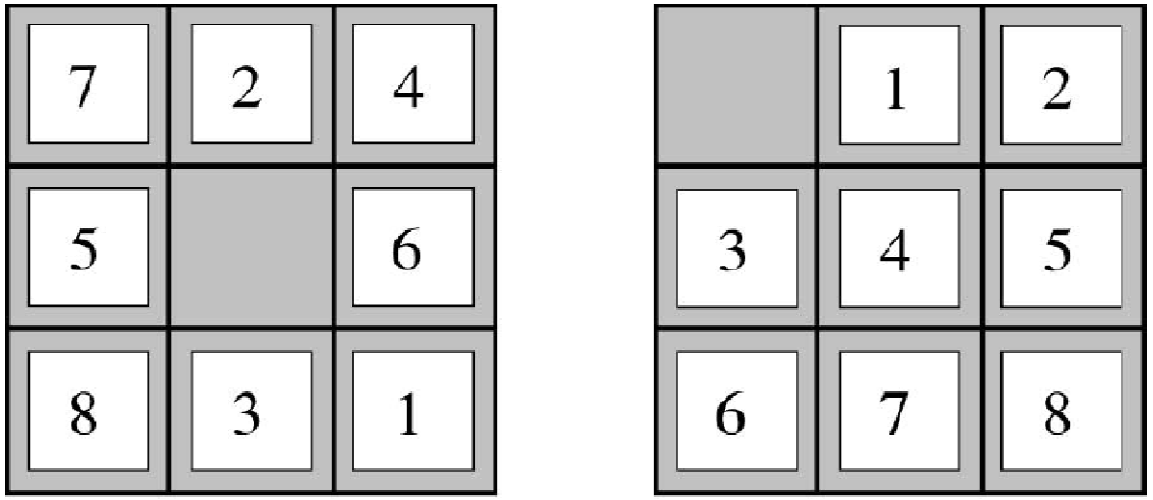
Estado Inicial



Estado Objetivo

¿Estados?
¿Acciones?
¿Test objetivo?
¿Coste del camino?

Ejemplo: el 8-puzle



Estado Inicial Estado Objetivo

¿Estados? Localizaciones completas de las piezas (ignorar las posiciones intermedias)

¿Nº estados?

¿Acciones? Mover el negro a la izquierda, derecha, arriba, abajo (ignorar los atascos, etc.)

¿Test objetivo? = estado objetivo (proporcionado)

¿Coste del camino? 1 por movimiento

Video Pacman/Roomba

UNIVERSITAT DE BARCELONA

- http://pacman.elstonj.com/index.cgi?dir=videos&num=&perpage=&section=

Algoritmos de búsqueda en árboles

Idea básica:

- exploración offline simulada del espacio de estado generado
- sucesores de estados ya explorados (también conocida como *expansión* de los estados)

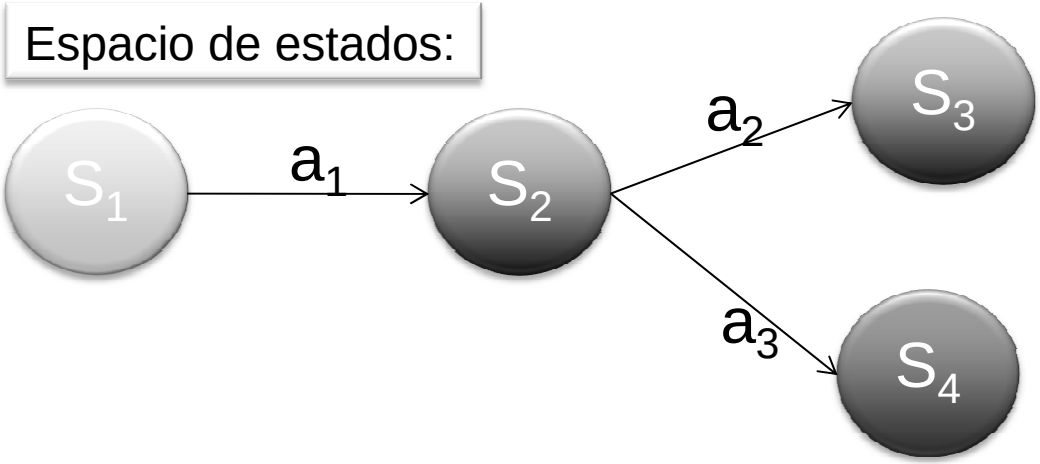
función BÚSQUEDA-ÁRBOLES(*problema*,*estrategia*) **devuelve** una solución o fallo

inicializa el árbol de búsqueda usando el estado inicial del *problema*

bucle hacer

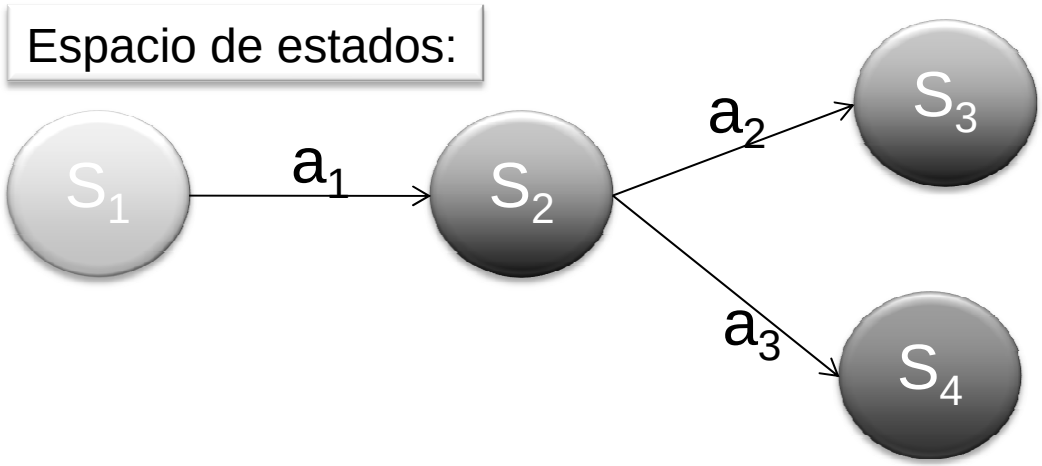
si no hay candidatos para expandir en el árbol de búsqueda **entonces devolver** fallo
escoger, de acuerdo a la *estrategia*, un nodo hoja para expandir
si el nodo contiene un estado objetivo **entonces devolver** la correspondiente solución
en otro caso expandir el nodo y añadir los nodos resultado al árbol de búsqueda

Ejemplo



- ¿Problema?
- ¿Búsqueda-árboles (problema, estrategia)?

Ejemplo: problema

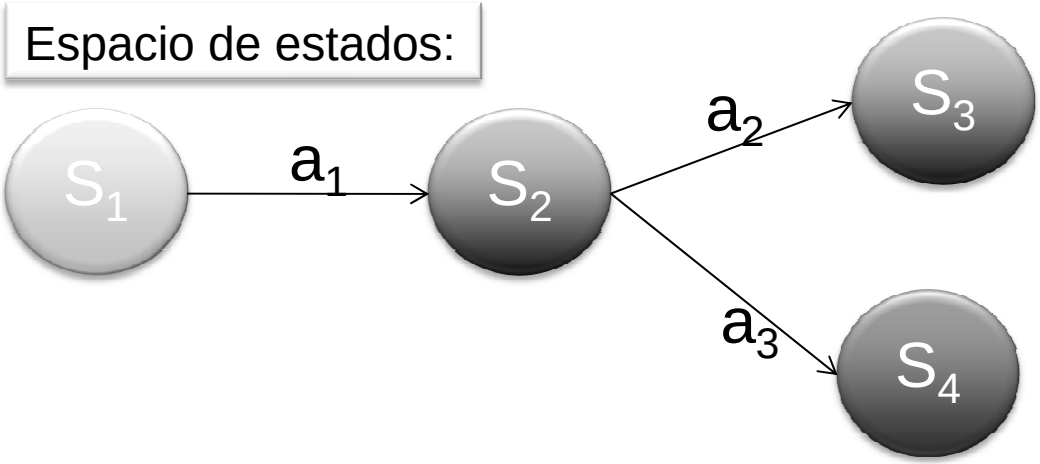


- Problema:

  estado inicial
  función sucesor :
  test objetivo
  coste del camino:

- Búsqueda-árboles (problema, estrategia)?

Ejemplo: problema



- Problema:

estado inicial s_1
función sucesor :
$S(s_1) = \{<a_1,s_2>\}$,
$S(s_2)=\{<a_2,s_3>, <a_3,s_4>\}$,
$S(s_3)=S(s_4)= \emptyset$
test objetivo, $x = s_4$
coste del camino:
(uniforme =1) $c(s_i,a_j,s_k)=1$ en las transiciones válidas

- Búsqueda-árboles (problema, estrategia)?

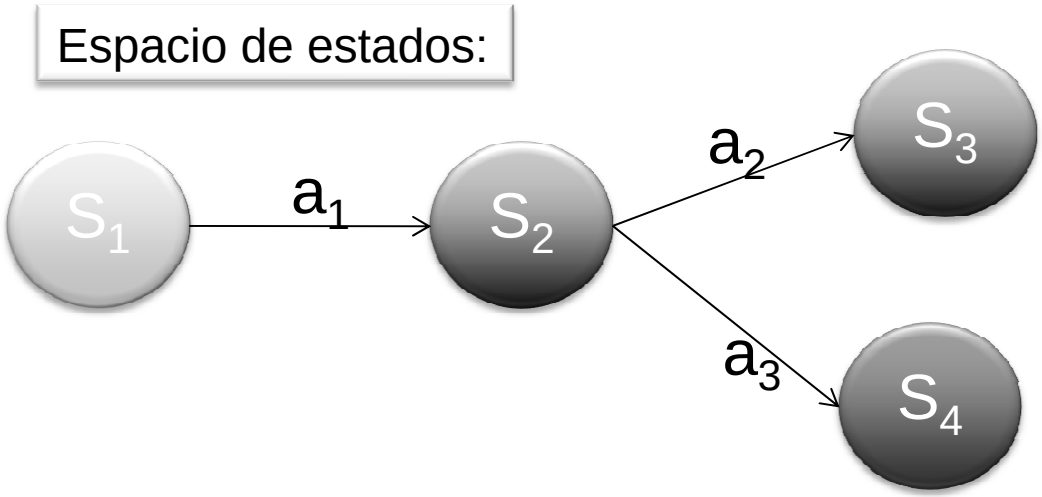
Ejemplo: ¿ejecución del algoritmo?

función BÚSQUEDA-ÁRBOLES(*problema*,*estrategia*) **devuelve** una solución o fallo

inicializa el árbol de búsqueda usando el estado inicial del *problema*

bucle hacer

- **si** no hay candidatos para expandir en el árbol de búsqueda **entonces devolver** fallo
- escoger, de acuerdo a la *estrategia*, un nodo hoja para expandir
- **si** el nodo contiene un estado objetivo **entonces devolver** la correspondiente solución
- **en otro caso** expandir el nodo y añadir los nodos resultado al árbol de búsqueda



Ejecución del algoritmo?

Ejemplo: ejecución del algoritmo

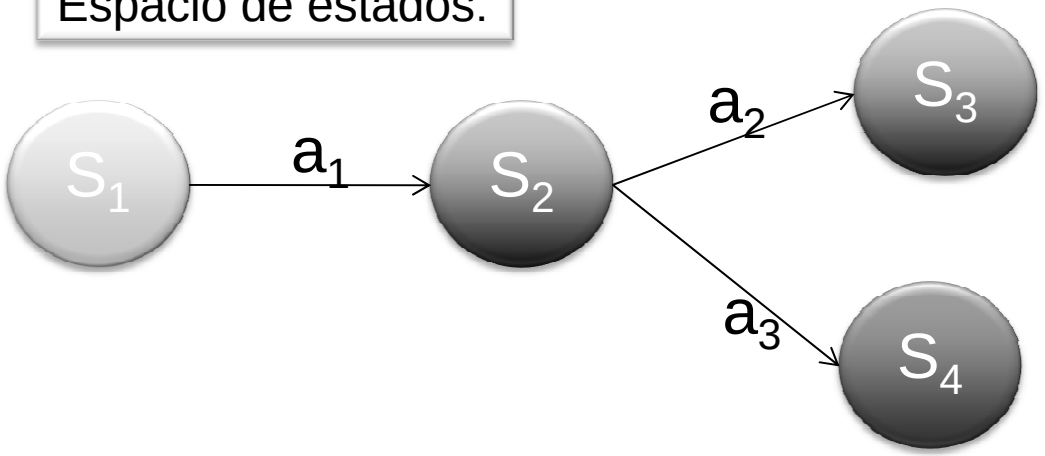
función BÚSQUEDA-ÁRBOLES(*problema*,*estrategia*) **devuelve** una solución o fallo

inicializa el árbol de búsqueda usando el estado inicial del *problema*

bucle hacer

- **si** no hay candidatos para expandir en el árbol de búsqueda **entonces devolver** fallo
- escoger, de acuerdo a la *estrategia*, un nodo hoja para expandir
- **si** el nodo contiene un estado objetivo **entonces devolver** la correspondiente solución
- **en otro caso** expandir el nodo y añadir los nodos resultado al árbol de búsqueda

Espacio de estados:



Inicialmente Arb= $\{s_1\}$
It 1: Arb != Ø
act = s_1
act != s_4 → Arb=$\{s_2\}$
It 2: Arb != Ø
act = s_2
act != s_4 → Arb=$\{s_3, s_4\}$
It 3: Arb != Ø
act = s_3
act != s_4 → Arb=$\{s_4\}$
It 4: Arb != Ø
act = s_4
act = s_4 → return solucion(s_4)

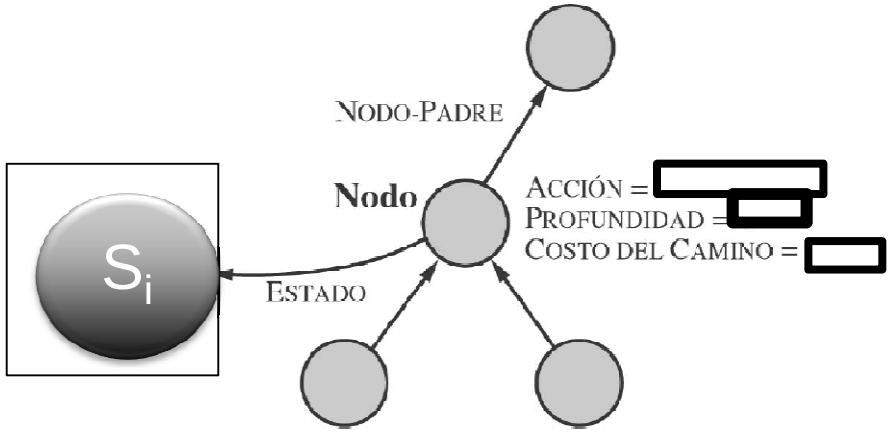
?

Implementación: estados frente a nodos

Un *estado* es una (representación de) una configuración física.
Un *nodo* es una estructura de datos que forma parte de un árbol de búsqueda que incluye *estado*, *padre*, *hijos*, *profundidad*, *coste del camino* $g(x)$.

¡Los estados no tienen padres, hijos, profundidad o coste del camino!

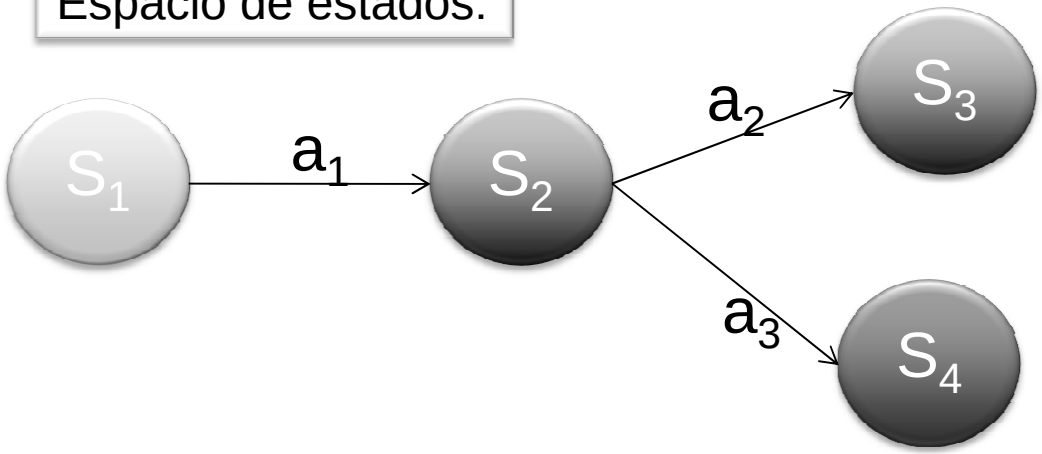
Notar que estado ≠ nodo del árbol de búsqueda:



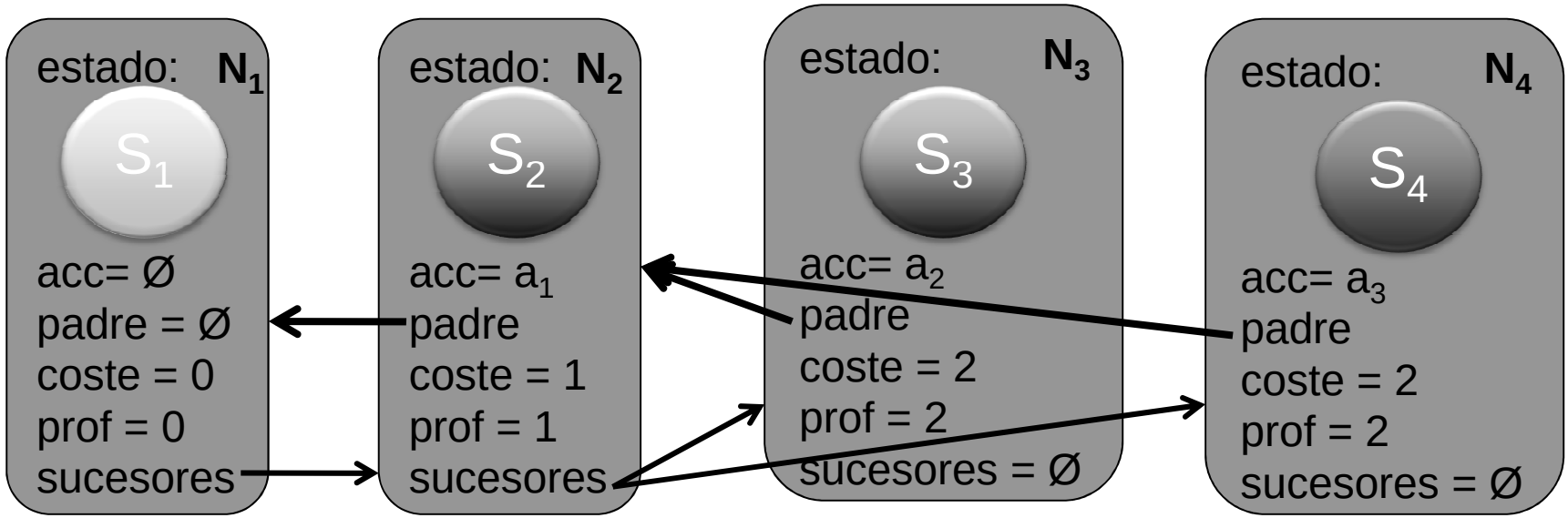
EXPANDIR un nodo consiste en utilizar la función sucesor (SUCESOR-EN) del problema para obtener los estados correspondientes y con ellos crear nuevos nodos (rellenando los distintos campos)

Implementación: estados frente a nodos

Espacio de estados:



Estructura de nodos (árbol de búsqueda):



Implementación: algoritmo general de búsqueda en árboles

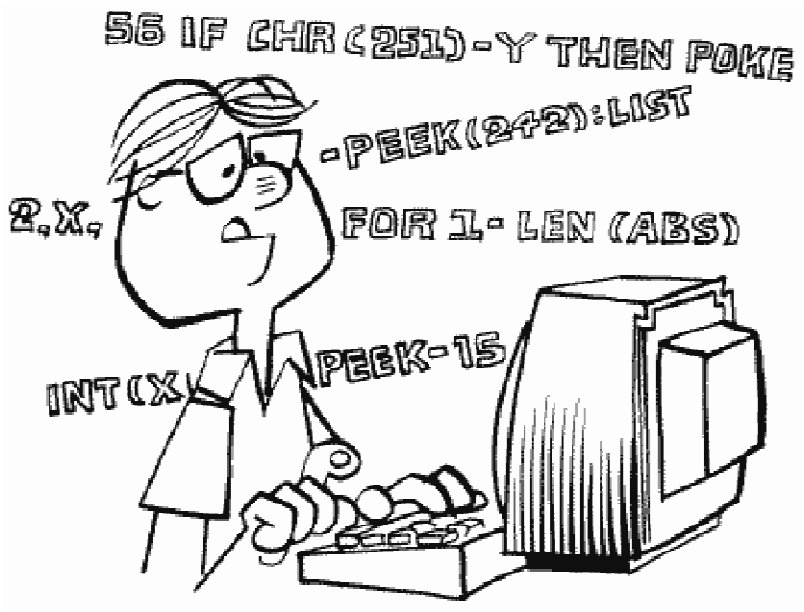
UNIVERSITAT DE BARCELONA

```
función BÚSQUEDA-ÁRBOLES(problema,frontera) devuelve una solución o fallo
    frontera ← INSERTA(HACER-NODO(ESTADO-INICIAL[problema]), frontera)
    bucle hacer
        si VACIA? (frontera) entonces devolver fallo
        nodo ←SACAR-BORRANDO-PRIMERO(frontera)
        si TEST-OBJETIVO[problema] aplicado al ESTADO[nodo] es cierto
            entonces devolver SOLUCION(nodo)
        frontera ← INSERTAR-TODO(EXPANDIR(nodo, problema), frontera)
```

```
función EXPANDIR(nodo, problema) devuelve un conjunto de nodos
    sucesores ← conjunto vacío
    para cada (acción, resultado) en SUCESOR-EN[problema](ESTADO[nodo]) hacer
        s ←un nuevo NODO
        ESTADO[s] ←resultado
        NODO-PADRE[s] ←nodo
        ACCIÓN[s] ←acción
        COSTE-CAMINO[s] ←COSTE-CAMINO[nodo] + COSTE-INDIVIDUAL(nodo.acción , s)
        PROFUNDIDAD[s] ←PROFUNDIDAD[nodo] + 1
        añadir s a sucesores
    devolver sucesores
```

Detalles de implementación

- Los nodos son conceptualmente caminos, pero es mejor representarlos con un estado, un coste, el valor de la última acción y una referencia al nodo padre.



Estrategias Básicas de Búsqueda

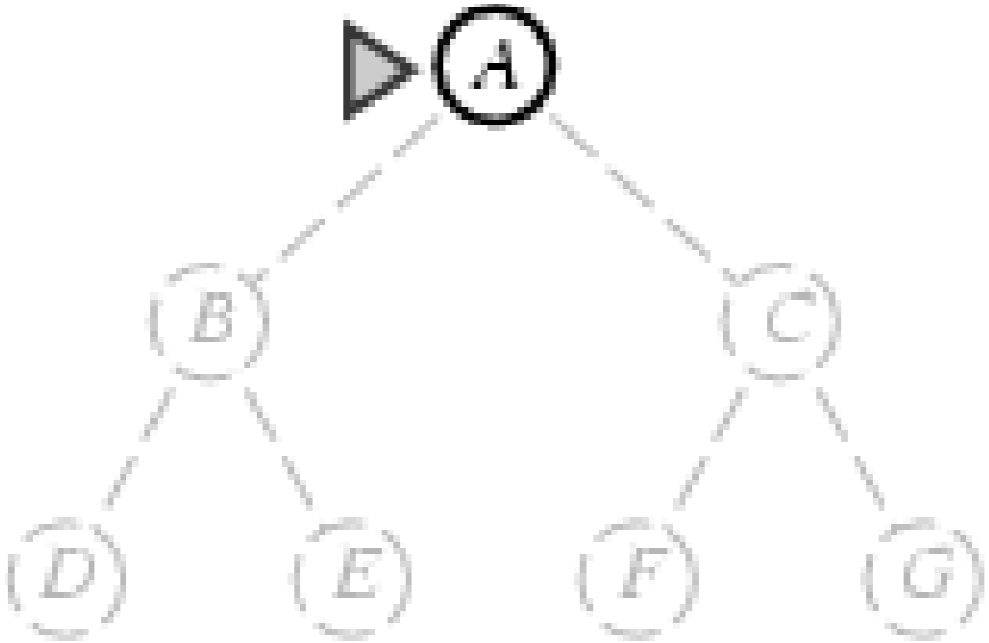
Vamos a ver estrategias que se denominan de búsqueda no informada (uninformed search) o de búsqueda ciega porque sólo usan la información de la definición del problema:

- búsqueda primero en anchura (BFS: Breadth First Search)
- búsqueda de coste uniforme
- búsqueda primero en profundidad (DFS: Depth First Search)
- búsqueda limitada en profundidad
- búsqueda por profundidad iterativa
- (búsqueda bidireccional)

Contrastan con las estrategias de búsqueda informada (informed search) o de búsqueda heurística que utilizan información del coste del estado actual al objetivo.

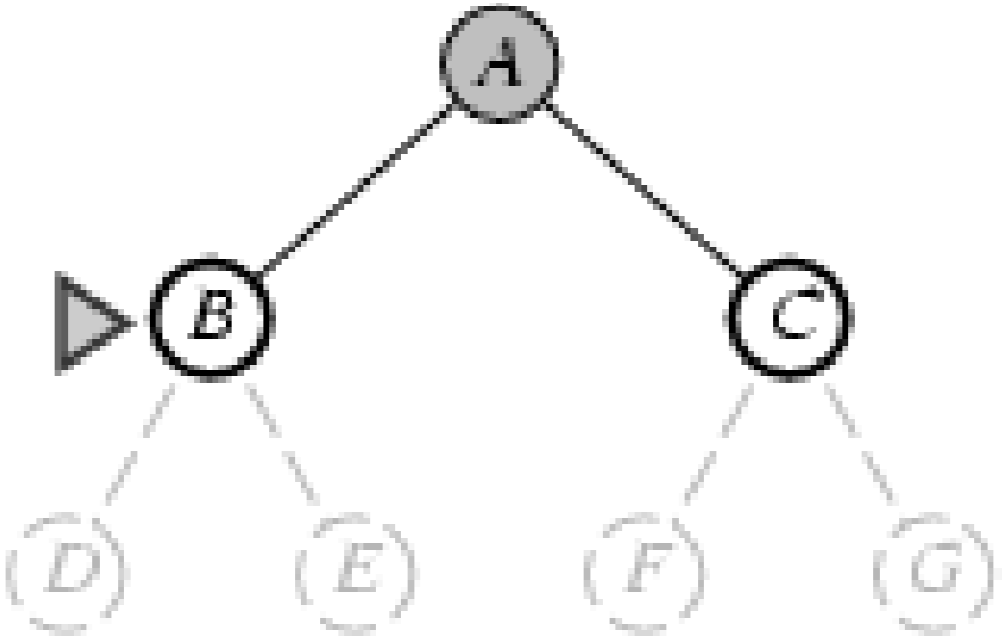
Búsqueda primero en anchura

- Expande el nodo no expandido cuya profundidad sea menor
- Implementación:
  - *La frontera* es una cola FIFO, es decir los nuevos sucesores se acumulan al final.



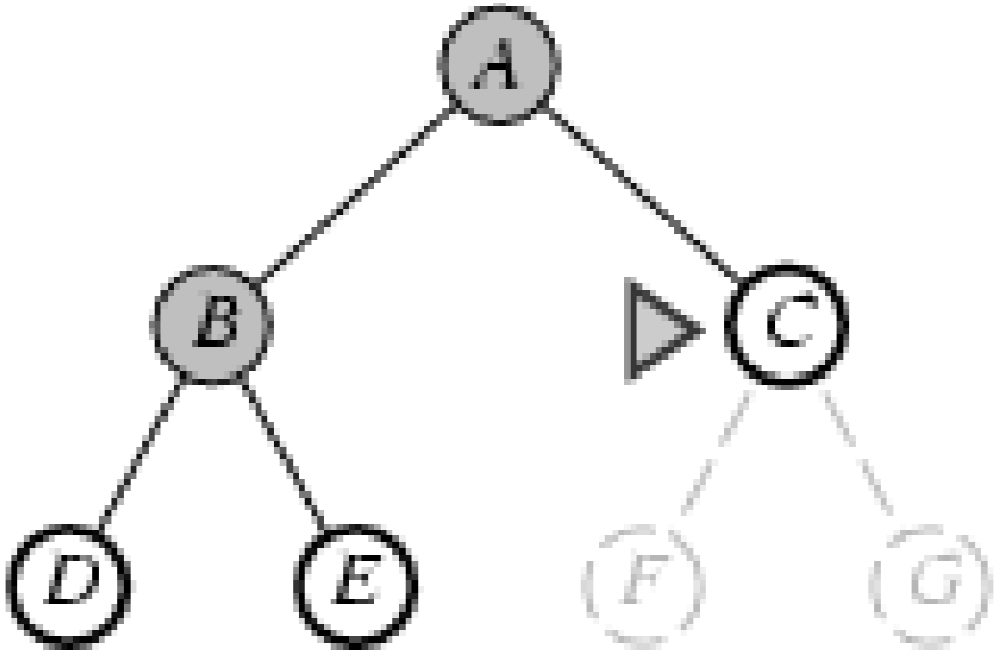
Búsqueda primero en anchura

- **Expande el nodo no expandido cuya profundidad sea menor**
- **Implementación:**
  - ***La frontera* es una cola FIFO, es decir los nuevos sucesores se acumulan al final.**



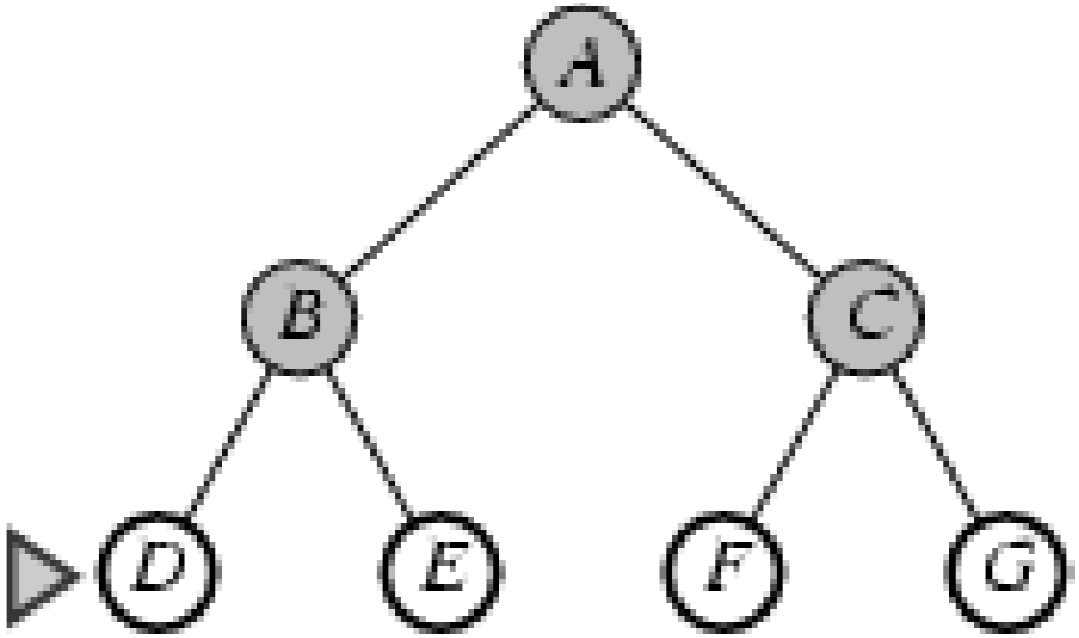
Búsqueda primero en anchura

- **Expande el nodo no expandido cuya profundidad sea menor**
- **Implementación:**
  - ***La frontera* es una cola FIFO, es decir los nuevos sucesores se acumulan al final.**



Búsqueda primero en anchura

- **Expande el nodo no expandido cuya profundidad sea menor**
- **Implementación:**
  - ***La frontera* es una cola FIFO, es decir los nuevos sucesores se acumulan al final.**



Evaluación de las Estrategias Básicas de Búsqueda

UNIVERSITAT DE BARCELONA

Una estrategia queda definida por el orden en que se expanden los nodos.

Vamos a evaluar las estrategias según cuatro aspectos:
Completitud —¿garantiza encontrar una solución si existe una?
Complejidad temporal —¿cuántos nodos deben expandirse?
Complejidad espacial —¿cuántos nodos deben almacenarse en memoria?
Optimalidad —¿garantiza encontrar la mejor solución (la de menor coste) si existen varias?

Las complejidades temporal y espacial se miden en términos de:
b — máximo factor de ramificación del árbol de búsqueda
d — profundidad de la solución de menor coste
m — profundidad máxima del espacio de estados (puede ser ∞)

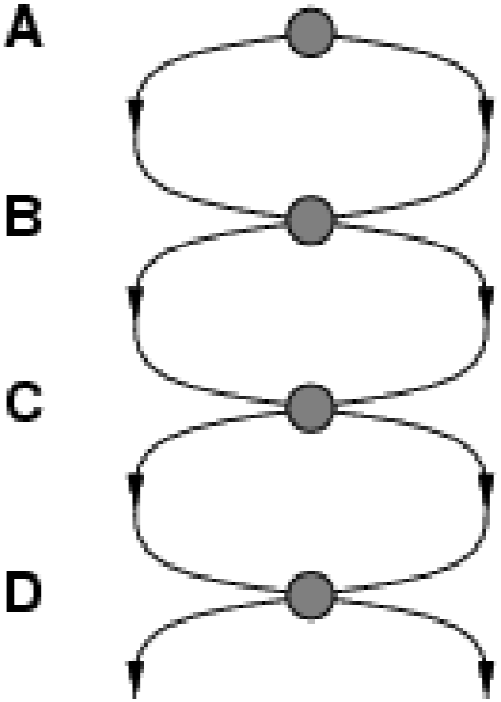
Propiedades de la búsqueda primero en achura

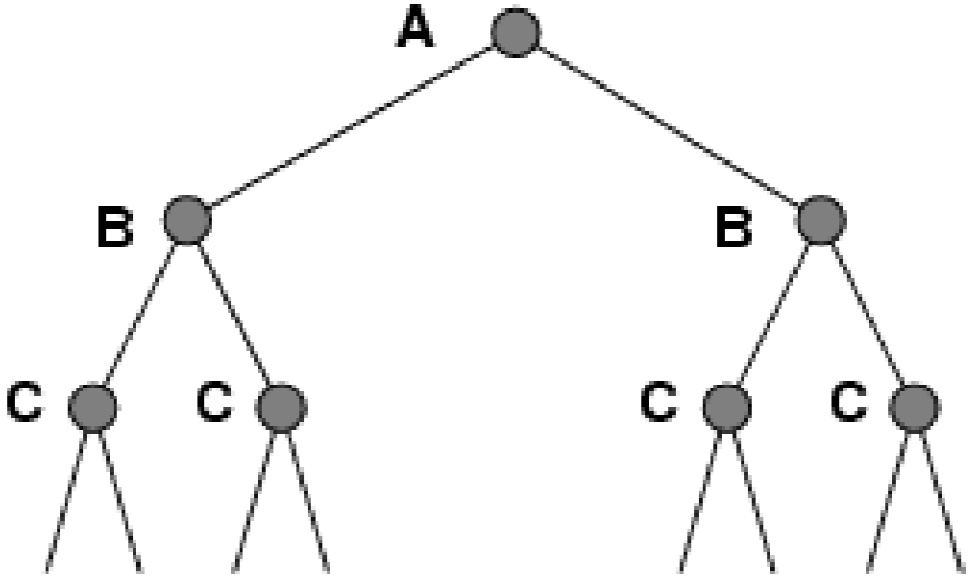
- Completa? Sí (si *b* es finita)
- Tiempo? $1+b+b^2+b^3+\ldots+b^d + b(b^d-1) = O(b^{d+1})$
- Espacio? $O(b^{d+1})$ (mantiene cada nodo en memoria)
- Óptima? Sí (si el coste es igual para todas las acciones)

- El espacio representa un problema mayor que el tiempo

Estados repetidos

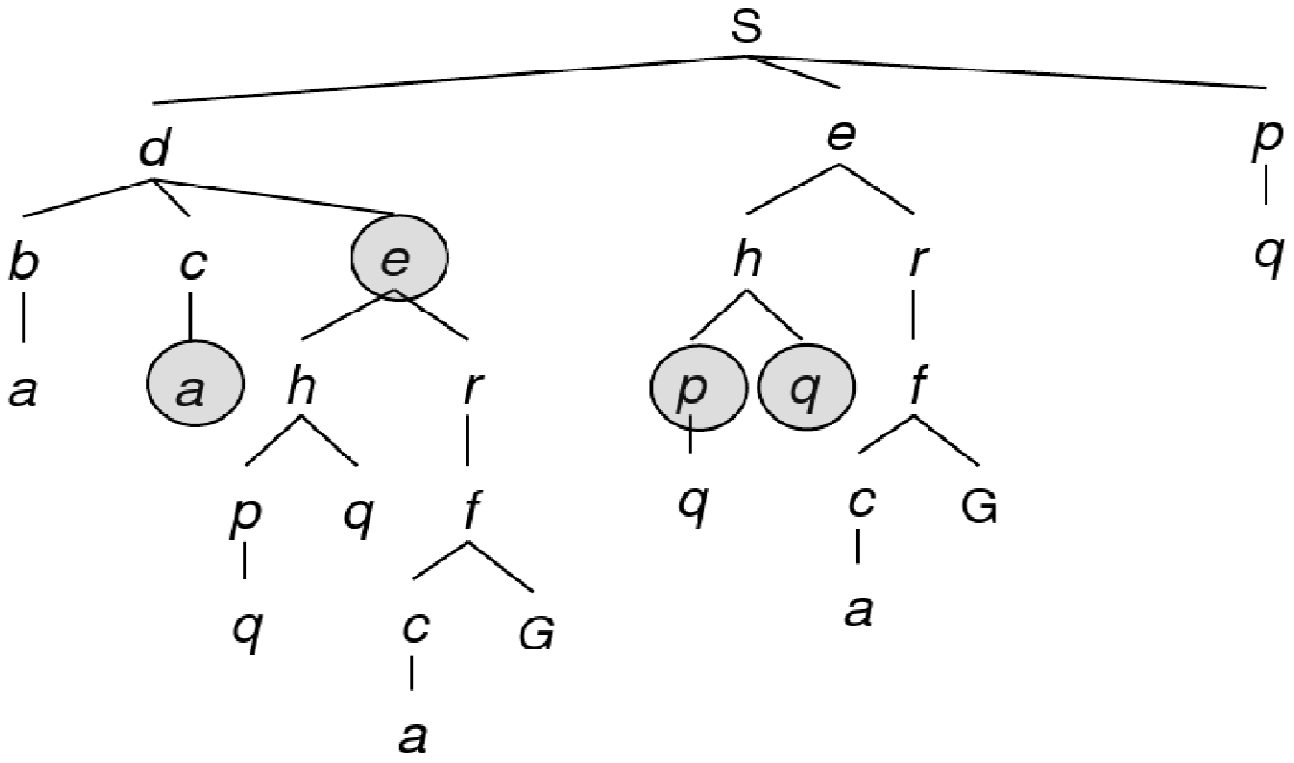
- Si no detectamos estados repetidos podemos convertir un problema lineal en un problema exponencial!!!





Búsqueda en grafos

- En una búsqueda primero en anchura no sería necesario expandir los nodos coloreados ¿por qué?



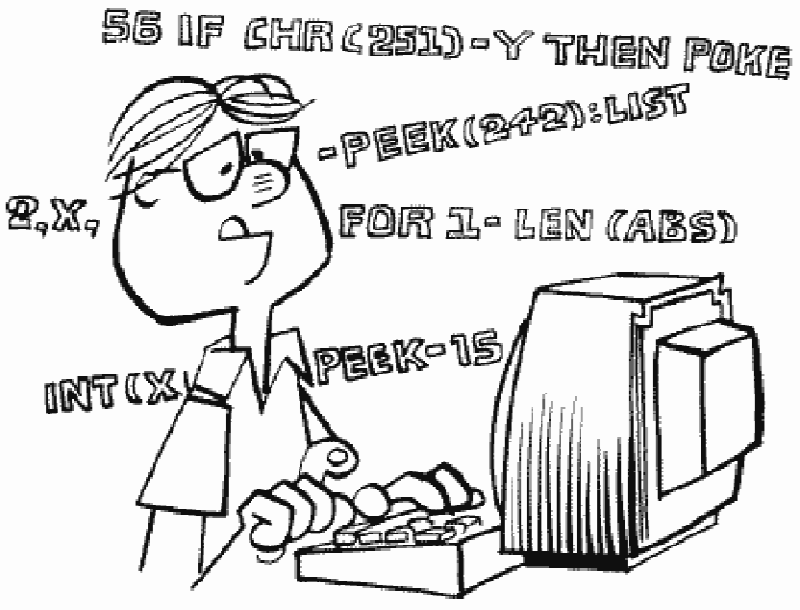
Algoritmos de búsqueda en grafos

Idea básica:

- Mecanismo de exploración igual que en árboles
- Evitar repeticiones mediante una lista de nodos cerrados (ya explorados)

```
función BÚSQUEDA-GRAFOS(problema,frontera) devuelve una solución o fallo
    cerrados ← un conjunto vacío
    frontera ← INSERTA(HACER-NODO(ESTADO-INICIAL[problema]), frontera)
    bucle hacer
        si VACIA? (frontera) entonces devolver fallo
        nodo ←SACAR-BORRANDO-PRIMERO(frontera)
        si TEST-OBJETIVO[problema] aplicado al ESTADO[nodo] es cierto
            entonces devolver SOLUCION(nodo)
        si ESTADO[nodo] no está en cerrados entonces
            añadir ESTADO[nodo] a cerrados
            frontera ← INSERTAR-TODO(EXPANDIR(nodo, problema), frontera)
```

Detalles de implementación

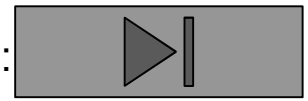


- Utilizad un dict o set para implementar la lista de estados cerrados

Video Pacman/Roomba

- http://pacman.elstonj.com/index.cgi?dir=videos&num=&perpage=&section=

- Más ejemplos de caracterización de problemas y espacio de estados:

Ir a18: 

  - El mundo de la aspiradora
  - El puzle de 8 piezas
  - Ver apartado 3.2 del libro para más ejemplos de problemas:
    - "Toy problems"
      - http://en.wikipedia.org/wiki/Eight_queens_puzzle
    - Problemas reales

Agentes reactivos

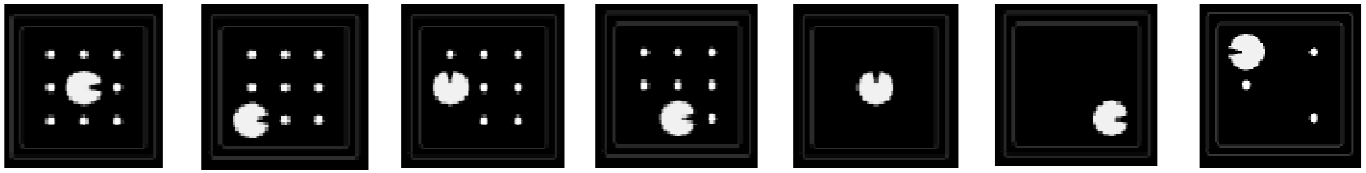
- Agentes reactivos
  - Escogen una acción basándose en su actual percepción y en su memoria.
  - No piensan más allá del primer paso.
- ¿Puede un agente reactivo ser racional?
- ¿Cómo de bueno es un agente reactivo Pacman?
  - Comer copos si están al lado y huir de los fantasmas
  - 31% de victorias contra fantasmas aleatorios
  - 5% de victorias en el Pacman original
  - 3% de victorias contra fantasmas reactivos en un escenario pequeño

Agentes basados en objetivos

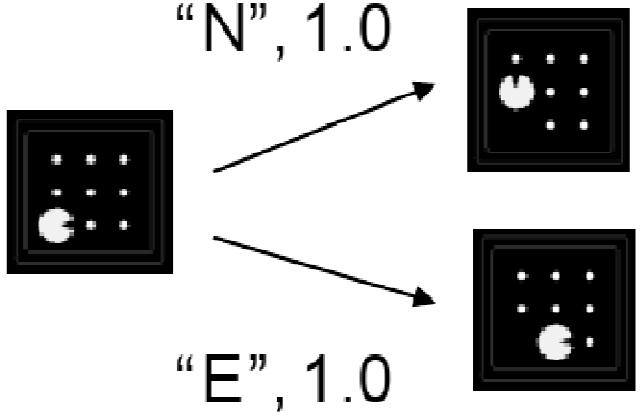
- Planifican con anterioridad
- Toman las decisiones en función de las consecuencias de sus acciones
- Deben tener un modelo de cómo el mundo evoluciona en respuesta a sus acciones

Problemas de búsqueda

- Un problema de búsqueda consiste en:
  - Un **espacio de estados**:



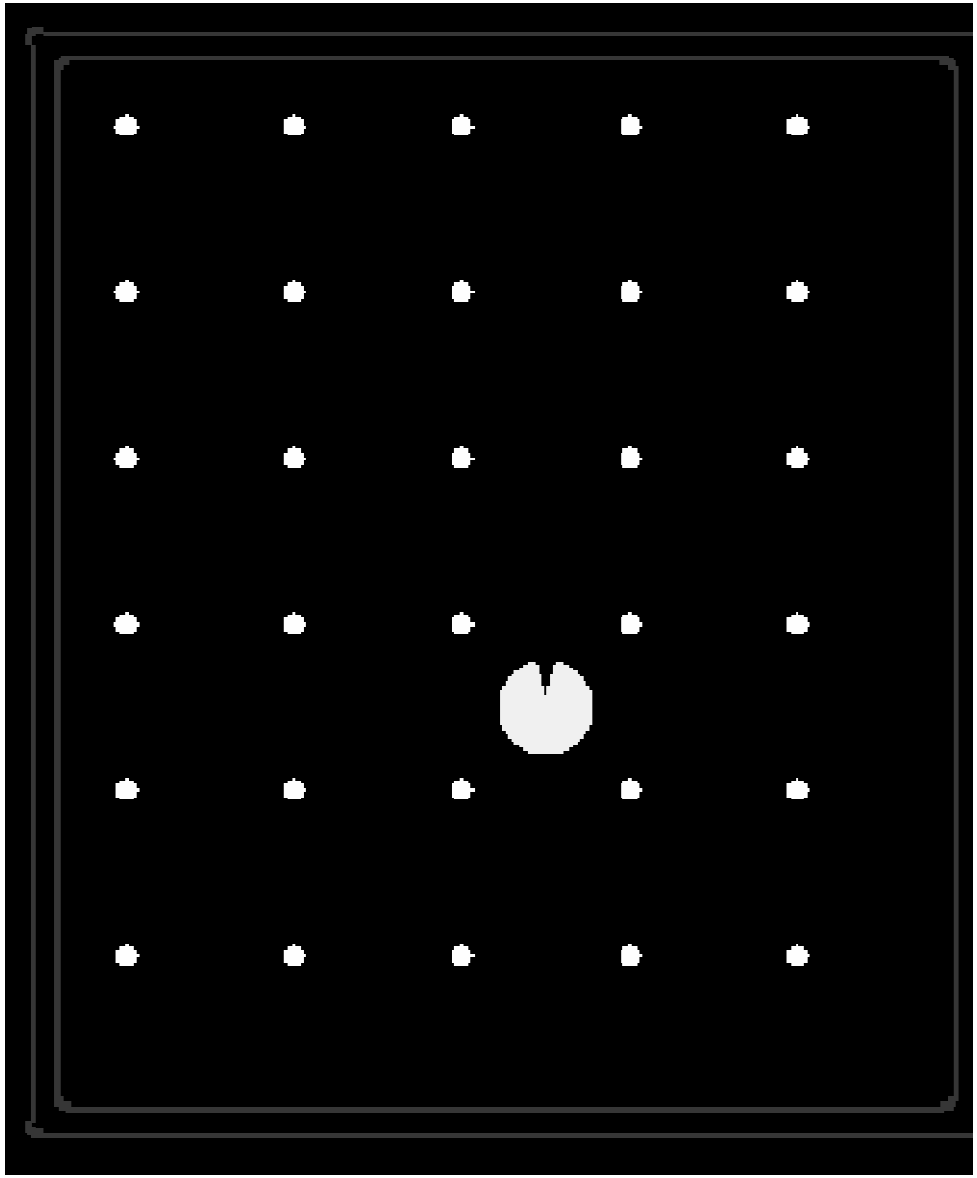
  - Una función **sucesor**:
  - Un **estado inicial** y un test que nos permita saber si hemos llegado al **objetivo**.



- Una **solución** es una secuencia de acciones que transforma un estado inicial en un estado objetivo.

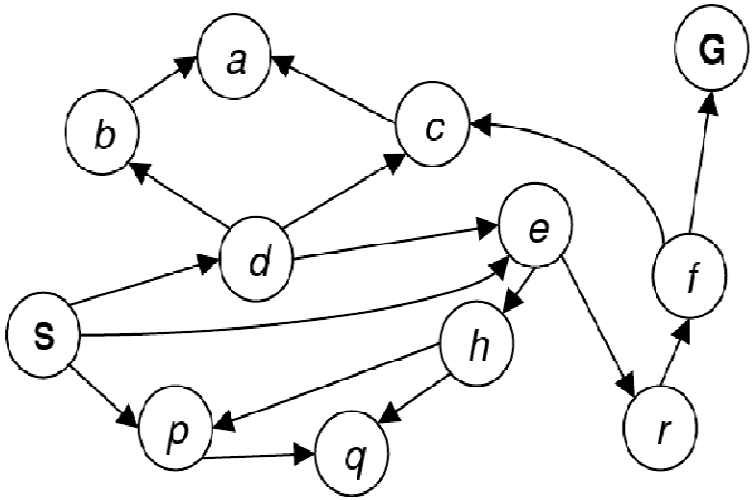
¿Cómo de grande es el espacio de búsqueda?

- Problema de búsqueda:
  - Comer toda la comida
- Posiciones posibles de Pacman:
  - 10x12
- Número de copos:
  - 30



Grafo del espacio de estados

- Todo problema de búsqueda tiene asociado un grafo de estados.
- La función sucesor se representa por arcos
- En contadas ocasiones se puede construir este grafo en memoria



Más Estrategias Básicas de Búsqueda

Seguimos viendo estrategias que se denominan de búsqueda no informada (uninformed search) o de búsqueda ciega porque sólo usan la información de la definición del problema:

- búsqueda primero en anchura
- búsqueda de coste uniforme
- búsqueda primero en profundidad
- búsqueda limitada en profundidad
- búsqueda por profundización iterativa
- (búsqueda bidireccional)

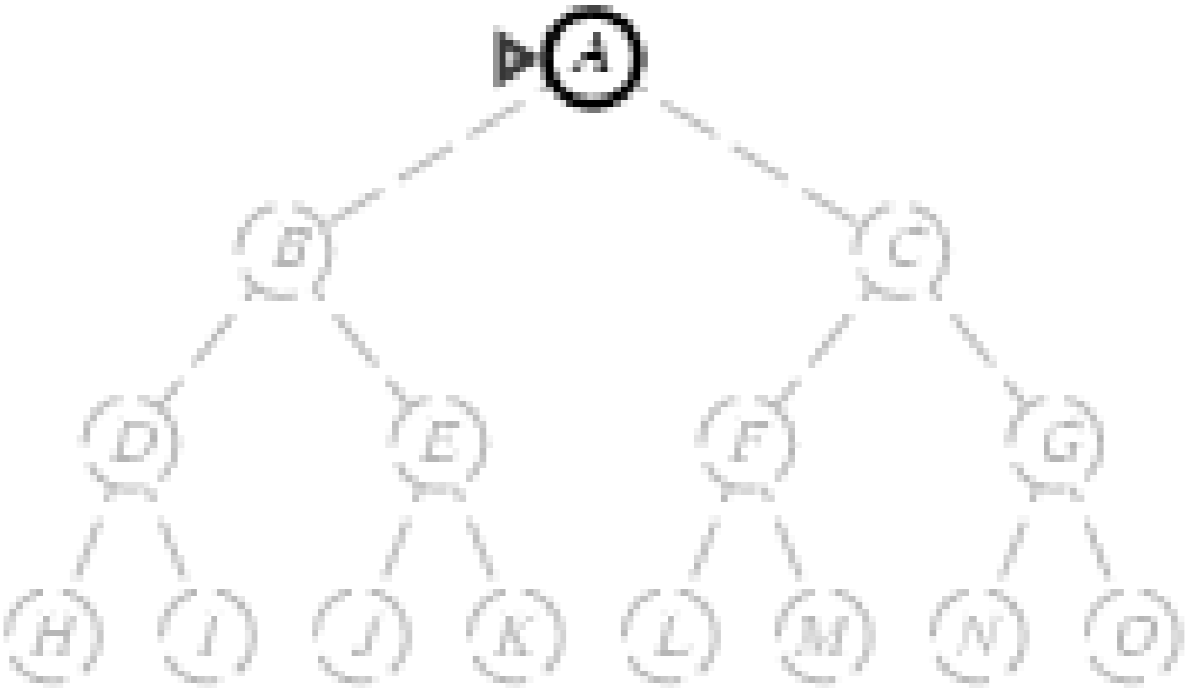
Contrastan con las estrategias de búsqueda informada (informed search) o de búsqueda heurística que utilizan información del coste del estado actual al objetivo.

Búsqueda de coste uniforme

- Expande el nodo no expandido cuyo coste sea menor
- Implementación:
  - La *frontera* es un cola ordenada por coste de camino
- Equivalente a la búsqueda primero en anchura si el coste de todos los pasos es igual.
- Completo? Sí, si el coste de cada paso es ≥ ε
- Tiempo? # de nodos con $g \leq$ coste de la solución óptima, $O(b^{ceiling(C^*/\varepsilon)})$ donde C^* es el coste de la solución óptima
- Espacio? # de nodos con $g \leq$ coste de la solución óptima $O(b^{ceiling(C^*/\varepsilon)})$
- Óptima? Sí. Los nodos se expanden en orden creciente de g(n)

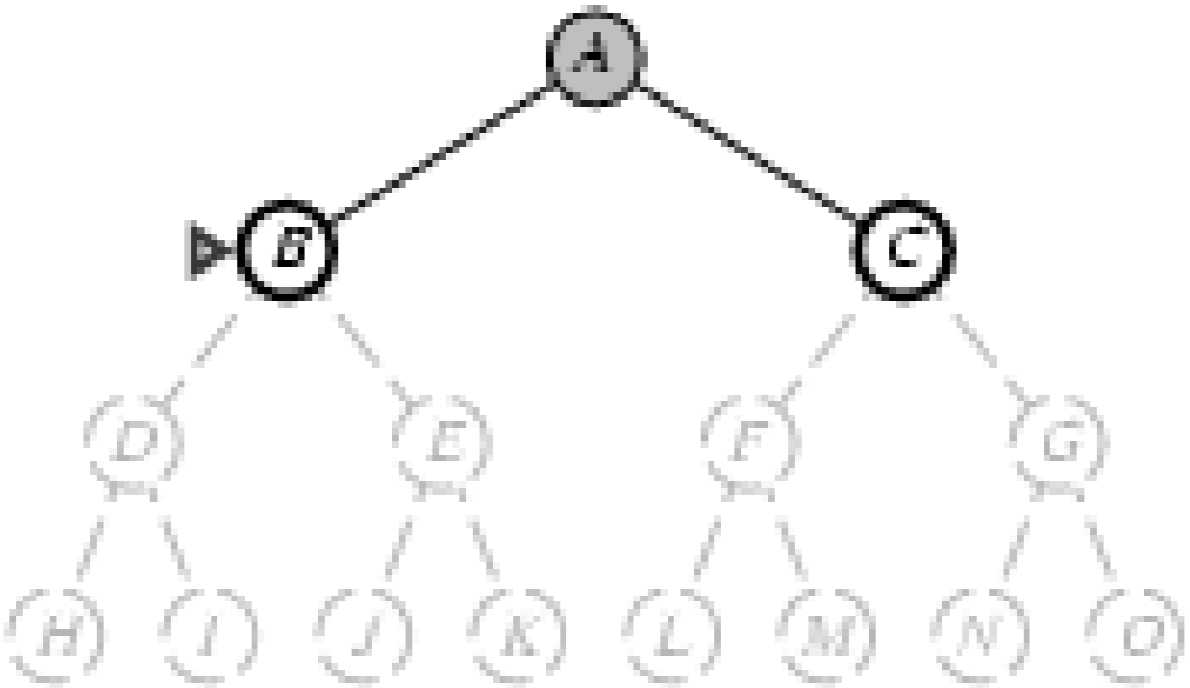
Búsqueda primero en profundidad

- Se expande el nodo más profundo no expandido aún
- Implementación:
  - La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio



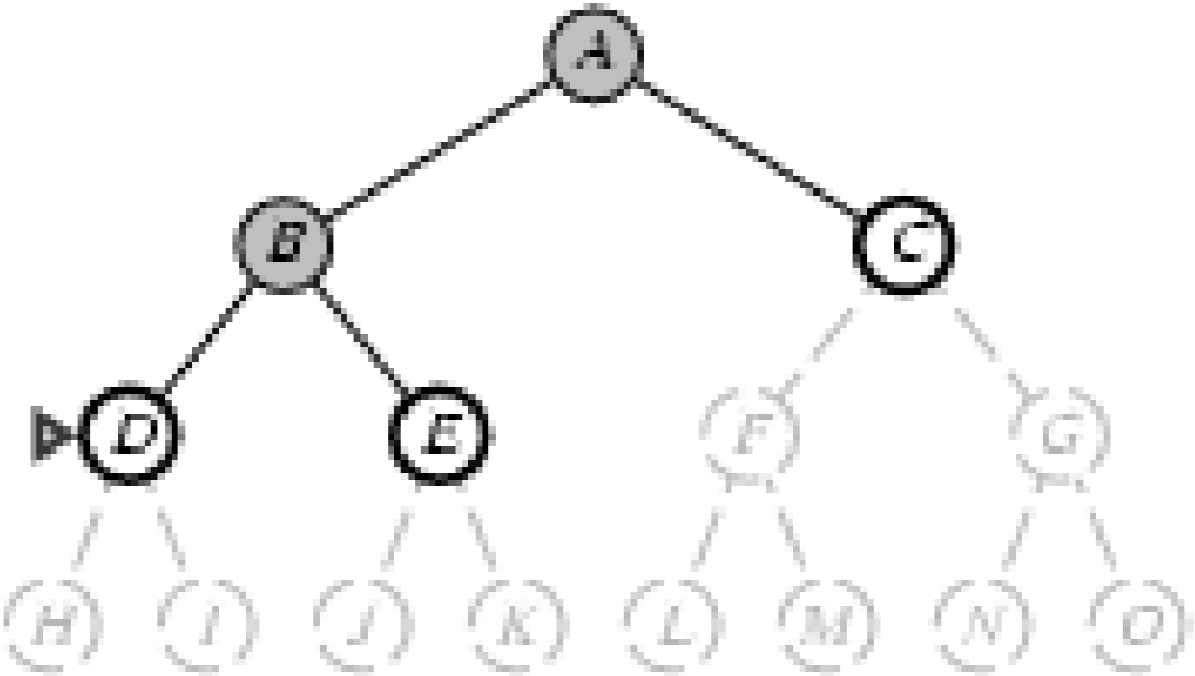
Búsqueda primero en profundidad

- Se expande el nodo más profundo no expandido aún
- Implementación:
  - La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio



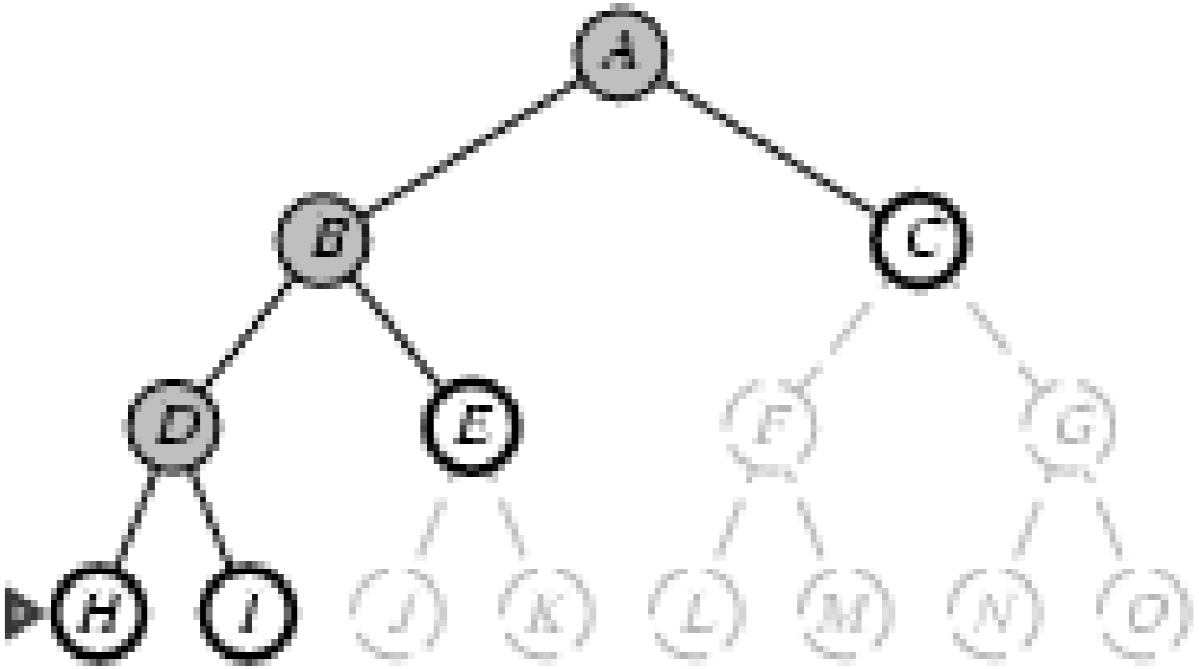
Búsqueda primero en profundidad

- Se expande el nodo más profundo no expandido aún
- Implementación:
  - La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio



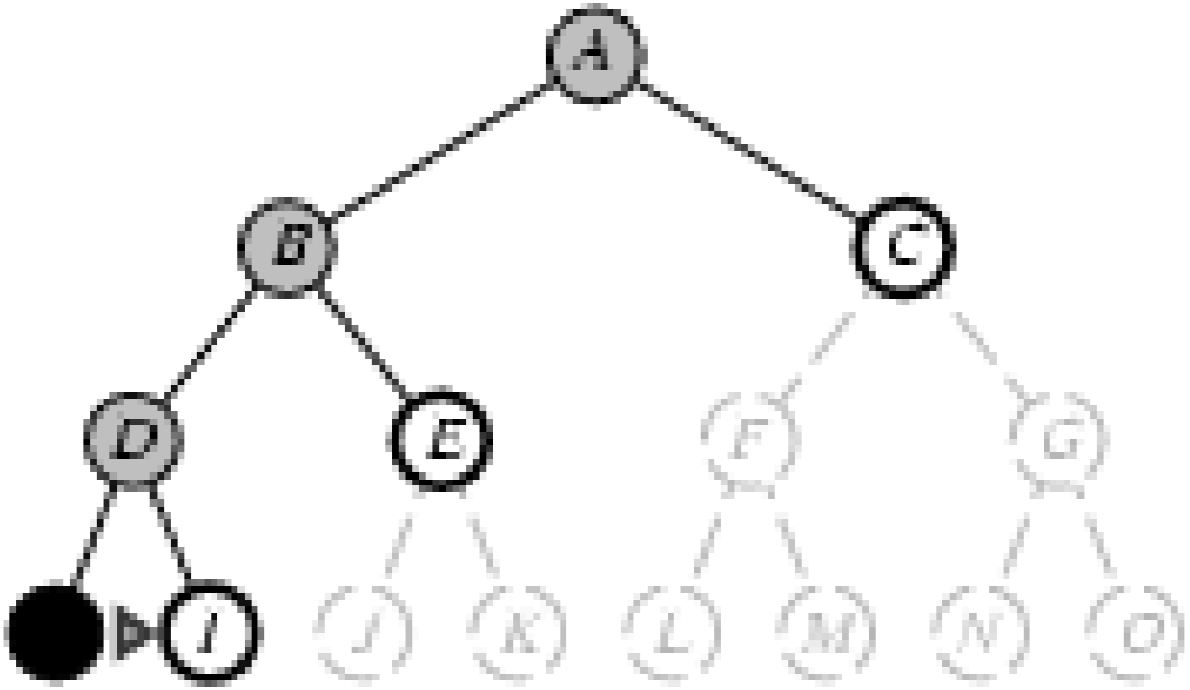
Búsqueda primero en profundidad

- Se expande el nodo más profundo no expandido aún
- Implementación:
  - La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio



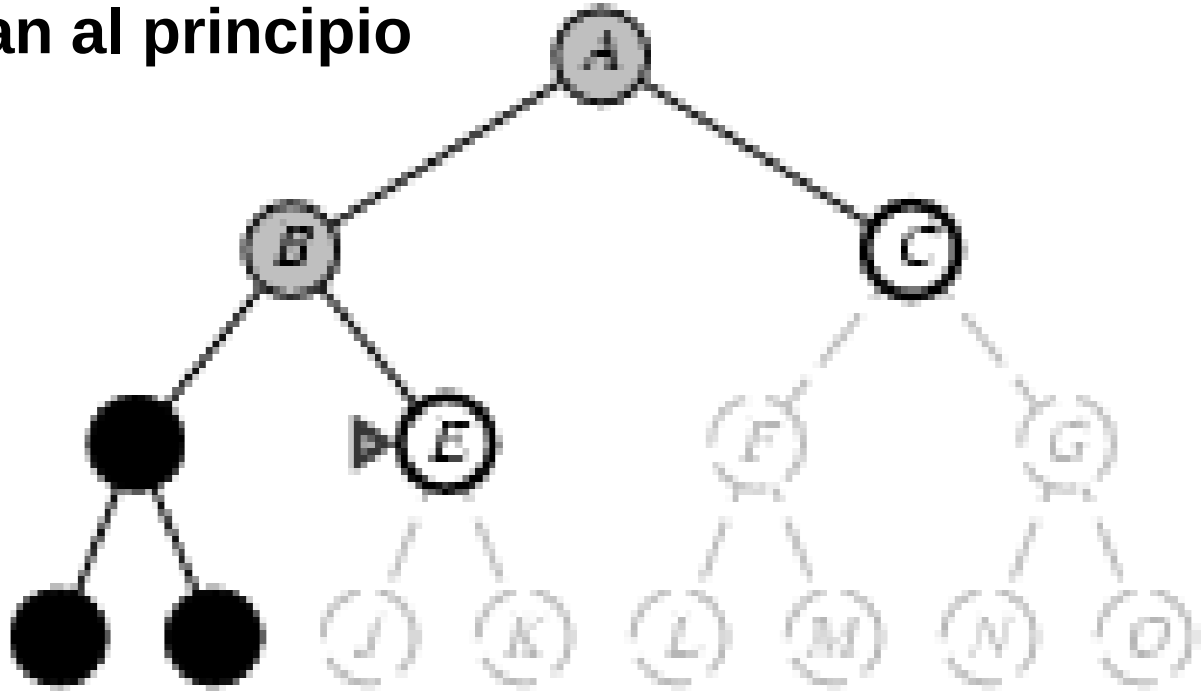
Búsqueda primero en profundidad

- Se expande el nodo más profundo no expandido aún
- Implementación:
  - La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio



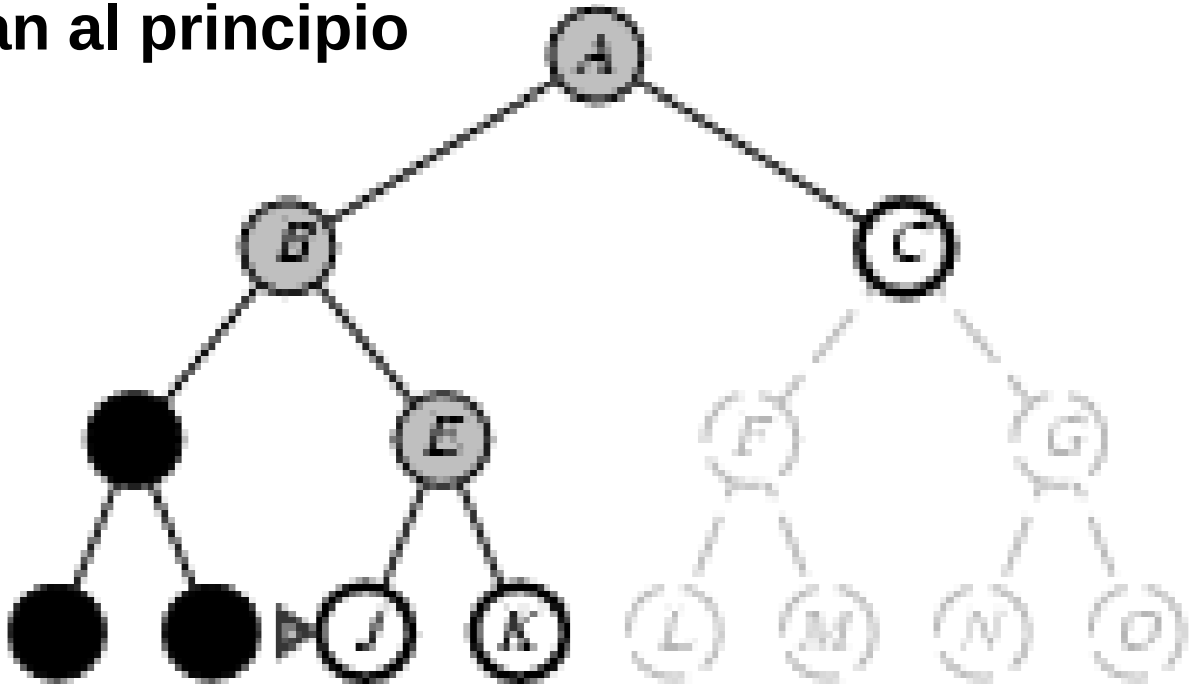
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



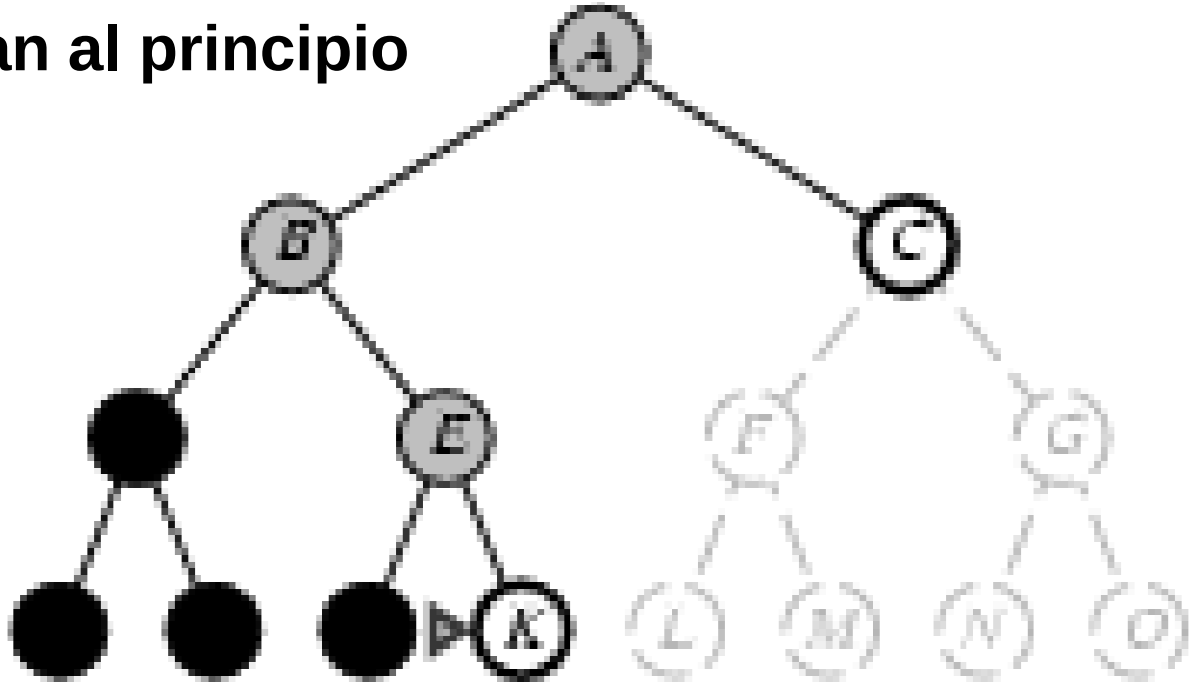
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



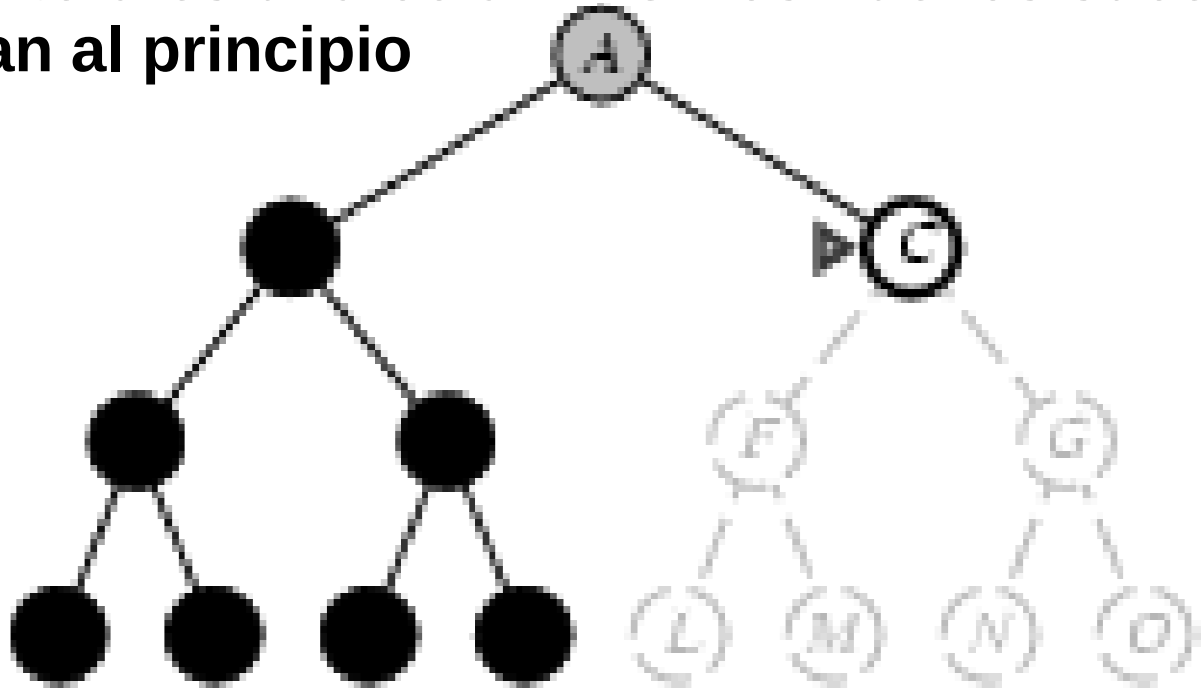
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



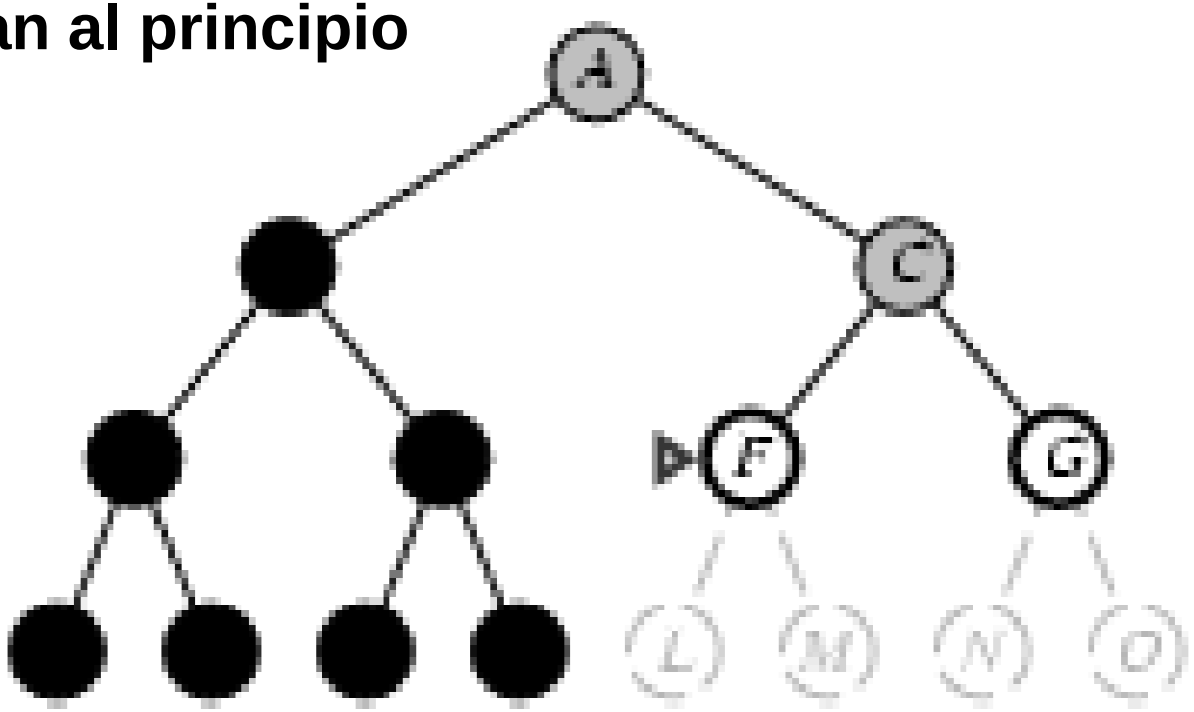
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



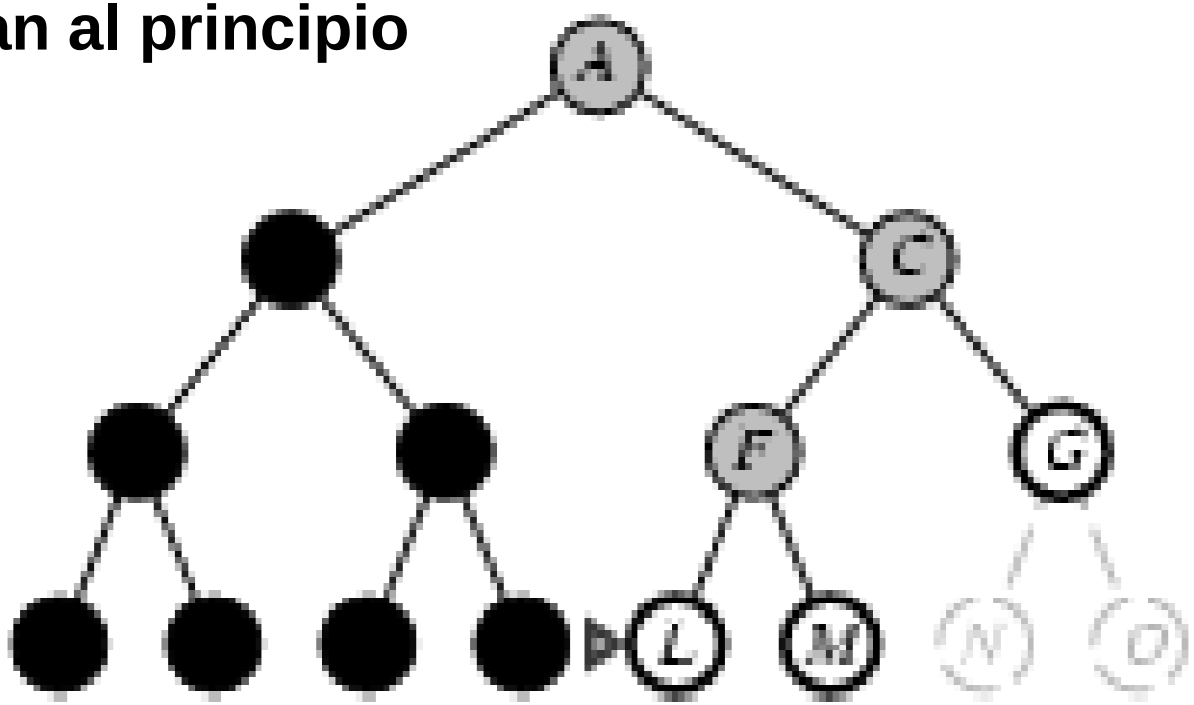
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



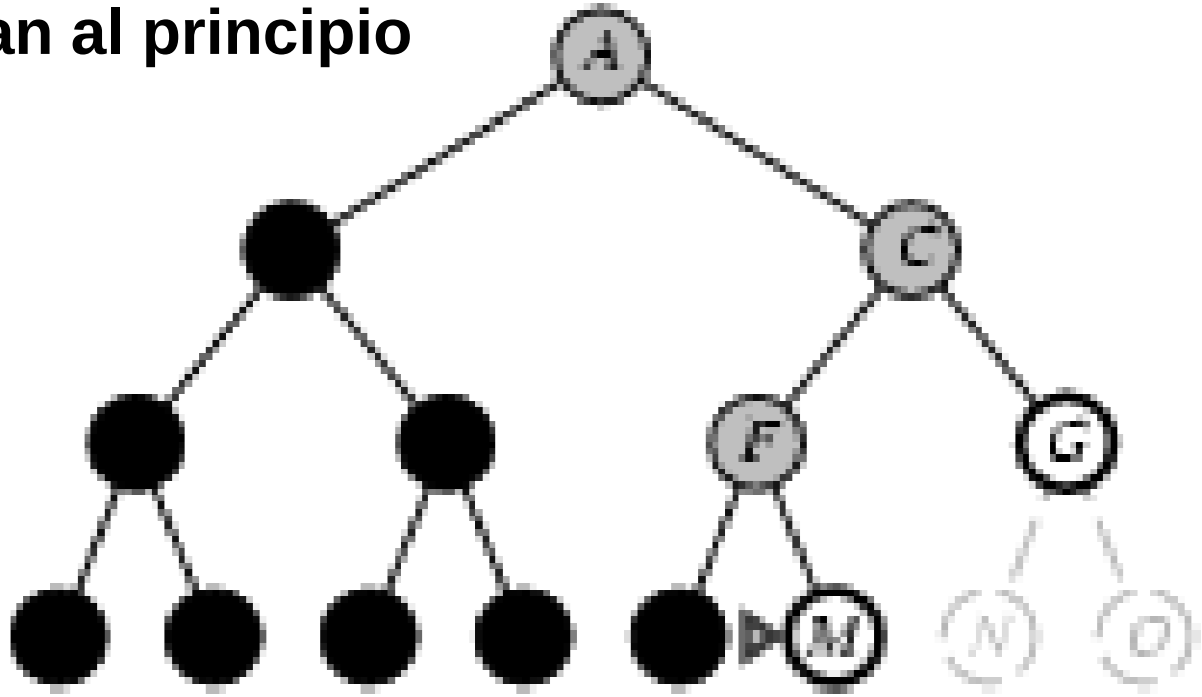
Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



Búsqueda primero en profundidad

- **Se expande el nodo más profundo no expandido aún**
- **Implementación:**
  - **La *frontera* es una cola LIFO: los nuevos sucesores se colocan al principio**



Propiedades de la búsqueda primero en profundidad

- Completa? No: falla en espacios de profundidad infinita, espacios con ciclos.
  - Si la modificamos para evitar estados repetidos en el camino actual → completa en espacios finitos
- Tiempo? $O(b^m)$: horrible si m es mucho mayor que d
  - Pero si las soluciones son densas, puede ser mucho más rápida que la búsqueda primero en anchura
- Espacio? $O(bm)$, es decir: espacio lineal!
- Óptima? No