# 2. Búsqueda no heurística (o desinformada)

## 2.1 Formulación de problemas en espacios de estados

Los problemas que se van a tratar en este tema son los más sencillos. Se trata de situaciones en las que el agente tiene unos fines o metas, que puede perseguir buscando sin incertidumbre en la representación que posea sobre un micro mundo. La representación consiste en un modelo formal del mismo (estructuras de datos sobre los que se pueda computar), que lo represente abstractamente en sus rasgos fundamentales (en relación con el problema) como un **espacio de estados** (estado que el agente puede identificar cuando está en ellos) donde uno o varios de los estados son **iniciales** o de partida, desde los cuales se ha de llegar o encontrar, mediante **transformaciones o cambios** **de estado** a uno o varios **estados meta, solución u objetivo.**

Los **estados meta** se identifican o valoran mediante algún o algunos **test o criterios** de que dispone el agente. Los **cambios de estado** se producen mediante la aplicación de **operadores de cambio de estado** o de **funciones sucesor** que determinan totalmente los estados a los que se puede pasar a partir de uno dado (estados sucesores del mismo), y que el agente conoce perfectamente, como conoce el **coste** (si existe) de dichos cambios.

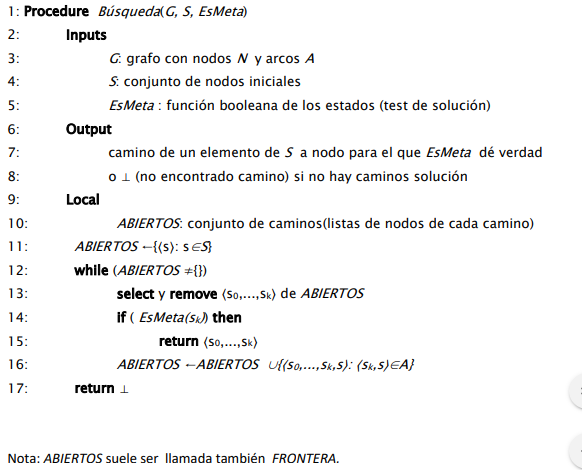
1. Definir el espacio de estados
2. Indicar el estado inicial o de partida
3. Indicar el/los estados meta
4. Especificar los operadores de transición o cambio de estado
5. Indicar el coste de los cambios
6. Test o criterios de identificación de soluciones

## 2.2 Ejemplos de problemas, concepto de solución

Por **resolución del problema** se entiende el proceso de determinar la serie de acciones de cambio de estado que hay que realizar (el camino a seguir) para llegar desde un estado inicial a uno o varios estados meta o solución, proceso sometido o no, según el contexto del problema, a la consecución de un coste mínimo **(solución óptima)** o a la obtención de estados que, de acuerdo con algún criterio de valoración dado, sean lo bastante satisfactorios.

Cuando un problema haya sido formulado, basta identificar cada estado con un nodo de un grafo, y cada uno de los operadores de cambio de estado con la arista que une los nodos correspondientes al estado de partida y al de llegada y se tendrá el problema convertido en uno de búsqueda en grados.

### 2.2.1 Procedimiento general de búsqueda en grafos



Los diversos métodos de búsqueda (anchura, profundidad, optimal o de coste uniforme etc.) se diferencian esencialmente en cómo se seleccione (línea 13 del algoritmo) el elemento de la lista ABIERTOS (tratándola como cola, como pila, como lista de prioridad según el coste acumulado, etc.)

## 2.3 Principales métodos de búsqueda no heurística

Una vez definida la formalización del problema hay que buscar una solución (que puede ser un estado o estados finales, o puede ser un camino desde el nodo de inicio al nodo meta) rastreando o explorando el grafo de búsqueda.

Idea general de la búsqueda:

1. Situarse en el estado inicial y comprobar si es solución
2. Si no, generar sus sucesores (estados a los que puede transitar) y:

* Almacenarlos en una lista
* Examinarlos como posibles metas, en el orden que indique el método que estamos siguiendo
* Para cada estado de la lista, examinarlo como posible meta, presentando si lo es o si no. Generar sus sucesores y poneros en la lista.

3. Repetir el proceso.

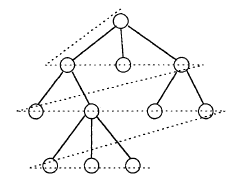
Los métodos de búsqueda se distinguen unos de otros por la manera de gestionar los nodos generados.

### 2.3.1 Búsqueda en anchura

La lista ABIERTOS se gestiona como una cola, donde los caminos se van extrayendo de la lista en el orden en que entraron a ella (FIFO), y sus nodos terminales se van examinando (mediante la función test EsMeta) para presentar al camino como solución (si lo fuera) o generar los sucesores del nodo terminal e incorporarlos a la lista ABIERTOS.

* **Inconveniente:** Si no se guarda la información de los nodos ya visitados, se encarece mucho el proceso porque se repiten muchos casos. Una variante del problema sería incluir una lista de cerrados en la cual se añadirían los nodos ya visitados.

La búsqueda en anchura es **completa,** es decir, si existe solución, la va a encontrar



**Algoritmo de búsqueda en amplitud:**

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado
2. Hasta que ABIERTA esté vacía o se encuentre una meta, realizar las siguientes acciones:

**2. 1.** Extraer el primer nodo de ABIERTA y llamarlo m.

**2.2.** Expandir m (generar todos sus sucesores). Para cada operador aplicable y cada forma de aplicación:

**1.** Aplicar el operador a m, obtener un nuevo estado y crear un puntero que permita saber que su predecesor es m.

**2.** Si el nuevo estado generado es meta, salir del proceso iterativo iniciado en 2.2 y devolver dicho estado.

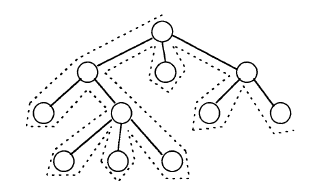
**3.** Incluir el nuevo estado al final de ABIERTA. (Una vez completado este proceso para todos los sucesores de m – cuando no se haya encontrado antes una meta – se continúa el proceso iterativo en el paso 2.)

### 2.3.2 Búsqueda en profundidad

La lista ABIERTOS se gestiona como una pila (LIFO). Existen variantes de la búsqueda en profundidad:

* **Profundidad acotada.** Se procede como en la búsqueda en profundidad, pero, hasta un límite preestablecido L, para la misma, es decir, “podando” las ramas del árbol de búsqueda con longitud L.
* **Profundidad iterativa.** Se establece una sucesión creciente de límites “L1 < L2 < L3 < … “y se repite, **desde el principio,** una serie de búsquedas en profundidad limitada Li, hasta que una de ellas encuentre la solución.

Nota: Tanto en la búsqueda en anchura como en la de profundidad y sus variantes, se da por supuesto que los costes de las aristas (es decir, de aplicar los operadores para transitar de unos nodos a otros) son siempre 1, de modo que los “gastos” hechos al recorrer los caminos dependen sólo del número de etapas de los mismos, que coincide con la profundidad de las ramas del árbol de búsqueda correspondiente.



**Algoritmo de búsqueda en profundidad:**

1. Crear una lista de nodos llamada ABIERTA y asignarle el nodo raíz, que representa el estado inicial del problema planteado.
2. Hasta que se encuentre una meta o se devuelva fallo, realizar las siguientes acciones:
3. Si ABIERTA está vacía, terminar con fallo; en caso contrario, continuar.
4. Extraer el primer nodo de ABIERTA y denominarlo m.
5. Si la profundidad de m es igual a lp (límite de profundidad), regresar a 2; en caso contrario, continuar.
6. Expandir m creando punteros hacia este nodo desde todos sus sucesores, de forma que pueda saberse cuál es su predecesor. Introducir dichos sucesores al principio de ABIERTA siguiendo un orden arbitrario.

(La “falta de orden” refleja el carácter no informado de este procedimiento.)

* Si algún sucesor de m es meta, abandonar el proceso iterativo señalado en 2, devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.
* Si algún sucesor de m se encuentra en un “callejón sin salida”, eliminarlo de ABIERTA. (Se continúa el proceso iterativo en el paso 2.)

**PREGUNTAS TEÓRICAS: CUESTIONES 2.1**

**1. Comprobar que la búsqueda en anchura es un caso particular de la búsqueda optimal.**

La búsqueda en anchura funciona de la misma manera que la búsqueda optimal aunque siendo todas las aristas constantes, en el caso de búsqueda en anchura damos por supuesto que el coste de las aristas para pasar de un estado a otro es 1.

**2. Comprobar, mediante ejemplos, que las búsquedas en profundidad y en profundidad acotada, pueden fallar, es decir, no encontrar ninguna meta, aunque las haya accesibles desde un estado inicial (es decir, estas búsquedas no son completas).**

En profundidad acotada o limitada se preestablece un límite, por tanto, puede que la solución esté a una profundidad mayor que el límite y no haya llegado a ella.

En búsqueda en profundidad se puede formar un bucle en un camino del árbol pudiendo no llegar a otros caminos y por tanto a la solución. Por ejemplo: tienes un robot que se puede mover para encontrar un objeto, si el robot se mueve en la misma dirección y el objeto no está en esa dirección entonces nunca encontrará el objeto.

**3. Razonar por que las búsquedas en anchura, profundidad iterativa y optimal no pueden fallar, es decir, si existe alguna meta accesible desde un estado inicial, acaban encontrándola (es decir, son completas).**

Las búsquedas en anchura, profundidad iterativa y optimal son completas ya que siempre acaban encontrando la solución.

En la búsqueda en anchura, siempre se va a dar con la solución ya que va buscando por niveles y en cada una de las ramas. Por tanto, siempre va a dar con uno o varios caminos correctos que le lleven a la solución o estado meta.

La profundidad iterativa dará siempre con la solución, ya que se van estableciendo una serie de límites crecientes que logran que no se formen bucles. De esta manera, aplica el método en profundidad en la rama hasta el nivel establecido por el límite, si no encontrase la solución mira en otra rama, así seguiría hasta que encontrase la solución.

La búsqueda optimal llega siempre a la solución ya que va eligiendo el camino con menor coste hasta que llegue a la solución.

**4. Comprobar que las búsquedas en anchura y optimal son óptimas, es decir que siempre encuentran primero el mejor camino (el de menor longitud=profundidad= gasto o menor gasto respectivamente)**

La búsqueda en anchura encuentra siempre el mejor camino debido a su forma de buscar el estado meta en un espacio de estados. La búsqueda en anchura empieza a buscar el estado de izquierda a derecha y nivel a nivel, de esta forma siempre encontrará el camino de menos longitud o profundidad.

La búsqueda optimal es óptima ya que hace lo mismo que la búsqueda en anchura, pero buscando los caminos con menor coste.

**5. Comprobar que, para un problema con soluciones, en la búsqueda en anchura, siempre hay en la lista ABIERTOS algún nodo de cada camino solución. ¿Ocurre lo mismo en la búsqueda en profundidad? ¿Y en la optimal?**

En la búsqueda en anchura siempre hay en lista ABIERTOS algún nodo de cada camino solución ya que consiste en una búsqueda por niveles, empezando a buscar por las ramas de la izquierda hacia las ramas de la derecha hasta que no haya más y luego baja de nivel siguiendo por las ramas de la izquierda hacia las de la derecha, así hasta que encuentre el estado meta o solución que estará en alguno de los caminos por los que ha ido buscando.

Sin embargo, en la búsqueda en profundidad no ocurre lo mismo, hay que recordar que la búsqueda en profundidad no es completa debido a que coge una rama y busca en ese camino pudiendo no existir en ese camino la solución.

En la búsqueda optimal no siempre hay en la lista ABIERTOS algún nodo de cada camino solución ya que depende del valor de los cambios de estado y puedes no coger el camino de la solución al tener mayor peso.

**6. Comprobar que si Li+1=Li+1 para cada i también la búsqueda en profundidad iterativa es óptima. En otro caso puede ser subóptima, es decir, puede encontrar una solución con un camino cuya longitud difiera de la del óptimo como máximo en Li+1 - Li para los Li y Li+1 correspondientes.**

Sería óptima ya que funcionaría totalmente igual que la búsqueda en anchura y por tanto, encontraría la solución más cerca del estado inicial, es decir encontraría el mejor camino para llegar a la solución.

**EJERCICIOS EXAMENES TEORIA**

**Dar ejemplos de problemas en los que sea preferible la búsqueda en profundidad sobre la búsqueda en anchura y al revés.**

Es preferible la búsqueda en profundidad sobre la búsqueda en anchura en problemas en el que el estado final se encuentre muy alejado del estado inicial, por tanto, la búsqueda en profundidad tardaría menos en encontrar esa solución que estaría a una gran profundidad. Este algoritmo será capaz de obtener soluciones que se encuentren a gran profundidad.

Por ejemplo, el algoritmo de búsqueda en anchura tendrá una estructura de nodos abiertos de tamaño O(rp), lo que agotará rápidamente la memoria impidiendo continuar la búsqueda, pero solo habiendo alcanzado una profundidad de camino de p. En cambio, el algoritmo en profundidad solo tendrá O(rp) nodos abiertos al llegar a esa misma profundidad. Esto quiere decir que, en problemas difíciles, la estrategia en profundidad será la única capaz de hallar una solución con un espacio de memoria limitado n.

Aquellos problemas donde la solución se encuentre muy cerca del estado inicial es preferible la búsqueda en anchura.

*Recordatorio: la búsqueda en profundidad es incompleta ya que nunca recorre algunos nodos.*

**T3Examen. Explique en qué se diferencian esencial los diversos métodos de búsqueda en espacio de estados**

La diferencia principal está en cómo se gestiona la prioridad de la lista abiertos, siendo:

-Búsqueda en anchura: cola (FIFO)

-Búsqueda en profundidad: pila (LIFO)

Además, la búsqueda de coste uniforme se diferencia en que los cambios de estados o transacciones tienen distintos valores, ya que en búsqueda en anchura y profundidad todos los cambios de estado tienen valor 1.

**T4Examen. ¿Es completa la búsqueda en profundidad? ¿Lo es en problemas con una cantidad finita de estados posibles? ¿Lo es la búsqueda en profundidad acotada? ¿Y en profundidad iterativa?**

La **búsqueda en profundidad no es completa** ya que puedes bajar hasta estados infinitos sin encontrar el estado meta.

(En problemas con una cantidad finita de estados posibles sería completa)

La búsqueda en **profundidad acotada no es completa** ya que “poda el árbol” hasta un límite preestablecido pudiendo no llegar a ninguna solución o estado meta.

La búsqueda en **profundidad iterativa** **es completa** ya que establece una sucesión creciente de límites L1 < L2 < L3 < … y se repite, desde el principio una serie de búsquedas en profundidad limitada Li hasta que una de ellas encuentre la solución.