

Semana 2 -Modelado de problemas y Búsqueda No Informada

. Modelado de problemas

Un problema de búsqueda se define por:

1. **Estado inicial:** punto de partida.
2. **Acciones posibles:** movimientos o transiciones válidas desde un estado.
3. **Modelo de transición:** define el estado resultante de aplicar una acción.
4. **Objetivo o prueba de objetivo:** condición que determina cuándo se ha encontrado una solución.
5. **Costo del camino (opcional):** valor asociado a alcanzar un estado.

Objetivos de aprendizaje

- Comprender cómo representar un problema para que pueda resolverse mediante búsqueda.
- Conocer los elementos básicos de un problema de búsqueda según el libro.
- Entender y aplicar los algoritmos de búsqueda no informada: **Búsqueda en Anchura (BFS)** y **Búsqueda en Profundidad (DFS)**.
- Comparar ventajas y desventajas de BFS y DFS.

Imagine un agente en la ciudad de Arad, Rumanía, disfrutando de un viaje de vacaciones. La medida de rendimiento del agente contiene muchos factores: desea mejorar su bronceado, mejorar su rumano, tomar fotos, disfrutar de la vida nocturna, evitar resacas, etcétera. El problema de decisión es complejo implicando muchos elementos y por eso, lee cuidadosamente las guías de viaje.

Ahora, supongamos que el agente tiene un billete no reembolsable para volar a Bucarest al día siguiente.

En este caso, tiene sentido que el agente elija el objetivo de conseguir Bucarest.

Las acciones que no alcanzan Bucarest se pueden rechazar sin más consideraciones y el problema de decisión del agente se simplifica enormemente. Los objetivos ayudan a organizar su comportamiento limitando las metas que intenta alcanzar el agente. El primer paso para solucionar un problema es la formulación del objetivo, basado en la situación actual y la medida de rendimiento del agente.

Un problema puede definirse, formalmente, por cuatro componentes:

- El estado inicial en el que comienza el agente. Por ejemplo, el estado inicial para nuestro agente en Rumanía se describe como **En(Arad)**.
- Una descripción de las posibles acciones disponibles por el agente. La formulación más común utiliza una **función sucesor**.
 - **Por ejemplo, desde el estado**

En(Arad), la función sucesor para el problema de Rumanía devolverá

{lr(Sibiu), En(Sibiu), lr(Timisoara), En(Timisoara), lr(Zerind), En(Zerind)}

- El espacio de estados forma un grafo en el cual los nodos son estados y los arcos entre los nodos son acciones. (El mapa de Rumanía que se muestra en la Figura 3.2 puede interpretarse como un grafo del espacio de estados si vemos cada carretera como dos acciones de conducir, una en cada dirección). Un camino en el espacio de estados es una secuencia de estados conectados por una secuencia de acciones.

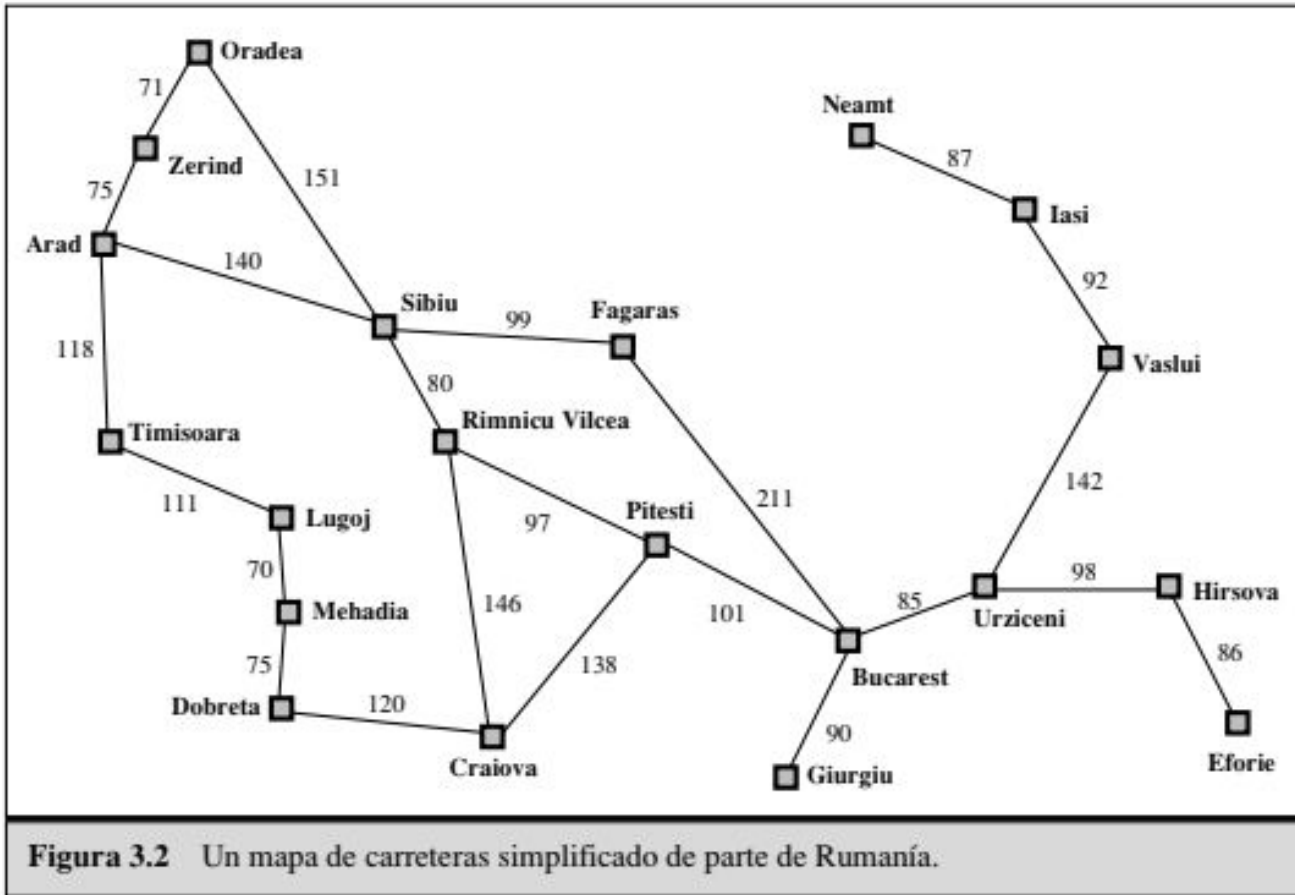


Figura 3.2 Un mapa de carreteras simplificado de parte de Rumanía.

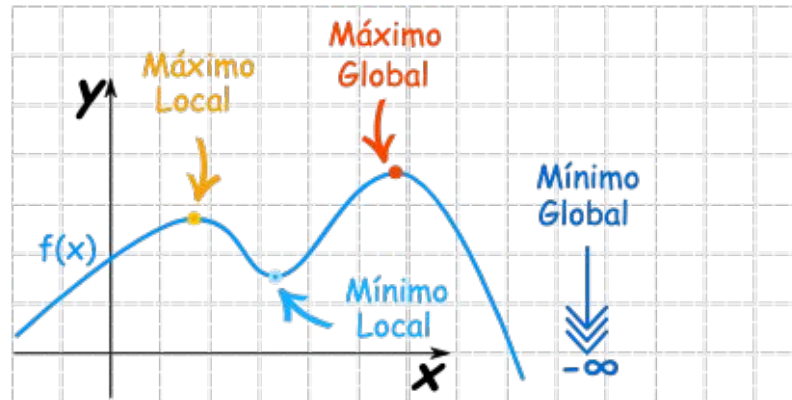
- El test objetivo, el cual determina si un estado es un estado objetivo. Algunas veces existe un conjunto explícito de posibles estados objetivo, y el test simplemente comprueba si el estado es uno de ellos. El objetivo del agente en Rumanía es el conjunto $\{En(Bucarest)\}$. Algunas veces el objetivo se especifica como una propiedad abstracta más que como un conjunto de estados enumerados explícitamente.

Por ejemplo, en el ajedrez, el objetivo es alcanzar un estado llamado «jaque mate», donde el rey del oponente es atacado y no tiene escapatoria.



- Una función costo del camino que asigna un costo numérico a cada camino. El agente resolvente de problemas elige una función costo que refleje nuestra medida de rendimiento. Para el agente que intenta llegar a Bucarest, el tiempo es esencial, así que el costo del camino puede describirse como su longitud en kilómetros.
- Suponemos que el costo del camino puede describirse como la suma de los costos de las acciones individuales a lo largo del camino. El costo individual de una acción a que va desde un estado x al estado y se denota por $c(x,a,y)$.
- Los costos individuales para Rumanía se muestran en la Figura 3.2 como las distancias de las carreteras. Suponemos que los costos son no negativos

- Los elementos anteriores definen un problema y pueden unirse en una estructura de datos simple que se dará como entrada al algoritmo resolvente del problema.
- Una solución de un problema es un camino desde el estado inicial a un estado objetivo. La calidad de la solución se mide por la función costo del camino, y una solución óptima tiene el costo más pequeño del camino entre todas las soluciones.



Instrucciones

En esta actividad modelaremos un problema de búsqueda usando un **laberinto 5×5**. El agente debe desplazarse desde la celda de inicio **S** hasta la celda objetivo **G**, evitando paredes marcadas con **X**. Cada movimiento tiene un costo de 1.

Laberinto

S	0	X	0	0
0	X	0	X	0
0	0	0	X	0
X	X	0	0	0
0	0	0	X	G

Una vez abordado el modelado recién ahora podemos intentar solucionar el laberinto pero cómo lo resolvemos si no sabemos donde está la meta , solo sabemos que encontraremos **G** pero ni idea de donde esta

S	0	X	0	0
0	X	0	X	0
0	0	0	X	0
X	X	0	0	0
0	0	0	X	G

1 ¿Qué es la búsqueda en Inteligencia Artificial?

- **Definición simple:**
Es un proceso para encontrar una secuencia de pasos que lleva desde un estado inicial hasta un estado objetivo.
- **Ejemplo cotidiano:**
Encontrar la salida de un laberinto, planear un camino en una ciudad, resolver un puzzle.
- **Idea clave:**
El agente explora posibilidades → evalúa → decide qué camino seguir.



¿Qué es la búsqueda en Inteligencia Artificial?

2 ¿Qué significa “no informada”?

- **No usa pistas adicionales** sobre qué tan cerca o lejos está el objetivo.
- Solo conoce:
 - **Estado inicial**
 - **Acciones posibles**
 - **Reglas de transición** (cómo cambia el estado al hacer una acción)
 - **Objetivo** (condición para ganar)
- **No sabe** cuál camino es más corto o rápido sin probarlo.
- 🔍 Se le llama también *búsqueda ciega*.



¿Qué es la búsqueda en Inteligencia Artificial?

3 Elementos de un problema de búsqueda

Según *Russell & Norvig*:

1. **Estado inicial** – punto de partida.
2. **Acciones** – qué movimientos puede hacer el agente.
3. **Modelo de transición** – cómo cambian los estados.
4. **Prueba de objetivo** – cómo sabe que llegó.
5. **Costo del camino** – valor numérico para medir eficiencia.

💡 *Ejemplo del laberinto:*

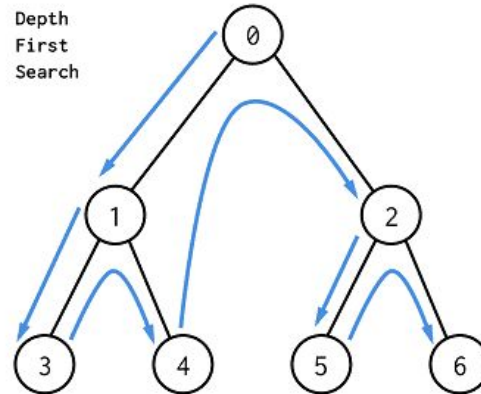
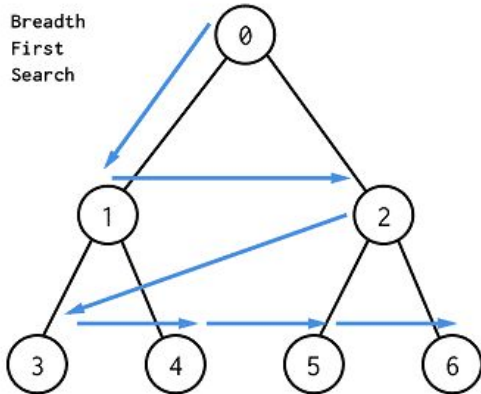
- Estado inicial: posición S.
- Acciones: norte, sur, este, oeste.
- Modelo de transición: mover a una celda adyacente libre.
- Prueba de objetivo: estar en G.
- Costo: 1 por movimiento.



¿Qué es la búsqueda en Inteligencia Artificial?

4 Tipos de búsqueda no informada

- **Búsqueda en Anchura (BFS)**
Explora primero lo más cercano.
- **Búsqueda en Profundidad (DFS)**
Explora primero un camino largo antes de retroceder.
- (Otros que veremos más adelante: búsqueda uniforme, profundidad iterativa, etc.)



Qué son BFS y DFS

- **BFS (Breadth-First Search / Anchura)**

Explora por **capas** (distancia creciente desde el inicio).





- Estructura: **cola FIFO**
- **Completo** (si hay solución y el espacio es finito)
- **Óptimo** si todos los pasos cuestan 1 (camino más corto)
- Complejidad: tiempo y memoria $O(b^d)$ (b: branching factor, d: profundidad de la solución)

- **DFS (Depth-First Search / Profundidad)**

Se va **lo más hondo posible** antes de retroceder.

- Estructura: **pila LIFO**
- **No es óptimo**, puede dar un camino más largo
- Completo solo en grafos finitos si controlas ciclos/ límites de profundidad
- Tiempo $\approx O(b^m)$, memoria $\approx O(bm)$ (m: profundidad máxima)

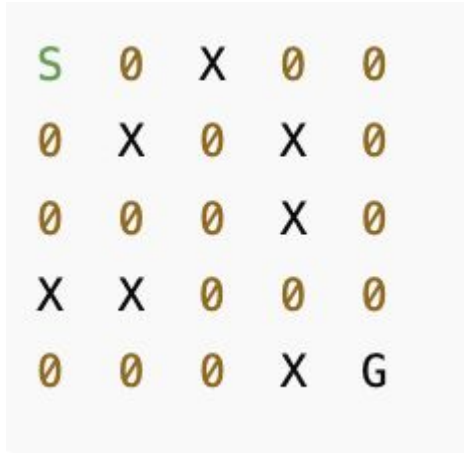
5 Ventajas y limitaciones

-  Sencillas de implementar.
-  Funcionan aunque no tengas información adicional del problema.
-  Pueden explorar muchos caminos inútiles.
-  Pueden tardar más que las búsquedas con heurísticas.

¿Qué es la búsqueda en Inteligencia Artificial?

- **Aplicación al laberinto**

- Mapa (S = inicio (0,0), G = objetivo (4,4), X = pared):



S	0	X	0	0
0	X	0	X	0
0	0	0	X	0
X	X	0	0	0
0	0	0	X	G

Resultado (camino encontrado)

Con costo uniforme (1 por movimiento), **BFS** devuelve un camino **óptimo** de longitud 8:

$(0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4) \rightarrow (4,4)$

Movimientos: S, S, E, E, S, E, E, S

Costo total: 8

DFS (con orden de acciones N, S, E, O y control de visitados) también llega a **(4,4)** en este caso con el mismo camino; recuerda que **no garantiza óptimo** en general.

Resultado (camino encontrado)

Con costo uniforme (1 por movimiento), **BFS** devuelve un camino **óptimo** de longitud 8:

$(0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4) \rightarrow (4,4)$

Movimientos: S, S, E, E, S, E, E, S

Costo total: 8

DFS (con orden de acciones N, S, E, O y control de visitados) también llega a **(4,4)** en este caso con el mismo camino; recuerda que **no garantiza óptimo** en general.

Actividad

El reto del río encantado

Hace mucho tiempo, en una aldea junto a un ancho y caudaloso río, vivía un granjero famoso por su ingenio. Un día, en el mercado, compró **un lobo** para proteger sus tierras, **una oveja** para obtener lana, y **una col** para cultivar su huerto.

Al volver a casa, llegó a la orilla del río... pero había un problema:

- Su barca era tan pequeña que **sólo podía llevarse a sí mismo y a uno de los tres** (lobo, oveja o col) en cada viaje.
- Además, el lobo y la oveja **no podían quedarse solos** sin supervisión: el lobo aprovecharía para devorarla.
- Y la oveja y la col **tampoco podían quedarse solas**: la oveja se comería la col



Problema del lobo, la cabra y la col

- El granjero debe cruzar un río llevando a los tres sin que se coman entre sí.
- Estados = qué hay en cada orilla.
- BFS/DFS exploran posibles secuencias de cruces.



Elementos del modelo

Estados. Representaremos en qué orilla está cada entidad. Usa binarios para que sea simple:

- 0 = orilla izquierda, 1 = orilla derecha.
- Estado = tupla $(G, L, O, C) = (\text{granjero}, \text{lobo}, \text{oveja}, \text{col})$.
- Estado inicial: $(0, 0, 0, 0)$ (todos a la izquierda).
- Objetivo: $(1, 1, 1, 1)$ (todos a la derecha).

Acciones. La barca siempre viaja con el granjero y puede llevar, además, a **exactamente uno** entre {lobo, oveja, col} o ir **solo**:

- G cruza solo.
- G+L, G+O, G+C.

Formalmente: elegir un subconjunto $M \subseteq \{G, L, O, C\}$ tal que $G \in M$ y $|M| \in \{1, 2\}$, y todos los de M están en la misma orilla que G antes del cruce.

Modelo de transición. Si el granjero está en s (0 o 1), la acción “mover M” produce el nuevo estado invirtiendo el bit de todos en M :

- $\text{next}[x] = 1 - \text{cur}[x]$ si $x \in M$
- $\text{next}[x] = \text{cur}[x]$ si $x \notin M$

Restricciones (estados inválidos). No puede quedar la oveja sola con su “comida” si el granjero no está presente:

- Prohibido L y O juntos en una orilla **sin** G en esa orilla.
Prohibido O y C juntos en una orilla **sin** G en esa orilla.

Chequeo práctico (para cada orilla $b \in \{0,1\}$):

- Si $L==b$ and $O==b$ and $G!=b \rightarrow$ inválido.
- Si $O==b$ and $C==b$ and $G!=b \rightarrow$ inválido.

Test de objetivo. $\text{state} == (1,1,1,1)$.

Costo. Uniforme: 1 por cruce (óptimo con BFS).

Propón una implementación en código python de Este problema y exponla el Lunes 18 de Agosto !

Grupo Max: 2 Integrantes