UBA - Carrera de Especialización en Inteligencia Artificial
Introducción a la Inteligencia Artificial - TP1
Algoritmos de búsqueda en Torre de Hanoi

Mgter. Ing. Alan Erik Perez Winter

24 de Marzo de 2024

1. Consigna

1. ¿Cuáles son los PEAS de este problema? (Performance, Environment, Actuators, Sensors)
2. ¿Cuáles son las propiedades del entorno de trabajo?
3. En el contexto de este problema, establezca cuáles son los: estado, espacio de estados, árbol de búsqueda, nodo de búsqueda, objetivo, acción y frontera.
4. Implemente algún método de búsqueda. Puedes elegir cualquiera menos búsqueda en anchura primero (el desarrollado en clase). Sos libre de elegir cualquiera de los vistos en clases, o inclusive buscar nuevos.
5. ¿Qué complejidad en tiempo y memoria tiene el algoritmo elegido?
6. A nivel implementación, ¿qué tiempo y memoria ocupa el algoritmo? (Se recomienda correr 10 veces y calcular promedio y desvío estándar de las métricas).
7. Si la solución óptima es $2^k - 1$ movimientos con *k* igual al número de discos. Qué tan lejos está la solución del algoritmo implementado de esta solución óptima (se recomienda correr al menos 10 veces y usar el promedio de trayecto usado).

2. Resolución

2.1. *PEAS*

- **Performance**: Es medida por la eficiencia de resolver el problema de las Torres de Hanoi, típicamente cuantificado por el número de movimientos o el tiempo que lleva resolver el algoritmo. El objetivo es el de mover todos los discos desde un poste inicial al poste objetivo, mientras se siguen las reglas del juego.
- **Eviroment**: En este problema, consiste en el número de postes y discos; e.g. 4 discos y tres postes. Sobre los postes se posicionan los discos de diferentes tamaños, agrupados en orden ascendente desde la parte superior hasta la inferior. Las reglas del entorno (*enviroment)* son dos: La primera dice que solo se puede mover de a un disco a la vez. La segunda dice que un disco solo puede ser colocado encima de otro más grande o un poste vacía.

- **Actuators**: Son los medios por los que el agente interactúa con el entorno para alcanzar los objetivos. En este problema, los actuadores son las acciones que el agente puede ejecutar para mover discos entre postes. Por otro lado, la única acción que puede realizar el agente es la de mover un disco de un poste a otra, siguiendo con las reglas del juego establecidas.

- **Sensors**: Proveen al agente de información acerca del estado del entorno. En este problema, los sensores proveen información acerca de la configuración actual de los discos en los postes. El agente necesita saber tres cosas. En primer lugar, la disposición actual de los discos en los postes. En segundo lugar, cuales discos son más grandes o chicos que otros. En tercer lugar, cuales postes tienen espacio disponible para recibir discos.

2.2. Propiedades del entorno

El problema de la Torre de Hanoi tiene varias propiedades de entorno que afectan como un agente interactúa con ellas. La comprensión de estas propiedades ayudan a diseñar apropiadamente algoritmos y heurísticas para resolver, el problema en cuestión, de manera eficiente usando técnicas de inteligencia artificial. Dichas propiedades se describen a continuación:

- **Discreta**: El problema de la Torre de Hanoi es representado en un entorno discreto. Cada acción (mover un disco de un poste a otra) resulta en un cambio completo de estado.

- **Observable**: El entorno es totalmente observable. El agente puede fácilmente percibir el estado actual del problema, a partir de una simple observación de la configuración de los discos en los postes.

- **Determinista**: Los resultados de la Torre de Hanoi son deterministas. Es decir, dado un particular estado y acción, el resultado es predecible y no involucra aleatoriedad.

- **Estático**: El entorno de Torre de Hanoi es estático, es decir, no cambia al menos que se lleve a cabo una acción por parte del agente. Los postes y los discos permanecen en las mismas posiciones al menos que el agente los mueva.

- **Secuencialidad**: El problema Torre de Hanoi involucra una secuencia de acciones, es decir, cada acción se construye en base a las anteriores, conllevando a la solución final.

- **Conocimiento**: El problema de Torre de Hanoi se encuentra bien definido y se conoce por completo, es decir, el agente tiene la información completa acerca de las reglas del juego y del estado actual del entorno.

2.3. *Estado, Espacio de estados, árbol de búsqueda, nodo de búsqueda, objetivo, acción y frontera*

- **Estado**: Un estado en el problema de la Torre de Hanoi describe la disposición actual de los discos en los tres postes. Cada estado se caracteriza por la ubicación de los discos en cada poste y su tamaño relativo. Por ejemplo, sea el Problema de Torre de Hanoi con 4 discos y 3 postes, donde discos con mayor índice corresponde a los de mayor tamaño, un posible estado sería: $[[4, 3, 2, 1], [], []]$, donde cada lista interna

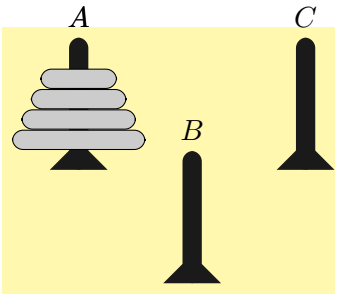

Figura 1: Ejemplo de Estado de Torre de Hanoi $[[4, 3, 2, 1], [], []]$

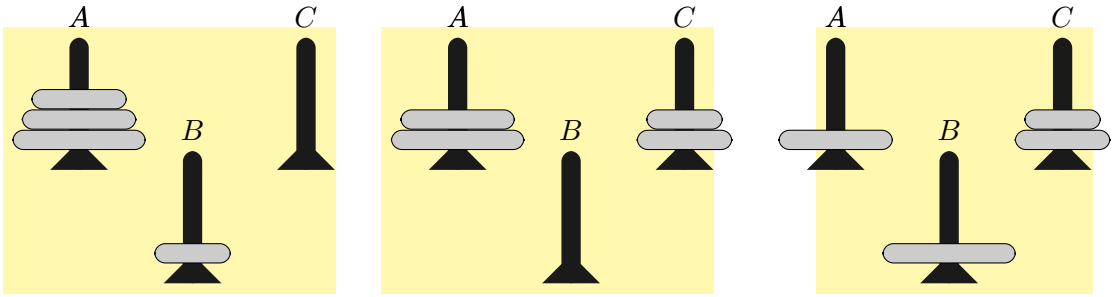

Figura 2: Ejemplos de Estados de Torre de Hanoi $[[4, 3, 2], [1], []]$, $[[4, 3], [], [2, 1]]$, $[[3], [4], [2, 1]]$

representa un poste y, el orden de los números corresponde a el orden de los discos desde la parte inferior a la superior en el poste (ver Figura 1). Si la lista interna se encuentra vacía eso implica que el poste no tiene discos.

- **Espacio de estados**: El espacio de estados es el conjunto de todos los posibles estados que pueden surgir durante la resolución del problema. En el caso de la Torre de Hanoi, el espacio de estados consiste en todas las posibles configuraciones de discos en los tres postes, obedeciendo las restricciones del problema. En base al ejemplo planteado en el ítem anterior, los siguientes son algunos de los posibles estados del espacio de estados: $[[4, 3, 2], [1], []]$, $[[4, 3], [], [2, 1]]$, $[[3], [4], [2, 1]]$ (ver Figura 2).
- **Árbol de búsqueda**: El árbol de búsqueda es una estructura jerárquica que representa todas las posibles secuencias de acciones que el agente puede tomar para resolver el problema. Cada nodo en el árbol de búsqueda representa un estado, y las ramas que salen de cada nodo representan las acciones posibles desde ese estado. A modo de ejemplo, en la Figura 3 se muestra el nodo raíz y sus dos nodos hijos. Luego en las Figuras 4 y 5 se muestran los nodos hijos de cada uno de los nodos hijos del nodo raíz, respectivamente.
- **Nodo de búsqueda**: Un nodo de búsqueda es un elemento en el árbol de búsqueda que contiene información sobre un estado específico del problema. Cada nodo puede tener uno o más nodos hijos que representan los posibles estados que pueden alcanzarse desde ese estado mediante una acción.
- **Objetivo**: El objetivo del problema de la Torre de Hanoi es mover todos los discos desde el poste de inicio hasta el poste objetivo, siguiendo las reglas del juego. El

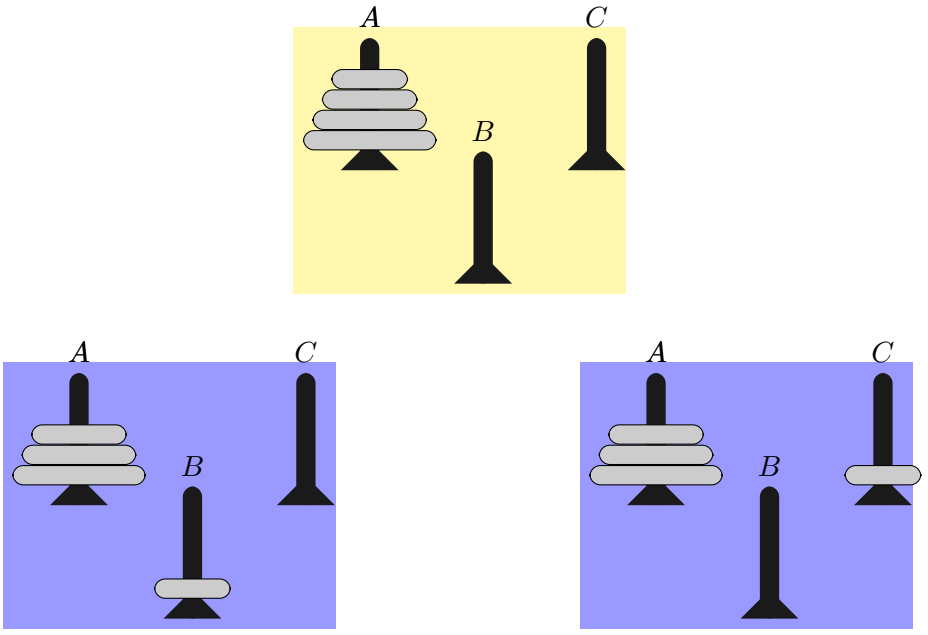

Figura 3: Ejemplo de Arbol de Estados. Estado Padre $[[4,3,2,1],[],[]]$. Estados Hijos $[[4,3,2],[1],[]]$, $[[4,3,2],[],[1]]$

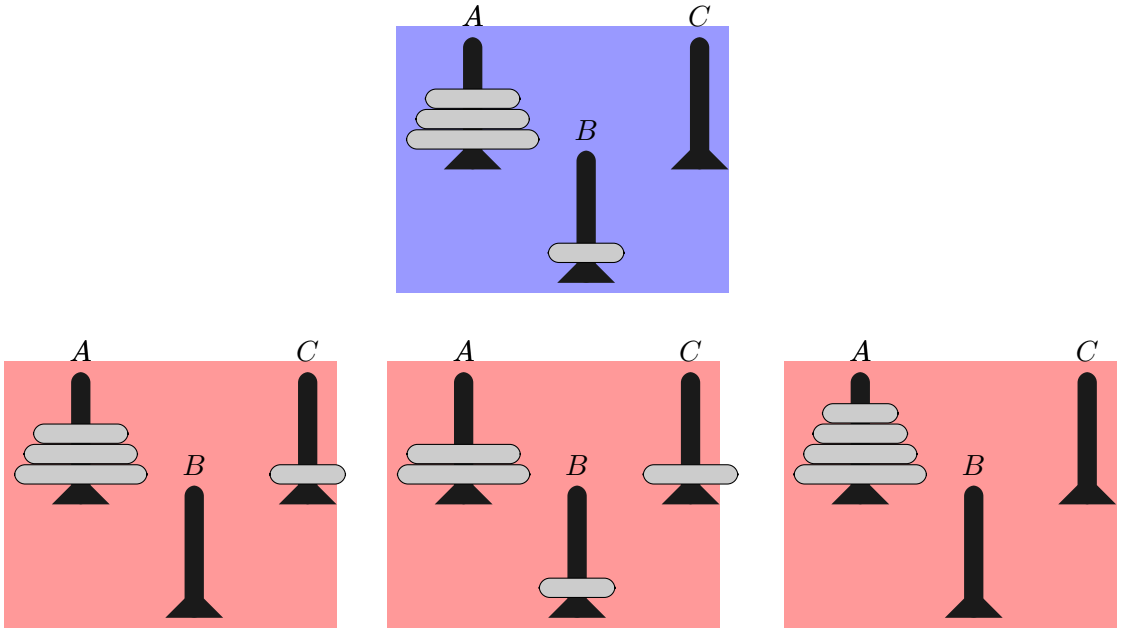

Figura 4: Ejemplo de Arbol de Estados. Estado Padre $[[4,3,2],[1],[]]$. Estados Hijos $[[4,3,2],[],[1]]$, $[[4,3],[1],[2]]$, $[[4,3,2,1],[],[]]$

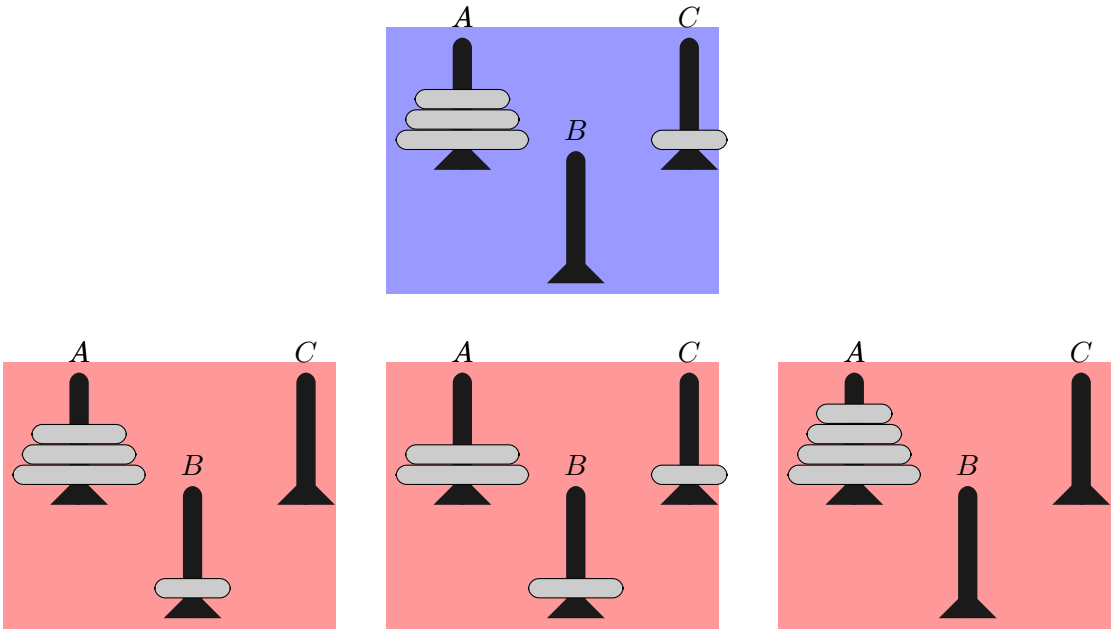

Figura 5: Ejemplo de Arbol de Estados. Estado Padre $[[4,3,2],[],[1]]$. Estados Hijos $[[4,3,2],[1],[]]$, $[[4,3],[2],[1]]$, $[[4,3,2,1],[],[]]$

objetivo se alcanza cuando todos los discos se han movido al poste objetivo en la misma disposición ordenada.

- **Acción**: Una acción en el problema de la Torre de Hanoi implica mover un disco de un poste a otro, siguiendo las reglas del juego. Solo se puede mover un disco a la vez y un disco no puede colocarse sobre uno más pequeño.
- **Frontera**: La frontera es el conjunto de nodos de búsqueda que aún no se han explorado pero que están disponibles para ser explorados. En un algoritmo de búsqueda, la frontera representa los posibles estados que el agente puede alcanzar desde el estado actual, pero que aún no se han explorado completamente. En la práctica, esto se implementa como una estructura de datos, como una cola o una pila, que contiene los nodos de búsqueda que deben expandirse en la siguiente iteración del algoritmo de búsqueda.

2.4. *Complejidad en tiempo y memoria*

2.4.1. Búsqueda en Amplitud (BFS)

- Complejidad en tiempo: En el peor caso, la complejidad en tiempo de BFS es $\mathcal{O}(b^d)$, donde b es el factor de ramificación promedio y d es la profundidad máxima del árbol de búsqueda. En la Torre de Hanoi, la profundidad máxima es $2^n - 1$, donde n es el número de discos.
- Complejidad en memoria: BFS requiere $\mathcal{O}(b^d)$ memoria en el peor caso para almacenar todos los nodos en un nivel dado del árbol de búsqueda.

2.4.2. Búsqueda en Amplitud (DFS)

- Complejidad en tiempo: En el peor caso, la complejidad en tiempo de DFS es $\mathcal{O}(b^d)$, donde b es el factor de ramificación promedio y d es la profundidad máxima del árbol de búsqueda. En el caso de la Torre de Hanoi, la profundidad máxima puede ser $2^n - 1$, donde n es el número de discos.
- Complejidad en memoria: DFS puede requerir $\mathcal{O}(b \cdot d)$ memoria en el peor caso para almacenar la ruta desde la raíz hasta el nodo actual.

2.5. Resultados

Los resultados presentados en esta sección se basan en resolver el problema de Torre de Hanoi con 3 postes y 4 discos, donde el estado inicial es $[[4, 3, 2, 1], [], []]$ y el estado final es $[[], [], [4, 3, 2, 1]]$. Se desea en primer lugar conocer el tiempo y memoria que ocupa el algoritmo, para ello se ejecutó el mismo unas 10 veces y se calculó su promedio y desvío estándar. Se resolvió con los algoritmos BFS y DFS y se obtuvieron los siguientes resultados:

- **BFS**: Tiempo= $126{,}8 \pm 6{,}4$ ms. Memoria= $2{,}2 \pm 0{,}2$ MB.
- **DFS**: Tiempo= $118{,}4 \pm 1{,}7$ ms. Memoria= $2{,}3 \pm 0{,}7$ MB.

Luego, se resolvió el problema de Torre de Hanoi pero para distintos casos y se calculó la cantidad de pasos hasta encontrar la solución. Los casos varían según el número de discos, en donde se comienza desde dos discos hasta finalizar con seis discos. Se sabe que la solución óptima implica una cantidad de pasos, los cuales vienen dados por la fórmula $2^k - 1$ con k el número de discos. En el Cuadro 1 se muestran los resultados obtenidos de los pasos necesarios para resolver el problema, según los dos tipos algoritmos implementados.

Discos	**Pasos teóricos**	**BFS**	**DFS**
2	3	9	5
3	7	27	23
4	15	81	31
5	31	243	193
6	63	729	225

Cuadro 1: Pasos obtenidos según los algoritmos BFS y DFS, para el problema de Hanoi con cuatro discos y tres postes.