Deusto

Unidad 2:

Resolución de problemas mediante búsquedas y heurísticas

Deusto

2.3 Algoritmos informados

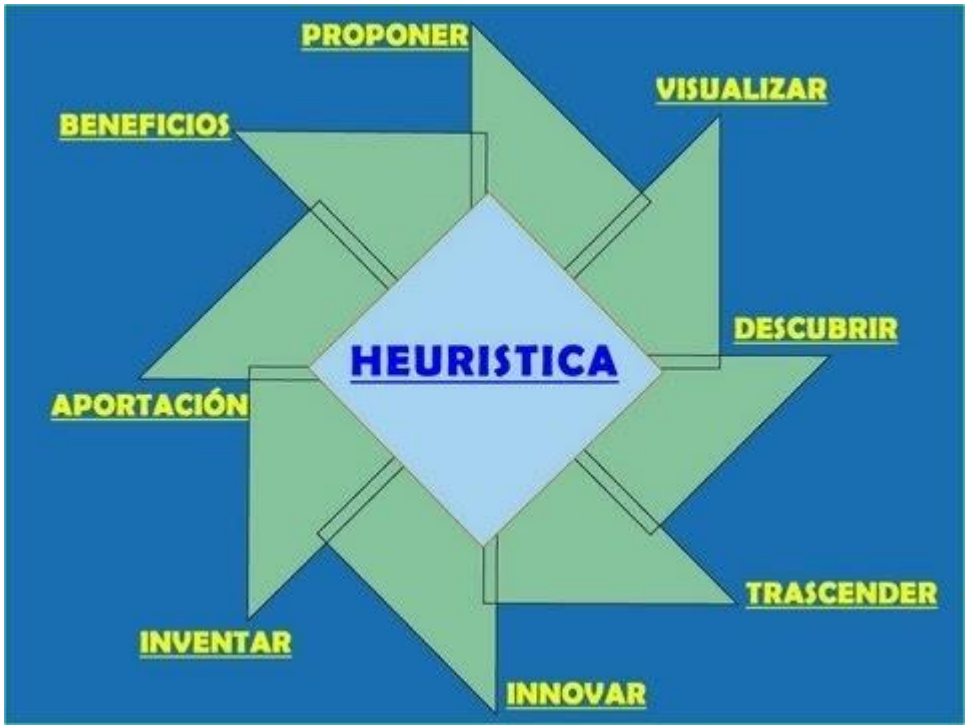
Introducción (1)

- Los algoritmos no informados pueden ser mejorados **proporcionando más información** sobre el problema.
- Búsqueda informada significa que el algoritmo tiene un **contexto específico** sobre cómo solucionar el problema.
- La heurística nos ayuda a **representar este contexto**.
  - Serie de reglas usadas para evaluar un estado.
  - Nos dice si un estado satisface o simplemente mide su rendimiento.
  - Solo se emplea en el caso de que no haya un método claro para encontrar una solución óptima.
  - No es una verdad absoluta sino una guía.
  - Emplea BFS y DFS de forma conjunta con algo de inteligencia.

Introducción (2)

- No existe un camino fijo para recorrer, al contrario de lo que sucede en BFS y DFS.
- Los nodos son visitados de acuerdo a su coste heurístico.
- Los algoritmos no saben los costes por adelantado, costes son calculados a medida que se recorre el árbol, cada nodo es visitado y añadido a la cola.
- Luego se ignoran los árboles cuyo coste es mayor que los que se han visitados, ahorrando coste de computación.

Heurística (1)



https://sites.google.com/site/razonamientomatematicobine/unidad-1

Heurística (2)

- Capacidad de realizar innovaciones para conseguir alcanzar los objetivos.
- Las soluciones se descubren por la evaluación del progreso logrado en la búsqueda del resultado final.
- “Un proceso que puede resolver un problema dado, pero que no ofrece ninguna garantía de que lo hará, se llama una heurística para ese problema“ (Shaw & Simon, 1963).

Observación reflexiva #1

- Imagina que tienes que programar algoritmos para planificar rutas en un GPS. Proporciona una mala heurística y una buena heurística.

Heurística (3)

- La heurística puede **reducir drásticamente** el coste de computación de problemas que son resueltos con algoritmos no informados.
- Ej.
  - Algoritmo que comprueba si un audio incluye contenido por copyright.
    - Solución primitiva: Comprobar si el audio completo coincide con algún elemento de la librería.
    - Solución heurística: Definir una heurística que minimice la diferencia de distribución de frecuencias entre dos audios.

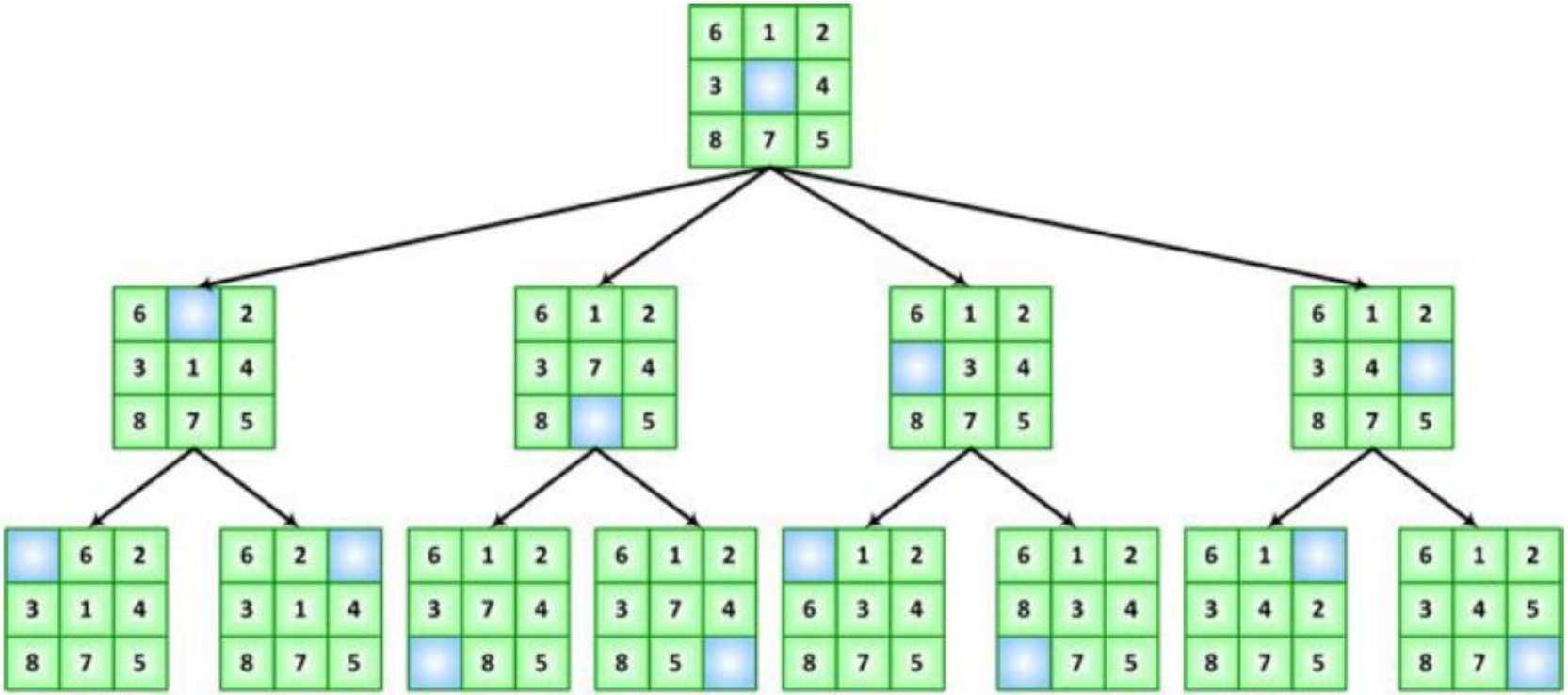
Heurística (3)

- Solución primitiva: Comprobar si el audio completo coincide con algún elemento de la librería.
- Solución heurística: Definir una heurística que minimice la diferencia de distribución de frecuencias entre dos audios.

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Búsqueda heurística (1)

Búsqueda heurística (2)



Búsqueda heurística (3)

- En la búsqueda no informada no se evalúan cuáles son los nodos en la frontera más prominentes para alcanzar el objetivo.
  - Ej. En *uniform-cost* escogemos el más barato.

- Sin embargo, a veces tenemos conocimientos sobre el estado.
  - Ej. Piezas mal colocadas.

Best First Approach (1)

- *Best First Approach*: El primero el mejor.
- Es una evolución del algoritmo básico de búsqueda o *basic search*.
- El siguiente nodo visitado será aquel que tenga una mejor función de evaluación f(n).
  - n = nodo
  - f(n) = coste de alcanzar una meta desde el nodo n.

Best First Approach (2)

- Para definir f(n) la mayoría de los algoritmos *Best First Approach* emplean una función heurística.
  - f(n) = h(n)
  - f(n) = h(n) + (otros)

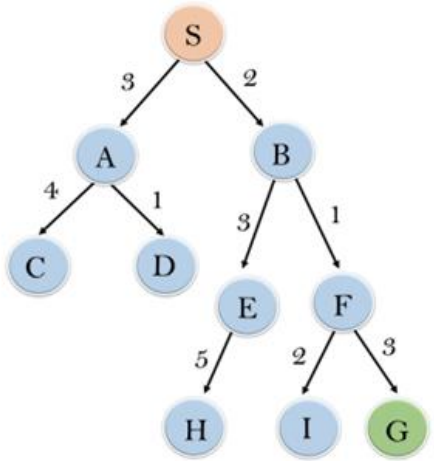
- h(n) = Coste **estimado** de la ruta más barata desde el nodo n al objetivo.
  - Estimación de cuán lejos está el nodo n del objetivo.
  - Por ahora consideramos que h(n) = no negativo y que h(nodo_objetivo) = 0

Greedy Best-First Search (1)

- *Greedy Best-First Search*: Búsqueda codiciosa del primero el mejor.
- Es una evolución del algoritmo básico de búsqueda o *basic search* donde los nodos de la frontera son ordenados de forma ascendente al valor f(n)
- f(n) = h(n)
  - En cada paso se expande aquel nodo que **parece** más cerca del objetivo de acuerdo a h(n).
    - Por esto es codicioso
  - Más cerca del objetivo <-> Menor valor para h(n)

Greedy Best-First Search (2)

- Para implementarlo se usa una cola en la que se ordena de acuerdo al valor de h(n) cada vez que se añade un elemento.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

https://www.javatpoint.com/ai-informed-search-algorithms

Greedy Best-First Search (3)

1. Make a node with the initial problem state
2. Insert node into the frontier data structure
3. **WHILE** final state not found **AND** frontier is not empty **DO**
   - **3.1** Remove first node from the frontier
   - **3.2 IF** node contains final state **THEN** final state found
   - **3.3 IF** node doesn't contain final state **THEN**
     - **3.3.1 EXPAND** node's state
     - **3.3.2** Insert successor nodes into frontier
     - **3.3.3 Sort frontier in ascending order of f(n)**
4. **IF** final state found **THEN**
   - **4.1 RETURN** sequence of actions found
5. **ELSE** "solution not found"

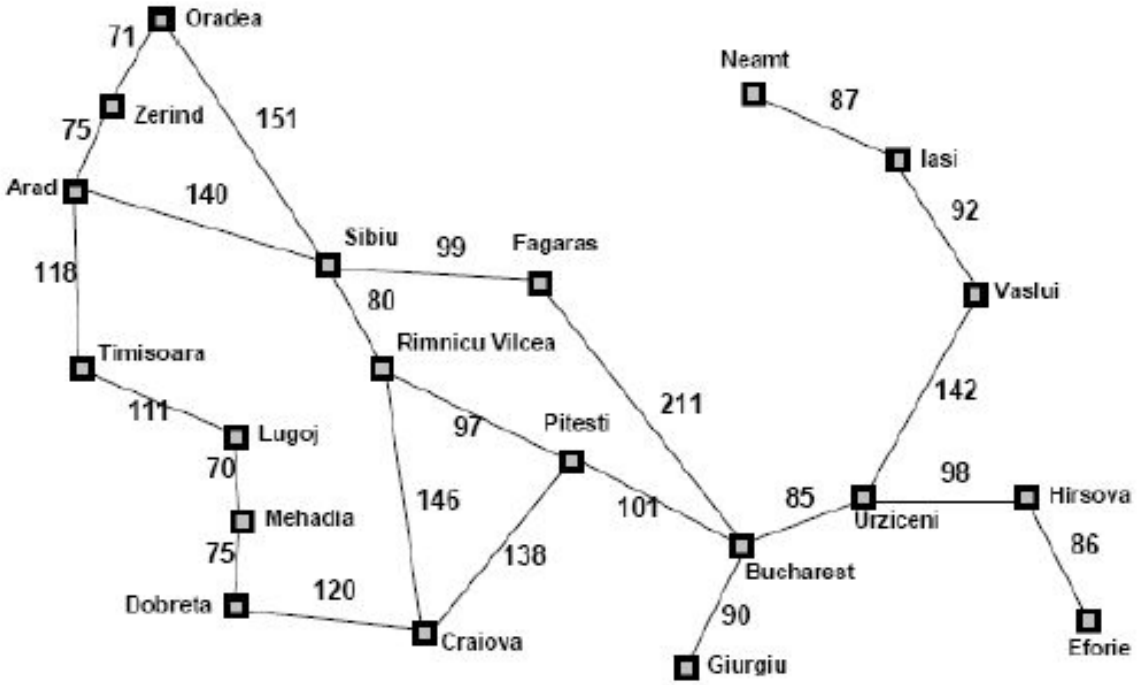
Lopez Gazpio, I. (2020). *Apuntes de Sistemas Inteligentes*. Universidad de Deusto.

Greedy Best-First Search (4) - Propiedades

- **Completitud**: No.
- **Óptimo**: No.
- **Complejidad temporal**: Peor caso $O(b^m)$
  - Una buena heurística lo mejora mucho.
- **Complejidad espacial**: $O(b^m)$ (mantiene cada nodo en memoria)

Observación reflexiva #2

Viaje de Arad a Bucharest usando el algoritmo *Greedy Best-First Search*, con h(n) = distancia en línea recta desde n a Bucharest.



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search (1)

- A* search = Búsqueda A estrella.
- El algoritmo *best-first search* más utilizado.
- Considera el coste de alcanzar el nodo así como la estimación del coste de llegar al objetivo.
- f(n) = g(n) + h(n)
  - g(n) Coste del nodo n en el camino actual.
  - Coste estimado del camino más barato desde n al objetivo.
  - f(n) Coste total estimado de la mejor solución pasando por n.
- Siempre se expande el nodo con el menor f(n) en la frontera.

A* search (2)

- En A*, como ocurre en DFS, el orden de los nodos hijos influencia el camino recorrido, pero no tan drásticamente ya que, si dos nodos tienen el mismo coste, el primer nodo es visitado antes del segundo.

https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/

A* search (3)

1. Make a node with the initial problem state
2. Insert node into the frontier data structure
3. **WHILE** final state not found **AND** frontier is not empty **DO**
   - **3.1** Remove first node from the frontier
   - **3.2 IF** node contains final state **THEN** final state found
   - **3.3 IF** node doesn't contain final state **THEN**
     - **3.3.1 EXPAND** node's state
     - **3.3.2** Insert successor nodes into frontier
     - **3.3.3 Sort frontier in ascending order of f(n)**
4. **IF** final state found **THEN**
   - **4.1 RETURN** sequence of actions found
5. **ELSE** "solution not found"

A* search (4) - Propiedades

- **Completitud**: Sí.
- **Óptimo**: Sí.
- **Complejidad temporal**: Peor caso $O(b^m)$
  - Una buena heurística lo mejora mucho.
- **Complejidad espacial**: $O(b^m)$ (mantiene cada nodo en memoria)

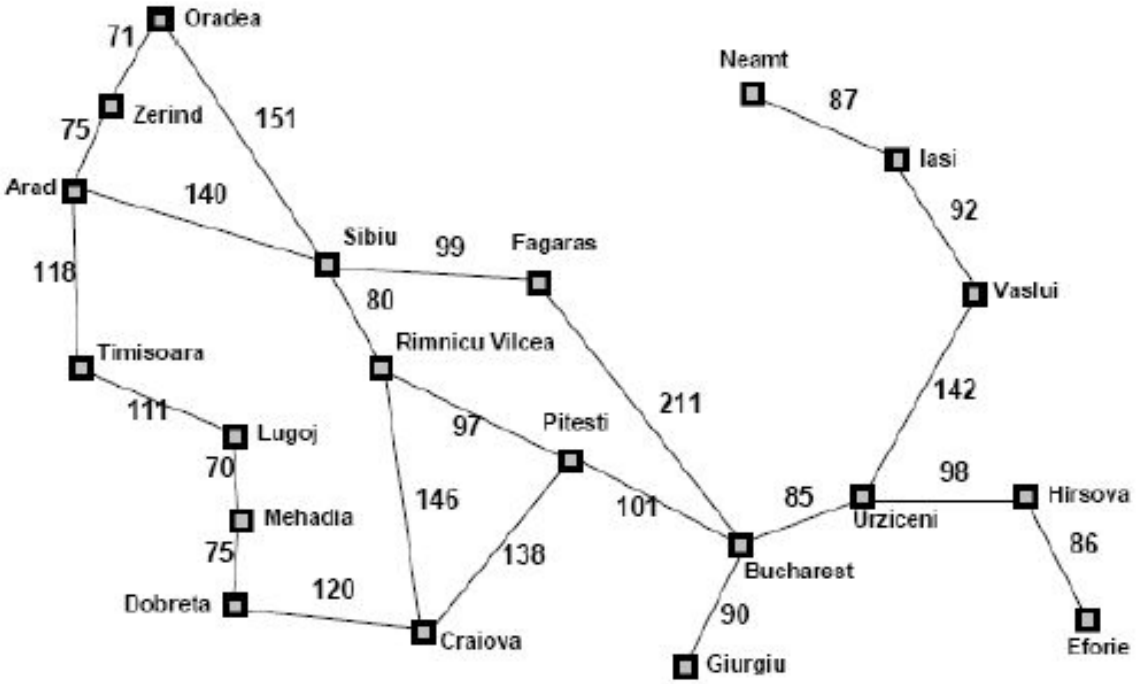
¡Siempre que h(n) sea admisible!

A* search (5) - Admisible

- Se asume que el coste de cualquier transición es mayor que cero.
- Sea h*(n) el coste del camino óptimo desde n al objetivo.
- Una heurística admisible satisface la condición h(n) <= h*(n)
  - Siempre subestimamos el verdadero coste <-> Nunca sobreestimamos el verdadero coste.
    - Esto es optimista.
  - h(g) = 0
  - h(n) = ∞, si no hay ningún camino desde n al objetivo.
- Si h(n) es admisible luego la búsqueda es óptima.

Observación reflexiva #2

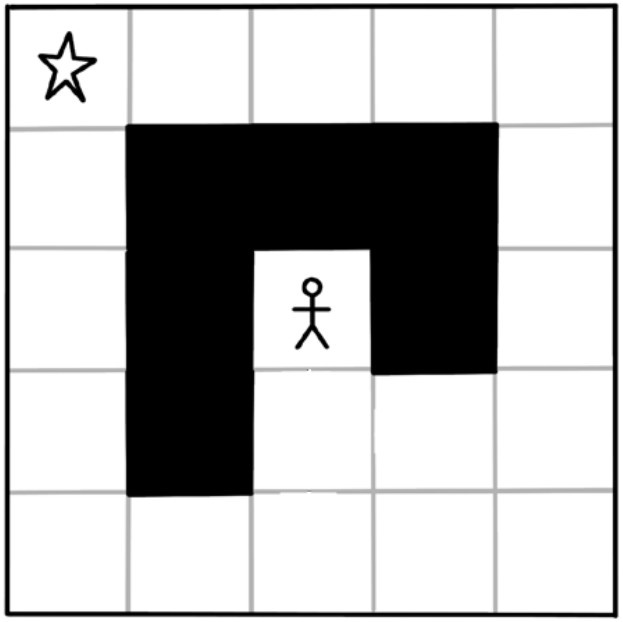
Viaje de Arad a Bucharest usando el algoritmo A*, con h(n) = distancia en línea recta desde n a Bucharest y g(n) la distancia actual para alcanzar el nodo.



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Entregable 2.4

¿Cómo se recorrería el siguiente árbol de acuerdo al algoritmo A*?



[Fecha límite: **08/11/2020 12:00**]

Comparación: no informados vs informados

No informados	Informados
No emplea conocimiento	Emplea conocimiento
Menos rápido	Más rápido
Menos eficiente	Más eficiente
Coste de computación más alto*	Coste de computación más bajo*
Coste temporal más alto*	Coste temporal más bajo*
No proporciona direcciones respecto a la solución	Proporciona direcciones respecto a la solución

Usos: no informados vs informados

No informados	Informados
Encontrar rutas entre los nodos de una red	Movimiento autónomo de personajes en videojuegos
Rastreador web: Web crawler [info]	Parseado de párrafos para NLP
Encontrar conexiones en redes sociales	Encontrar rutas de telecomunicaciones
Coste de computación más alto*	Juegos de un jugador y puzzles
Coste temporal más alto*	Coste temporal más bajo*
No proporciona direcciones respecto a la solución	Proporciona direcciones respecto a la solución

Bibliografía

Esta presentación se basa principalmente en información recogida en las siguientes fuentes:

- Hurbans, R. (2020). *Grokking Artificial Intelligence Algorithms*. Manning Publications.
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A modern approach. 3ª Ed*. Prentice-Hall.
- Lopez Gazpio, I.. (2020). *Apuntes de Sistemas Inteligentes*. Universidad de Deusto.