Búsqueda

Dra. Cecilia Reyes Peña

Problemas de búsqueda

Estos problemas implican encontrar una secuencia de acciones o pasos que llevarán a un agente desde un estado inicial a un estado objetivo en un espacio de estados.

Agentes basados en objetivos

Los agentes basados en objetivos establecen un conjunto de acciones para alcanzar sus metas u objetivos. Para esto, establece un conjunto de estados del ambiente y trata de satisfacerlos.

Algoritmos de búsqueda:

Entrada: un problema

Salida: un conjunto de acciones

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

Elementos de los problemas de búsqueda

1. Espacio de Estados: representa todas las posibles configuraciones o situaciones en las que puede encontrarse el sistema. Cada estado es una configuración única del sistema.
2. Estado Inicial y Objetivo: El estado inicial es el punto de partida del agente, y el estado objetivo es el estado al que el agente desea llegar. El objetivo es encontrar una secuencia de acciones que conduzcan del estado inicial al estado objetivo.
3. Operadores y Acciones: Los operadores son acciones que pueden aplicarse para cambiar de un estado a otro. La búsqueda implica encontrar una secuencia de operadores que transforman el estado inicial en el estado objetivo.
4. Árbol de Búsqueda: Se puede representar la exploración del espacio de estados como un árbol de búsqueda, donde cada nodo es un estado y las aristas son las acciones que conducen de un estado a otro.

Elementos de los problemas de búsqueda

1. Complejidad Temporal y Espacial: La complejidad temporal se refiere al tiempo necesario para encontrar una solución, mientras que la complejidad espacial se refiere al espacio de almacenamiento necesario durante la búsqueda.
2. Búsqueda No Informada (ciega): En la búsqueda no informada, el agente no tiene información sobre la probabilidad de alcanzar el estado objetivo desde un estado dado. Métodos como la búsqueda en anchura y la búsqueda en profundidad son ejemplos de estrategias no informadas.
3. Búsqueda Informada (heurística): En la búsqueda informada, el agente utiliza información adicional para guiar la exploración hacia el estado objetivo de manera más eficiente. La heurística es una función que estima el costo desde un estado hasta el objetivo.

Elementos de los problemas de búsqueda

1. Heurísticas Admisibles e Inadmisibles: Una heurística es admisible si nunca sobreestima el costo para llegar al objetivo desde un estado dado. Las heurísticas inadmisibles pueden sobreestimar o subestimar el costo.
2. Optimalidad: Una solución es óptima si es la más corta o la de menor costo entre todas las soluciones posibles.
3. Problema del Espacio de Estado Infinito: Algunos problemas de búsqueda pueden tener un espacio de estados infinito, lo que plantea desafíos adicionales en la representación y exploración.

Ejemplo de un problema de búsqueda

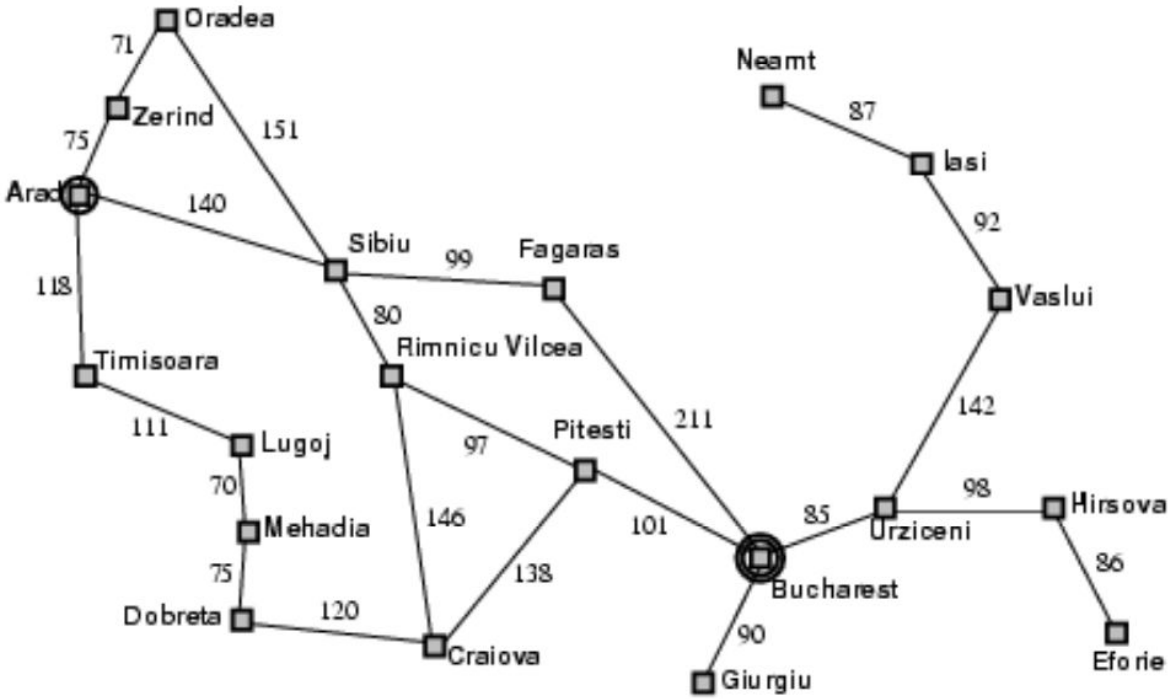
Viaje de Arad a Bucarest

Formulación del problema:

-Estados: cada ciudad

-Acciones: moverse de una ciudad a otra

Solución: encontrar una secuencia para ir de C1 a C2



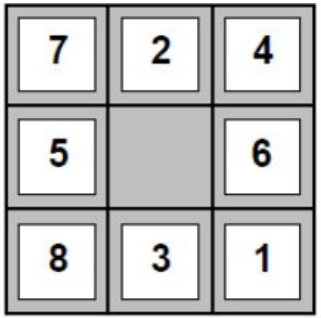
Ejemplo de un problema de búsqueda

Un problema consta de cuatro componentes

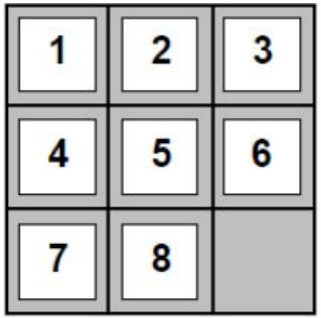
- Estado inicial: estado en el que comienza el agente; p.ej: “en Arad“
- Descripción de las posibles acciones disponibles por el agente: función sucesor *S(x)* = conjunto de pares acción–estado; p.ej. *S(Arad) = {<Arad → Zerind, Zerind>, … }*
- Test objetivo: el cual determina si un estado es un estado objetivo; p.ej. “en Bucarest?”
- Función de costo del camino: asigna un costo numérico a cada camino; usualmente descrita como la suma de los costos de las acciones individuales a lo largo del camino; p.ej. suma de distancias

Una solución de un problema es una camino desde el estado inicial a un estado objetivo

Ejemplo de problema de búsqueda



Estado inicial



Estado final

Estados: ubicaciones de las piezas

Estado inicial: el que muestra la figura de la izquierda

Función sucesor: genera estados legales al intentar mover el espacio vacío hacia a la izquierda, a la derecha, arriba o abajo

Test objetivo: ¿se llegó al estado mostrado en la figura de la derecha?

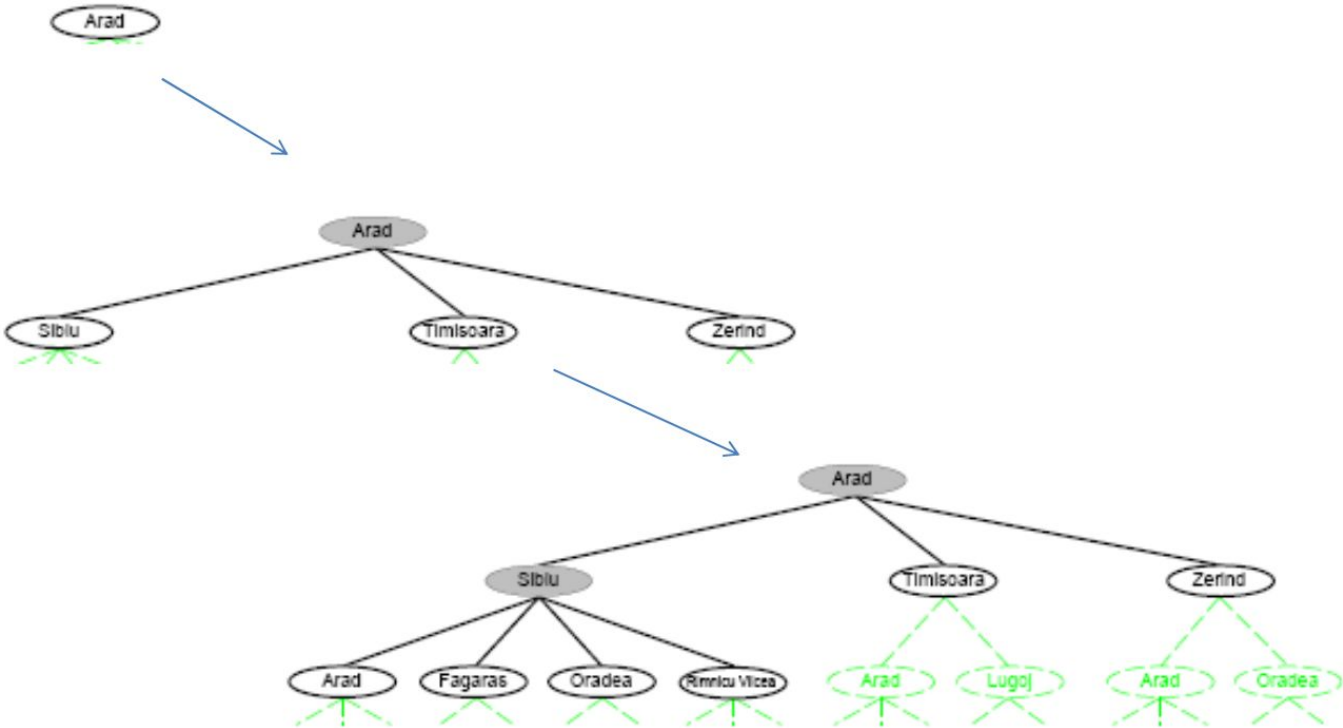
Costo del camino: 1 por movimiento

Búsqueda de soluciones

La resolución de problemas se hace mediante búsqueda a través del espacio de estados.

Las técnicas básicas usan un árbol de búsqueda explícito generado a partir del estado inicial y la función sucesor para generar nuevos estados.

Ejemplo árbol de búsqueda viajes

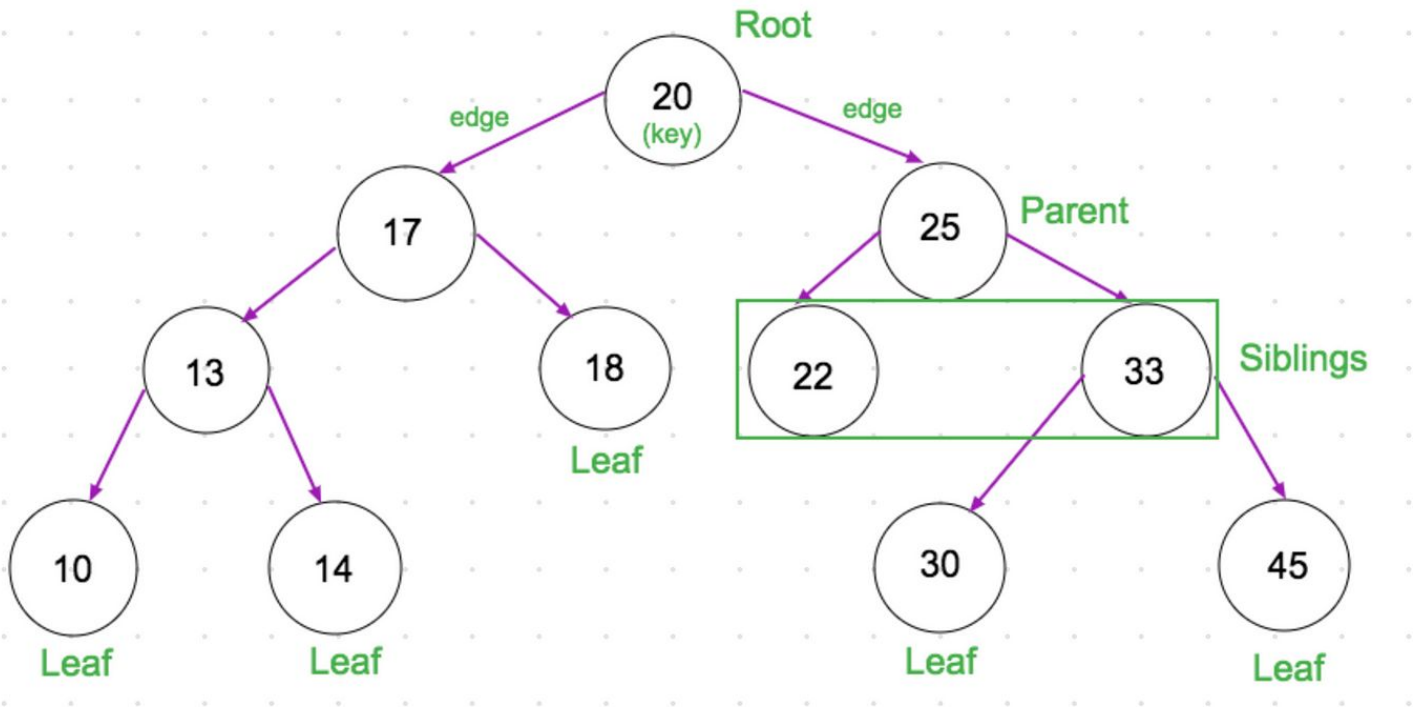


Estructura general del árbol de búsqueda

Los árboles están formados por:

1. Estado: el estado que corresponde con el nodo
2. Nodo Padre: el nodo del árbol de donde se generó un nodo
3. Acción: es la acción aplicada al padre para generar un nodo
4. Costo del camino: el costo desde el nodo conteniendo al estado inicial hasta el nodo actual
5. Profundidad: el número de pasos a lo largo del camino desde el nodo inicial

Estructura de un árbol de búsqueda



Estrategia de búsquedas

Una estrategia se define eligiendo el orden de expansión de los nodos.

Las estrategias se evalúan de acuerdo a:

1. Completitud: Se encuentra la solución si esta existe?
2. Complejidad temporal: número de nodos generados
3. Complejidad espacial: máximo número de nodos en memoria
4. Optimalidad: siempre se encuentra la solución de más bajo costo?

Búsquedas no informadas (a ciegas)

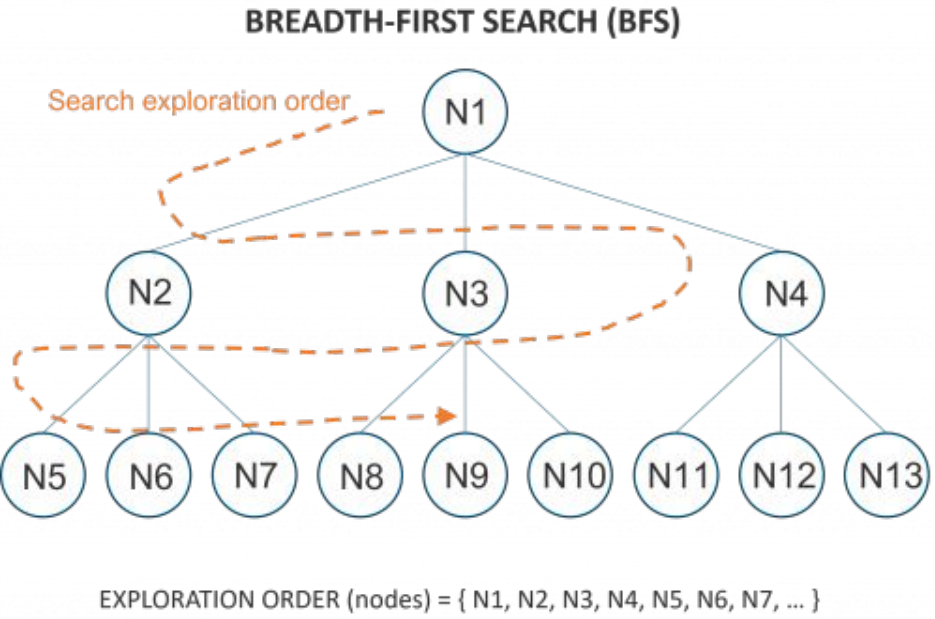
Las búsquedas no informadas se refieren a métodos de búsqueda que no utilizan información adicional o heurísticas para guiar el proceso de búsqueda, simplemente exploran el espacio de estados de manera ciega, sin ninguna orientación.

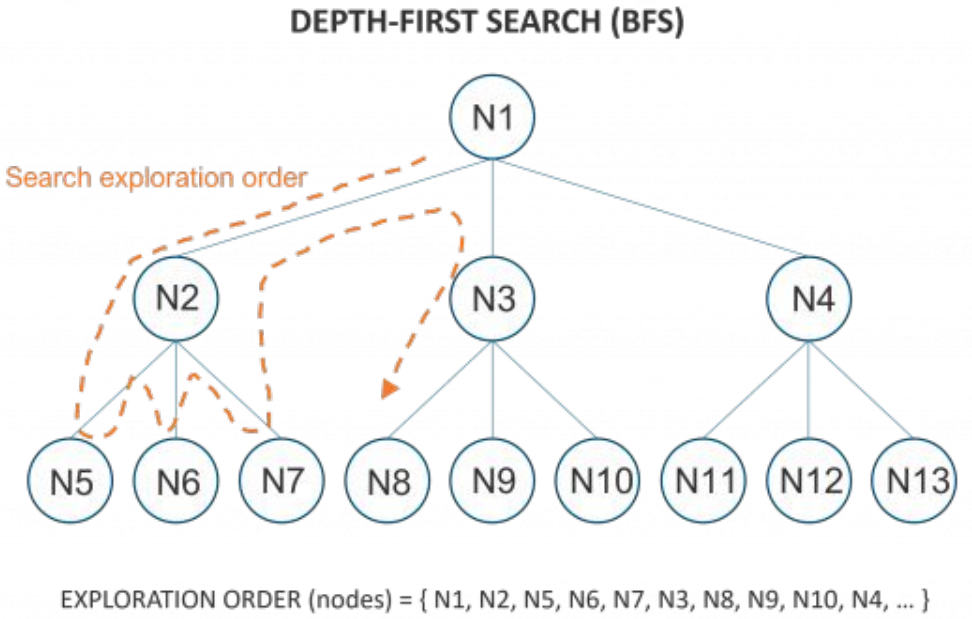
No detecta si se está aproximando o alejando de la solución.

No es capaz de encontrar una solución aceptable en caso de que no exista o sea demasiada costo de encontrar.

Búsqueda no informada (a ciegas)

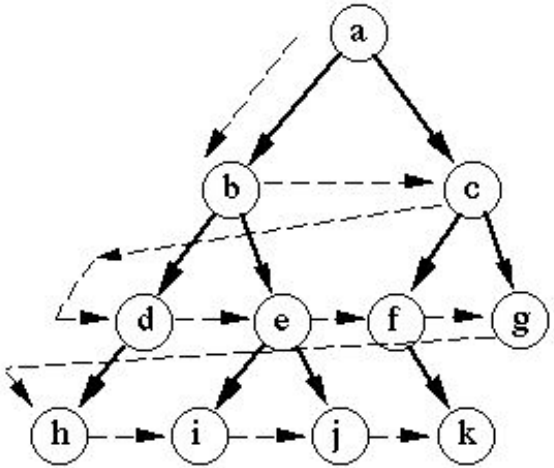
Los tipos de búsqueda no informada son: primero a lo ancho y primero a lo profundo.





Búsqueda en primero a lo ancho (BFS)

Este algoritmo explora todos los nodos vecinos del nodo actual antes de pasar a los nodos vecinos de esos nodos. BFS garantiza encontrar la solución más cercana, pero puede ser costoso en términos de memoria y tiempo, especialmente en espacios de estados grandes o infinitos. La frontera es una cola FIFO, es decir, nuevos sucesores van al final.



Breadth-first search

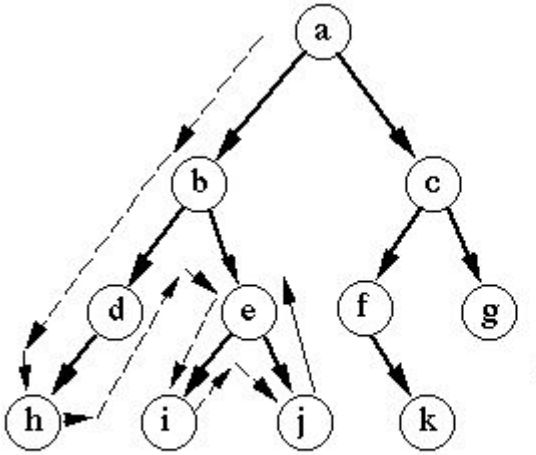
Orden de visita???

Algoritmo de BFS

1. **Crear** una lista con un solo elemento consistente en una trayectoria o camino de longitud cero: *el nodo raíz*
2. **Hasta que** el primer camino de la lista llegue al nodo objetivo o se llegue a la lista vacía **hacer**
   a. Extraer el primer camino de la lista
   b. Expandir el nodo final de este camino a todos los vecinos del nodo terminal.
   c. Eliminar los ciclos de los caminos expandidos.
   d. Insertar estos nuevos caminos al ***Final*** de la lista.
3. **FIN Hasta**
4. **Si** se halla el nodo meta notifique el éxito, si no el fracaso

Búsqueda primero en lo profundo (DFS)

DFS explora un camino hasta que alcanza un estado final o un estado sin opciones, luego retrocede y explora otro camino. DFS es menos costoso en términos de memoria que BFS, pero puede quedarse atascado en ciclos infinitos y no garantiza encontrar la solución más cercana. La frontera es una cola LIFO, es decir los nuevos sucesores van al inicio.



Depth-first search

Orden de visita???

Algoritmo DFS

1. **Crear** una lista con un solo elemento consistente en una trayectoria de longitud cero: *el nodo raíz*
2. **Hasta que** el primer camino de la lista llegue al nodo objetivo o se llegue a la lista vacía **HACER**
   a. Extraer el primer camino de la lista
   b. Expandir el nodo final de este camino.
   c. Eliminar los ciclos de los caminos expandidos.
   d. Insertar estos nuevos caminos al ***INICIO*** de la lista

   FIN Hasta
1. **Si** la lista está vacía, **entonces** NO hay solución; **Si no** el primer camino de la lista es la solución

Otras búsquedas no informadas

1. Búsqueda de costo uniforme: En este algoritmo, la búsqueda se realiza expandiendo los nodos con menor costo de ruta acumulado hasta el momento. A diferencia de BFS y DFS, que no tienen en cuenta el costo de las rutas, la búsqueda de costo uniforme prioriza los nodos con menor costo de ruta.
2. Búsqueda en profundidad limitada (DLS): Es similar a DFS, pero se detiene después de un número fijo de niveles o profundidad en el árbol de búsqueda. Esto evita que el algoritmo se quede atascado en profundidades infinitas.
3. Búsqueda en profundidad iterativa (IDDFS): Es una combinación de DFS y DLS, donde se ejecutan múltiples iteraciones de DFS con límites de profundidad crecientes hasta que se encuentra una solución.

Búsquedas informadas (heurísticas)

Las búsquedas informadas o búsquedas heurísticas, utilizan información adicional para guiar el proceso de búsqueda hacia la solución de manera más eficiente. Esta información adicional se proporciona mediante una función heurística que estima cuánto falta para llegar al estado objetivo desde un estado dado.

Las heurísticas tienen algún conocimiento sobre la ubicación del objetivo y se utiliza para dirigir la búsqueda de manera más eficiente.

Reduce el espacio de búsqueda y es capaz de determinar su proximidad a una solución y la calidad de la misma utilizando conocimiento a priori.

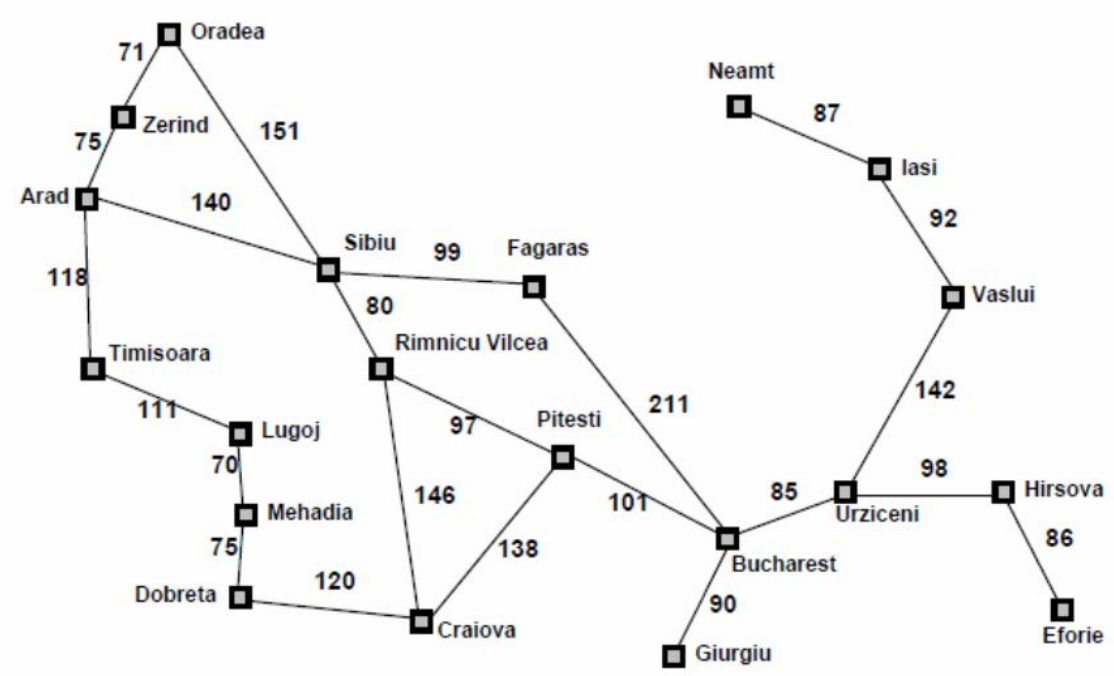
Función de evaluación

La función heurística es una función que proporciona una estimación del costo o la distancia desde un estado dado hasta el estado objetivo en un problema de búsqueda. Esta función es esencial para guiar la búsqueda hacia la solución de manera más eficiente.

La función heurística se utiliza para evaluar qué tan prometedores son los estados en un problema de búsqueda. Proporciona una estimación del costo restante para alcanzar el estado objetivo desde un estado dado. Cuanto más cercana sea esta estimación al costo real, más efectiva será la función heurística. Una función heurística se considera admisible si nunca sobreestima el costo restante para alcanzar el objetivo. En otras palabras, la estimación proporcionada por la función heurística nunca es mayor que el costo real.

Ejemplo función heurística

Distancia en línea recta desde Bucarest (hDLR)



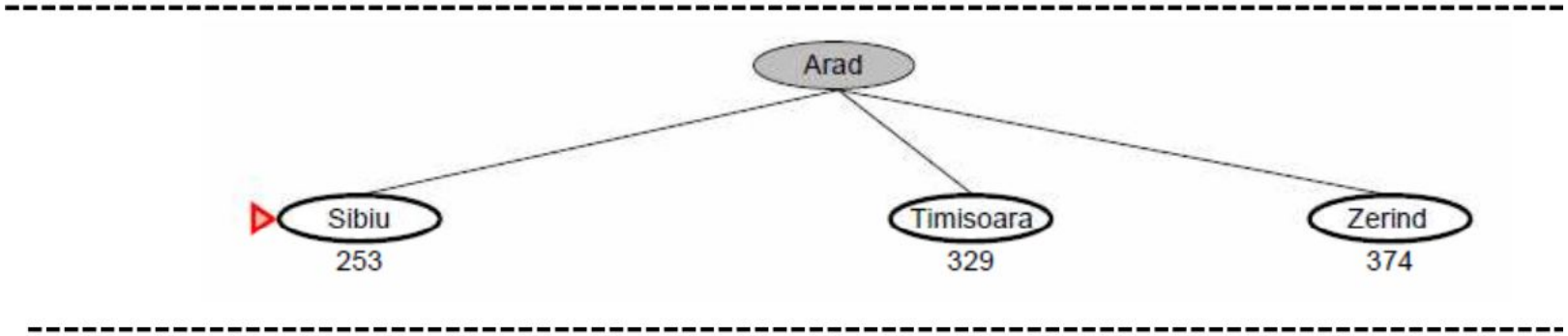
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Búsqueda primero el mejor (BFS)

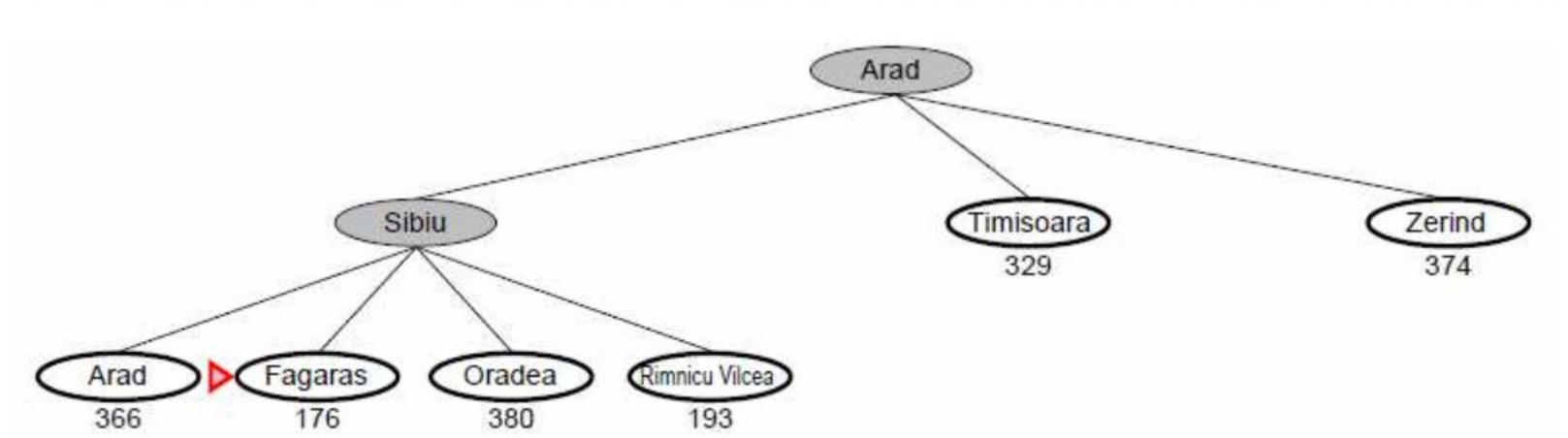
Este es un tipo de búsqueda informada que expande el nodo que parece estar más cerca del objetivo, según la función heurística, pero no toma en cuenta el costo total acumulado para llegar a ese nodo. El algoritmo busca una solución optima global de la resolución del problema.

No garantiza la solución optima.

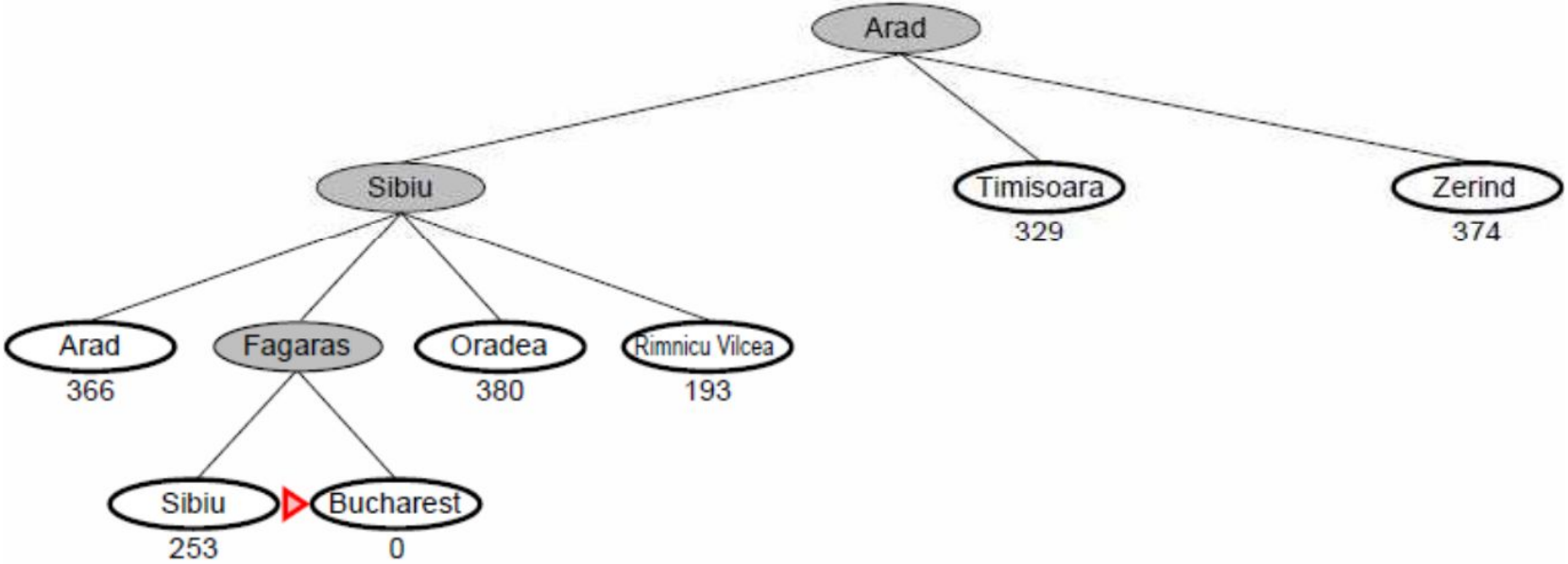
BFS-Voraz



BFS-Voraz


Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
176
Oradea
380
Rimnicu Vilcea
193

BFS-Voraz


Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
Oradea
380
Rimnicu Vilcea
193
Sibiu
253
Bucharest
0

BFS-Voraz

Solución de Búsqueda voraz primero el mejor:

- Arad – Sibiu – Fagaras – Bucharest
- Costo total: (140+99+211) = 450

Sin embargo:

- Arad – Sibiu – Rimmicu – Pitesti – Bucharest
- Costo total: (140+80+97+101) = 418

Búsqueda A*

Utiliza tanto el costo acumulado para llegar a un estado como una estimación (heurística) del costo restante para alcanzar el objetivo. La función de evaluación utilizada en A* es:

$$f(n) = g(n) + h(n)$$

donde:

1. $g(n)$ es el costo acumulado para llegar al estado actual n
2. $h(n)$ es la estimación heurística del costo restante para alcanzar el objetivo desde n
3. n.A* garantiza encontrar la solución óptima si la heurística es admisible (no sobreestima el costo) y consistente (satisface la condición de monotonía).

Búsqueda A*

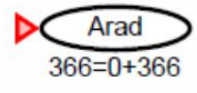
En cada paso se expande el nodo con el valor más bajo de f(n), ó sea, de g(n)+h(n)

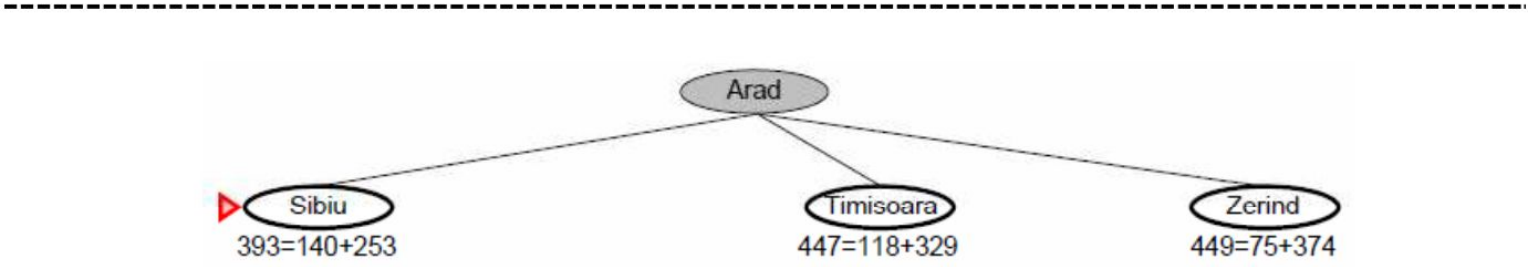
La búsqueda A* es óptima siempre y cuando la función heurística h(n) sea una heurística *admisible,* i.e. nunca sobreestime el costo de alcanzar el objetivo

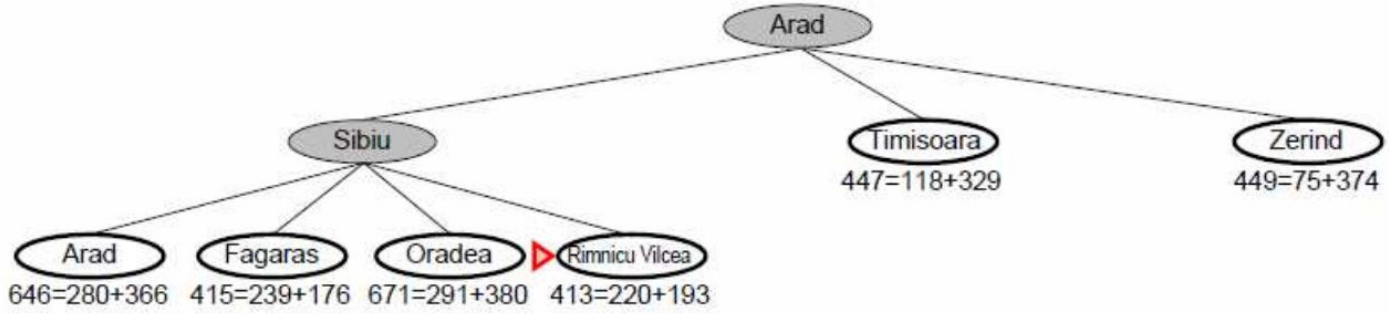
Son funciones optimistas

En el ejemplo hDLR es admisible ya que la distancia en línea recta entre dos puntos es el camino más corto.

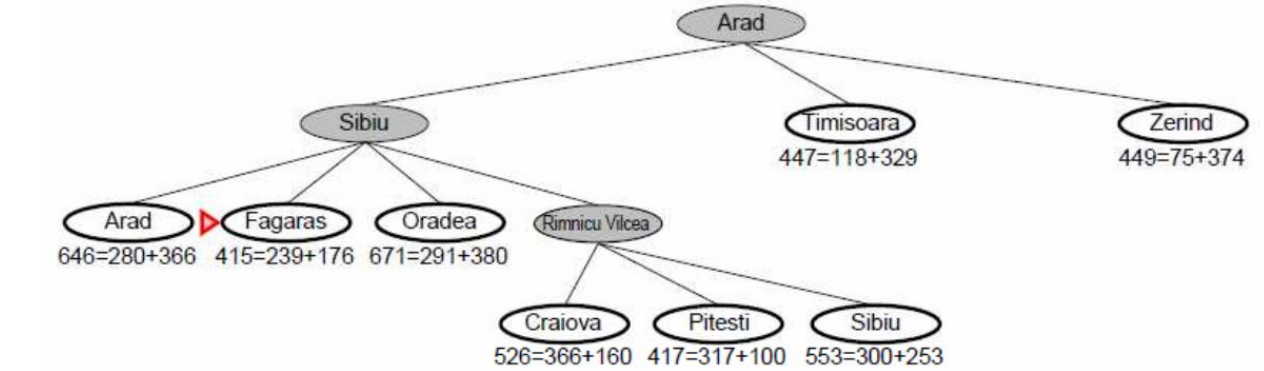
Ejemplo Viaje A*

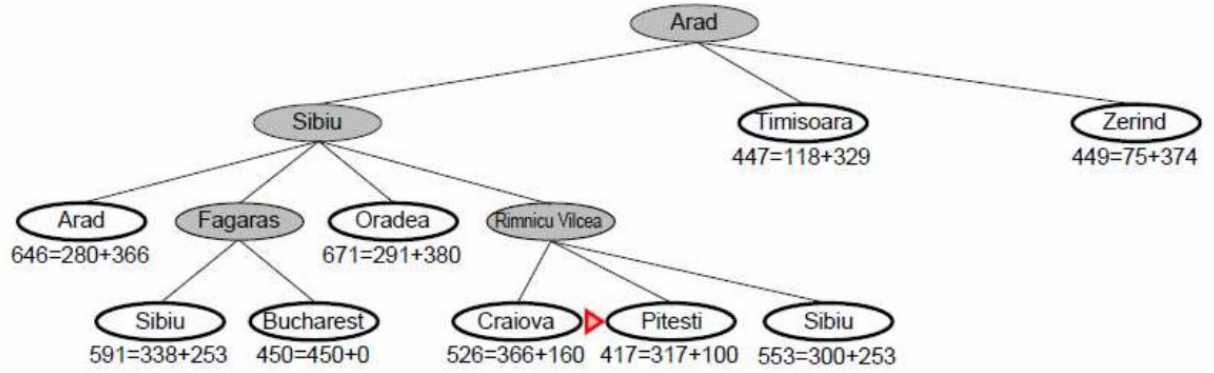






Ejemplo Viaje A*





Ejemplo Viaje A*