TÉCNICAS HEURÍSTICAS DE BÚSQUEDA

Abraham Sánchez López
FCC/BUAP
Grupo MOVIS

MOVIS

Facultad de Ciencias de la Computación

Introducción

- Problemas que no poseen una solución exacta.
- Problemas que dan lugar a una explosión combinatoria.
  - Aplicar un método de búsqueda exhaustiva en ambos tipos de problemas resulta ineficiente.
  - El número de nodos a expandir es muy grande, lo que acaba con los recursos de tiempo y memoria disponibles.
- La exploración del espacio de estados puede verse reducida por el uso de información peculiar asociada a cada problema concreto.
- En la mayoría de los casos esta información se obtiene de la experiencia y de las características específicas de cada problema.
- A este tipo de información se le llama **información heurística**.

Búsqueda informada

- Búsqueda ciega o no informada (anchura, profundidad,. . .): no cuenta con ningún conocimiento sobre como llegar al objetivo.
- Búsqueda informada: aplicar *conocimiento* al proceso de búsqueda para hacerlo mas eficiente.
- El conocimiento estará dado por una función que *estima* la “bondad” de los estados:
  - Utilizar una estimación del costo de la solución para guiar la búsqueda.
  - Dar preferencia a los mejores estados.
  - Ordenando la cola de ABIERTOS, comparando su bondad estimada.
  - Objetivo: podar el espacio de búsqueda, ganando *eficiencia* en la practica.
  - No siempre garantizan el óptimo, ni una solución.

Algunos detalles, I

- Por lo general tendremos un costo asociado a los operadores que nos permiten pasar de un estado a otro, por lo que ese costo tendrá un papel fundamental en el cálculo del costo del camino
- Los algoritmos que estudiaremos en esta parte del curso, utilizarán el cálculo del costo de este camino para saber que nodos vale la pena explorar antes.
- De esta manera, perderemos la sistematicidad de los algoritmos de búsqueda no informada, y el orden de visita de los estados de búsqueda no vendrá determinado por su posición en el grafo de búsqueda, sino por su costo.
- Dado que el grafo se va generado a medida que lo exploramos, podemos ver que tenemos dos elementos que intervienen en el costo del camino hasta la solución que buscamos.
- En primer lugar, tendremos el costo del camino recorrido, que podremos calcular simplemente sumando los costos de los operadores aplicados desde el estado inicial hasta el nodo actual.
- En segundo lugar, tendremos el costo más difícil, que es el del camino que nos queda por recorrer hasta el estado final.

Algunos detalles, II

- Dado que lo desconocemos, tendremos que utilizar el conocimiento del que disponemos del problema para obtener una aproximación.
- Evidentemente, la calidad de ese conocimiento que nos permite predecir el costo futuro, hará más o menos exitosa nuestra búsqueda.
- Si nuestro conocimiento fuera perfecto, podríamos dirigirnos rápidamente hacia el objetivo descartando todos los caminos de mayor costo, en este extremo podríamos encontrar nuestra solución en tiempo lineal.
- Por otro lado, si estuviéramos en la total ignorancia, tendríamos que explorar muchos caminos antes de hallar la solución óptima, todos en el peor caso.
- Esta última situación es en la que se encuentra la búsqueda no informada y desgraciadamente la primera situación es rara.
- Quedará pues como labor fundamental en cada problema, obtener una función que nos permita calcular el costo futuro desde un estado al estado solución.
- Cuanto más podamos ajustarlo al costo real, mejor funcionarán los algoritmos que veremos a continuación.

Búsqueda heurística

- Idea: no ignores el objetivo al seleccionar caminos.
- A menudo existe un conocimiento adicional que puede usarse para guiar la búsqueda: heurística.

- $h(n)$ es una estimación del costo del camino más corto desde el nodo n hasta un nodo objetivo.

- $h(n)$ necesita ser eficiente para calcular.

- h puede extenderse a caminos: $h(\langle n_0, \ldots, n_k \rangle) = h(n_k)$.

- $h(n)$ es subestimada si no hay un camino desde n hasta una meta con un costo menor que $h(n)$.

- Una heurística admisible es una función heurística no negativa que subestima el costo real de un camino hacia una meta.

Concepto de heurística

- Heurística:
  - Del griego heuriskein, descubrir: !Eureka¡
  - Según el diccionario: “Técnica de la indagación y del descubrimiento”
  - Otro significado: método para resolver problemas que no garantizan la solución, pero en general funciona bien.
  - En nuestro caso, una heurística será una función numérica sobre los estados.

- Función de evaluación heurística, heurística(estado):
  - Estima la “distancia” al objetivo
  - Siempre mayor o igual que 0
  - Valor en los estados finales: 0
  - Se admite el valor ∞

- Todo el conocimiento específico sobre el problema está codificado en la función heurística.

Problema del viajero

Ejemplos de heurística

- Problema del viajero:
  - heurística(estado) = |estado - *estado-final*|

- Problema del agente viajero por Puebla
  - Coordenadas:
  - Puebla : (409.5 93)
  - Atlixco : (309 127.5)
  - Cholula : (232.5 75)
  - San Martín: (63 57)
  - Tehuacan: (3 139.5)
  - Zacatlan : (90 153)
  - Izúcar : (198 207)
  - Teziutlan : (295.5 192)

- Función de evaluación heurística (distancia en línea recta):
  heurística(estado) = distancia(coordenadas(estado),coordenadas(Puebla))

8-puzzle

- Recordemos, que es un problema clásico en IA, se utiliza un cajón cuadrado en el que hay situados 8 bloques cuadrados. El cuadrado restante está vacío (hueco). Cada bloque tiene un número. Un bloque adyacente al hueco puede deslizarse hacia él. El juego consiste en transformar la posición inicial a la posición final mediante el deslizamiento de los bloques.
- Consideremos el estado inicial y final

2	**8**	**3**
1	**6**	**4**
7		**5**

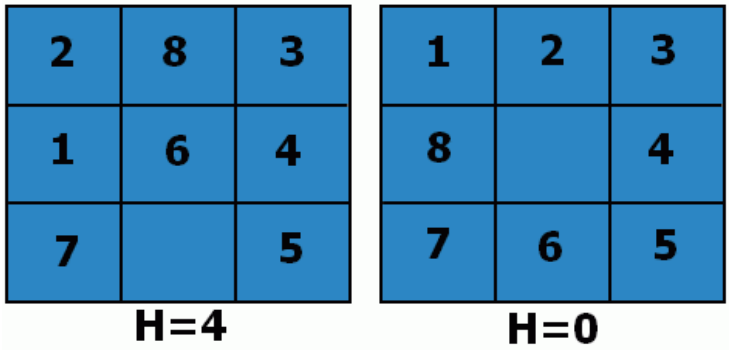
Estado inicial

1	2	3
8		4
7	6	5

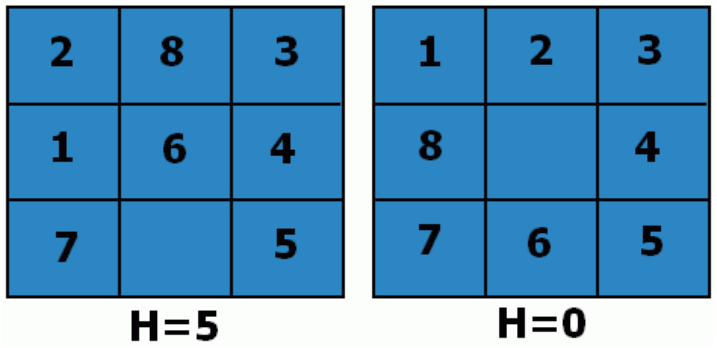
Estado final

Heurísticas para el 8-puzzle

- Heurística(estado)="número de piezas mal colocadas respecto de su posición en el estado final"

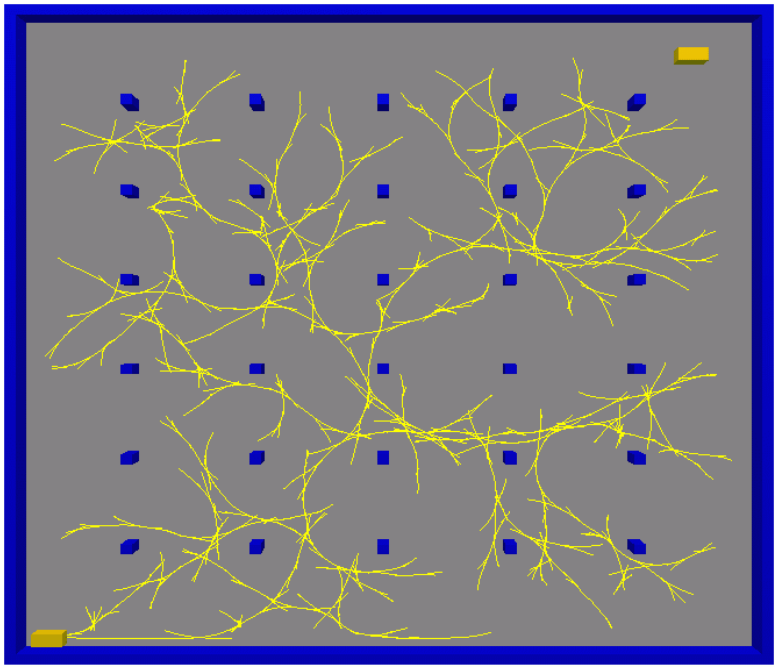


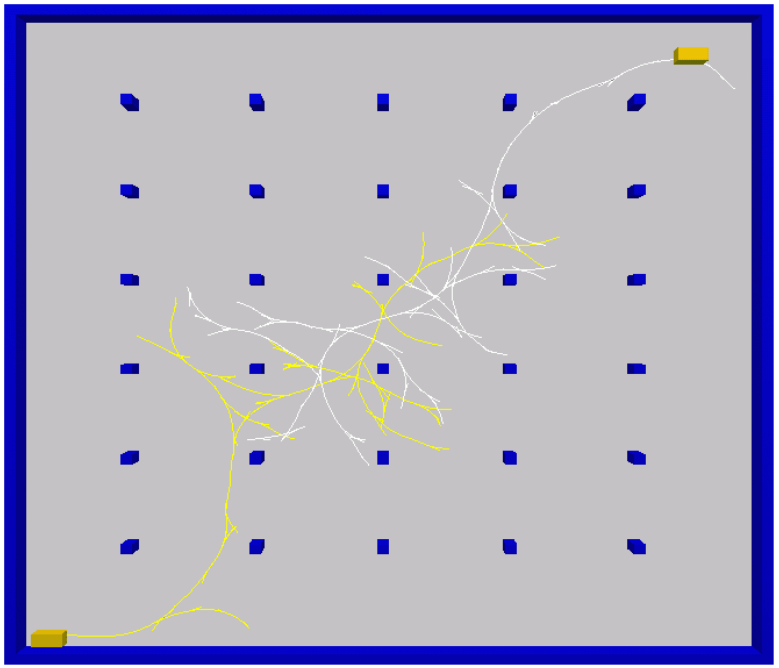
- Heurística(estado)="suma de las distancias Manhattan de cada pieza a donde debería estar en el estado final"



Aplicaciones en robótica

- Planificación de movimientos de un robot tipo carro





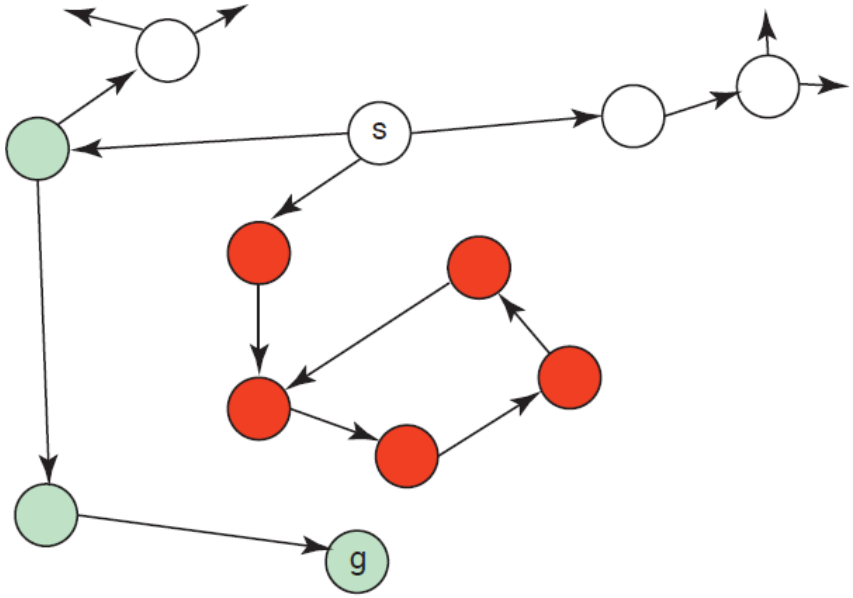
- RRT BUAP: Aplicaciones en robótica, animación por computadora.
- SRT BUAP: Exploración en ambientes desconocidos

Ejemplo de funciones heurísticas

- Si los nodos son puntos en un plano euclidiano y el costo es la distancia, $h(n)$ puede ser la distancia en línea recta desde n hasta la meta más cercana.
- Si los nodos son ubicaciones (localidades) y el costo es tiempo, podemos usar la distancia a una meta dividida por la velocidad máxima.
- Si el objetivo es recolectar todas las monedas y no quedarse sin combustible, el costo es una estimación de cuántos pasos se tomarán para recolectar el resto de las monedas, recargar combustible cuando sea necesario y volver a la posición del objetivo.
- Se puede encontrar una función heurística resolviendo una versión más simple (menos restringida) del problema.

Búsqueda primero el mejor, I

- Idea: selecciona el camino cuyo final está más cerca de una meta de acuerdo con la función heurística.
- La búsqueda primero el mejor selecciona un camino en la frontera con un valor *h* mínimo.
- Trata la frontera como una cola de prioridad ordenada por *h*.
- Grafo ilustrativo de la búsqueda.



Búsqueda primero el mejor, II

Complejidad:

- ¿La búsqueda primero el mejor garantiza la búsqueda del camino más corto o el camino con menos arcos?
- ¿Qué sucede en los grafos infinitos o en los grafos con ciclos si hay una solución?
- ¿Cuál es la complejidad del tiempo en función de la longitud del camino seleccionado?
- ¿Cuál es la complejidad del espacio en función de la longitud del camino seleccionado?
- ¿Cómo afecta el objetivo a la búsqueda?

Búsqueda primero el mejor, III

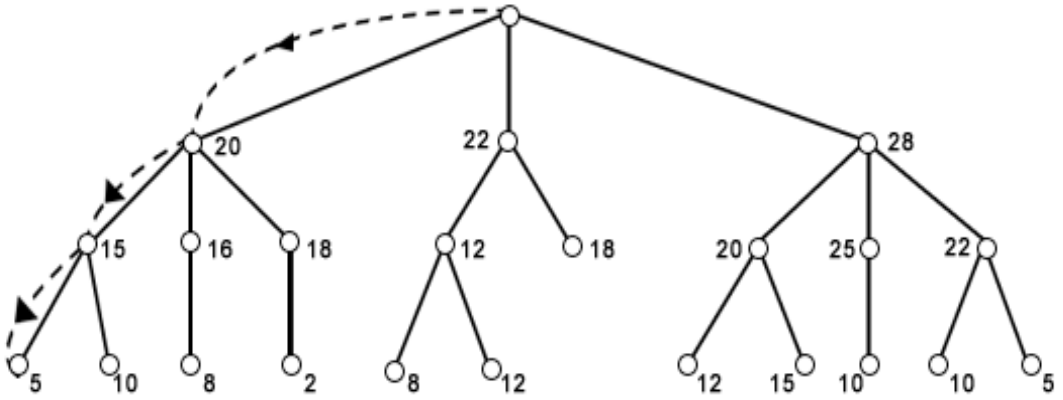
- Es un procedimiento general de búsqueda en grafos.
- Esta búsqueda combina:
  - La búsqueda en profundidad (puede encontrar una solución sin expandir todos sus nodos).
  - La búsqueda en anchura (no queda atrapada en caminos sin salida).
  - El método del gradiente (tiene en cuenta los valores heurísticos de los estados).

- Consiste en recorrer el grafo de búsqueda eligiendo en cada momento el nodo que tenga mejor valor para una determinada función heurística *f*.
- A diferencia del método de gradiente, cuando el camino actual se aleja de la meta, se pueden retomar caminos de exploración abandonados anteriormente.

  Por lo tanto, este método no pertenece al grupo de métodos de estrategia irrevocable, pertenece al conjunto de métodos de estrategias de exploración de alternativas o métodos denominados vuelta atrás o backtracking.

Función sucesor

Función SUCESOR(NODO, OPERADOR)
1. Hacer ESTADO-SUCESOR igual a APLICA(OPERADOR, ESTADO(NODO))
2. Si ESTADO-SUCESOR = NO-APLICABLE
   devolver NO-APLICABLE
   en caso contrario,
   devolver un nodo cuyo estado es ESTADO-SUCESOR, cuyo camino es el resultado de añadir OPERADOR a CAMINO(NODO) y cuya heurística es la de ESTADO-SUCESOR



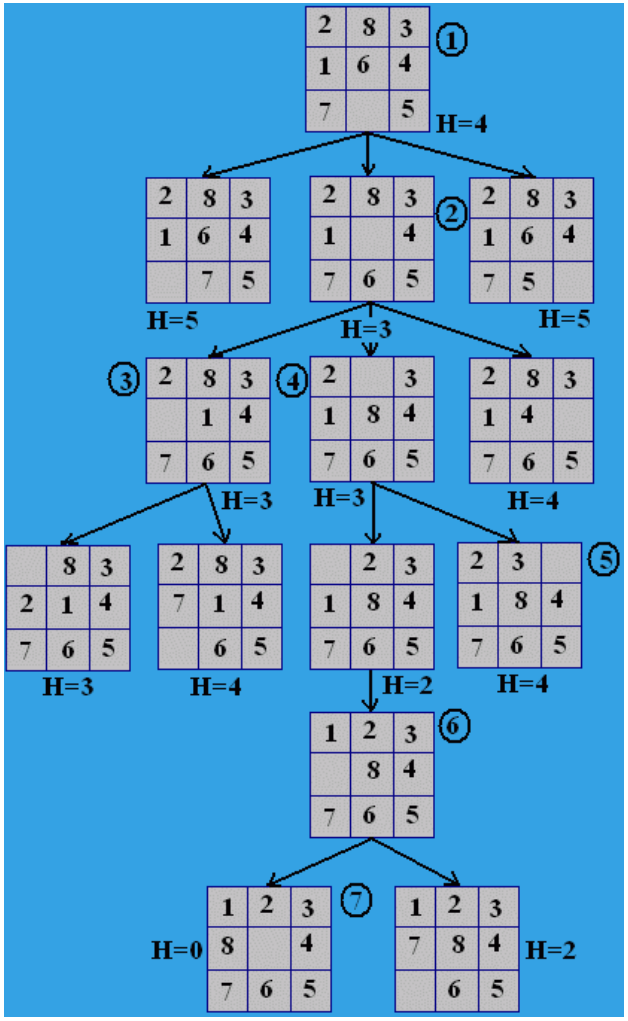
Best-first search of a tree

Algoritmo de primero el mejor

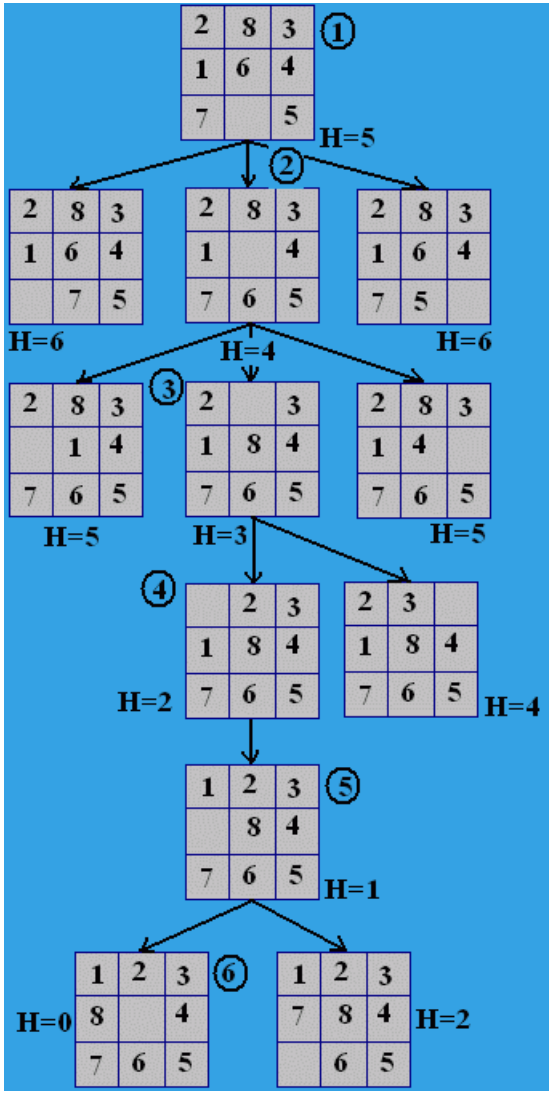
FUNCION BUSQUEDA-POR-PRIMERO-EL-MEJOR()

1. Hacer ABIERTOS la cola formada por el nodo inicial (es decir, el nodo cuyo estado es *ESTADO-INICIAL*, cuyo camino es vacío y cuya heurística es la *ESTADO-INICIAL*);
2. Mientras que ABIERTOS no esté vacía,
   - 2.1 Hacer ACTUAL el primer nodo de ABIERTOS
   - 2.2 Hacer ABIERTOS el resto de ABIERTOS
   - 2.3 Poner el nodo ACTUAL en CERRADOS.
   - 2.4 Si ES-ESTADO-FINAL(ESTADO(ACTUAL)),
     - 2.4.1 devolver el nodo ACTUAL y terminar
     - 2.4.2 en caso contrario
       - 2.4.2.1 Hacer NUEVOS-SUCESORES la lista de nodos SUCESORES(ACTUAL) cuyo estado no está ni en ABIERTOS ni en CERRADOS
       - 2.4.2.2 Hacer ABIERTOS el resultado de incluir NUEVOS-SUCESORES en ABIERTOS y ordenar en orden creciente de sus heurísticas
3. Devolver FALLO.

Ejemplo con la primera heurística



Ejemplo con la segunda heurística

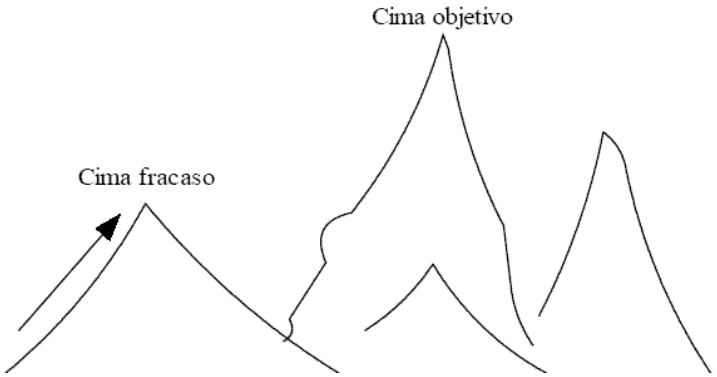


Propiedades de la búsqueda

- Complejidad:
  - *r*: factor de ramificación.
  - *p*: profundidad de la solución.
  - Complejidad en espacio: $O(r^p)$
  - Complejidad en tiempo: $O(r^p)$
  - En la práctica, la complejidad depende del problema concreto y de la calidad de la heurística usada.
- No es completa
  - Por ejemplo, una mala heurística podría hacer que se tomara un camino infinito en el problema del agente viajero.
- No es minimal
  - La heurística podría guiar hacia una solución no minimal

Búsqueda en escalada

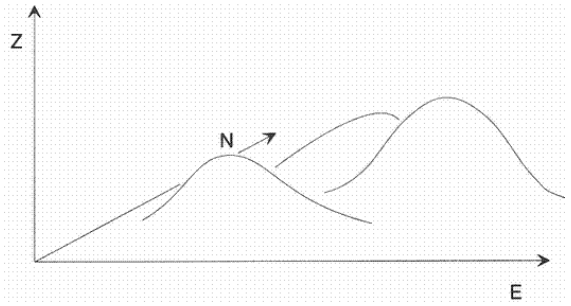
- El algoritmo es una evolución de la búsqueda en profundidad.
- Ahora no seleccionaremos un nodo hijo cualquiera del nodo padre a expandir o aquel que aparece más a la izquierda, sino que tomaremos el nodo más prometedor en el camino hacia la solución.
- La elección del nodo más prometedor se hace de acuerdo a una heurística que evalúa cuál es la distancia que resta hasta alcanzar la solución.



Falda de colina

Meseta

Risco



Algoritmo de la búsqueda en escalada

1. Crear la variable local ACTUAL inicializándola con el nodo heurístico cuyo estado es el *ESTADO-INICIAL*, cuyo camino es la lista vacía y cuya heurística es la del *ESTADO-INICIAL*.

2. Repetir mientras que el nodo ACTUAL no sea nulo:

   2.1 Si el estado del nodo ACTUAL es un estado final, devolver el nodo ACTUAL y terminar;

   2.2 SINO cambiar actual por su mejor sucesor (es decir, uno de sus sucesores cuya heurística sea menor que la de ACTUAL y menor o igual que las heurísticas de los restantes sucesores, si existen y NIL en caso contrario).

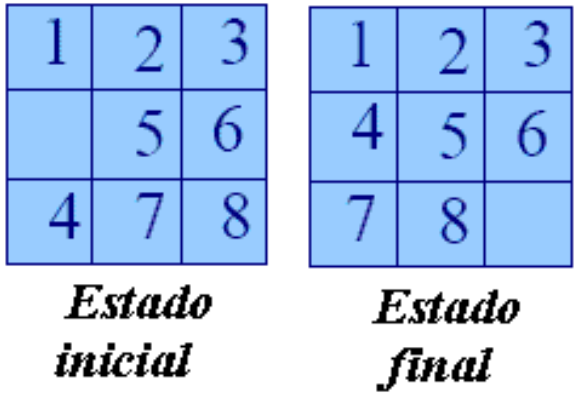
3. Si se ha encontrado un nodo objetivo, reportar éxito, si no, fallo.

Desventajas de la búsqueda

- Existe una gran dependencia de la definición correcta de la función de evaluación. Este método exige que la función aplicada sea lo más *informativa* posible. Es decir:
  - debe tomar un máximo o un mínimo en el nodo objetivo;
  - debe ser sencilla de calcular;
  - no debe tener máximos ni mínimos locales.
- Este método no es completo.
- No es mínima.
- Problemas del ascenso de colina:
  - Falda de colina, ocurre cuando hay picos secundarios (máximo local).
  - Mesetas, se presenta cuando hay un área casi plana que separa los picos.
  - Risco, es un tipo especial de máximo local, difícil de atravesar con movimientos simples.

Ejemplo del 8-puzle con escalada

- Heurísticas:
  - Fichas mal colocadas
  - Fichas bien colocadas



- Como se interpreta el fallo del algoritmo?
- Porqué falla el algoritmo?

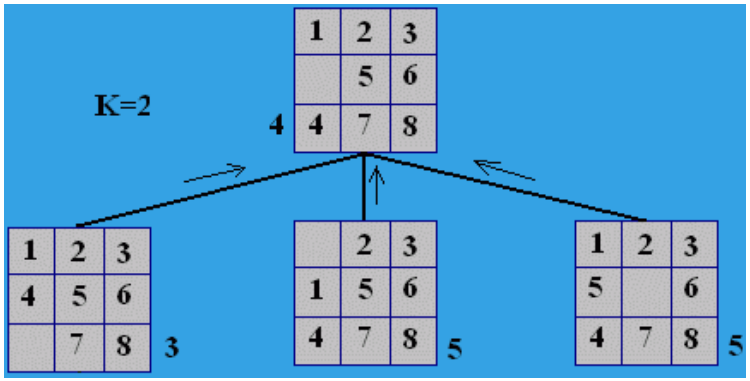
Siempre que se pueda identificar una función de evaluación que tenga la propiedad de ir creciendo (o decreciendo) hasta el valor que tenga la función en el nodo meta.

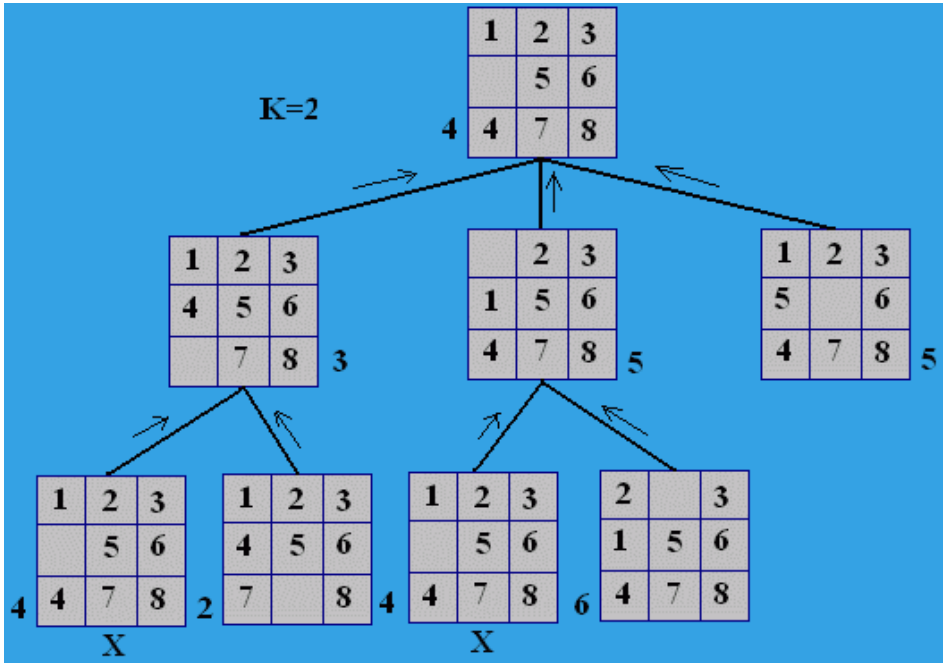
Beam search

- Es una variación de la búsqueda en escalada. *k* es el número de ramas a expandir.

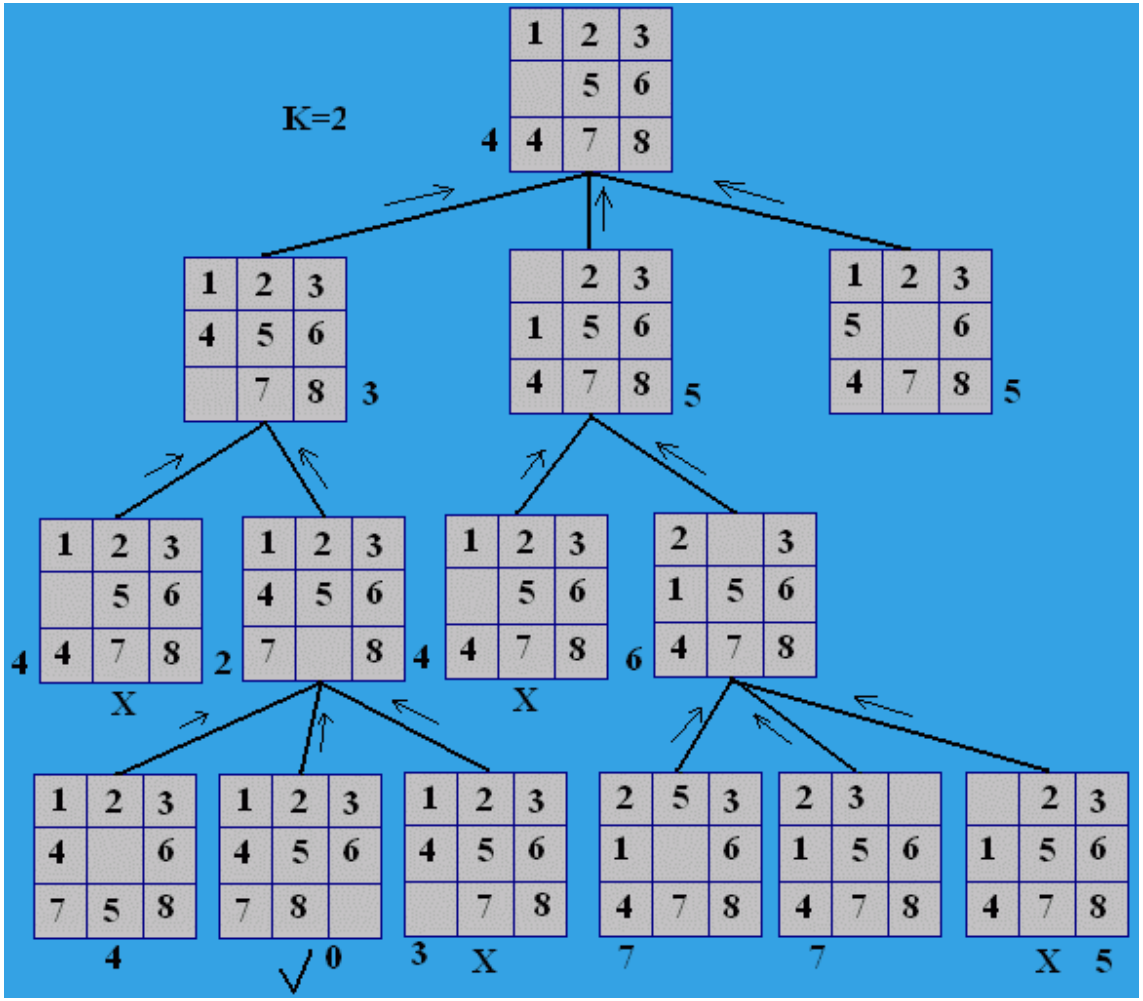
```
Procedimiento Haz (Estado-inicial Estado-final k)
ABIERTA = Estado-inicial, EXITO = Falso
Hasta que ABIERTA esté vacía o EXITO
   ABIERTA = todos los sucesores de los nodos de ABIERTA
   Si algún nodo de ABIERTA es Estado-final
   Entonces EXITO = Verdadero
   Si no, evaluar cada nodo con la función de evaluación f(n)
            ABIERTA = k mejores nodos de ABIERTA
Si EXITO
Entonces Solución = camino desde nodo del Estado-inicial al nodo N por los
     apuntadores
Si no, Solución = fracaso
```

Ejemplo de la búsqueda en haz, I





Ejemplo de la búsqueda en haz, II



Espacio de estados con costo

- En algunos problemas, existe un costo (positivo) asociado a la aplicación de los operadores.
  - Función costo-de-aplicar-operador(estado, operador).
  - Asumimos que operador **es aplicable** a estado.
- Costo asociado a un camino:
  - Suma de los costos de la aplicación de cada uno de sus operadores.
  - ***Una solución óptima*** es una solución con costo mínimo.
- En algunos problemas, no existe un costo claramente especificado.
  - En ese caso, costo-de-aplicar-operador(estado, operador)=1, y las soluciones óptimas son las minimales (las de menor número de operadores).
- Costo en el problema del agente viajero:
  - costo-de-aplicar-operador(estado, operador)=distancia-euclideana(estado, aplica(operador,estado)).

Idea de la búsqueda óptima

- Búsqueda óptima
  - Analizar primero los nodos con menor costo.
  - Ordenar la cola de abiertos por costo, de menor a mayor.
- De esta manera, cuando se llega por primera vez a un estado final, se llega con el menor costo posible.
- Se trata de una búsqueda ciega:
  - No usa conocimiento para guiar la búsqueda hacia el objetivo
  - Caso particular: búsqueda en anchura.
- Nodo de búsqueda: estado + camino + costo del camino
  - Funciones de acceso: ESTADO(NODO), CAMINO(NODO) Y COSTO-CAMINO(NODO)
- Sucesores de un nodo con costo:

```
FUNCION SUCESOR(NODO, OPERADOR)
1. Hacer ESTADO-SUCESOR igual a APLICA(OPERADOR,ESTADO(NODO))
2. Si ESTADO-SUCESOR = NO-APLICABLE
       devolver NO-APLICABLE
   en caso contrario,
       devolver un nodo cuyo estado es ESTADO-SUCESOR, cuyo camino es el
        resultado de añadir OPERADOR a CAMINO(NODO) y cuyo costo es
        COSTO-CAMINO(ESTADO))+COSTO-DE-APLICAR-OPERADOR(
        ESTADO(NODO), OPERADOR)
```

Algoritmo de la búsqueda óptima

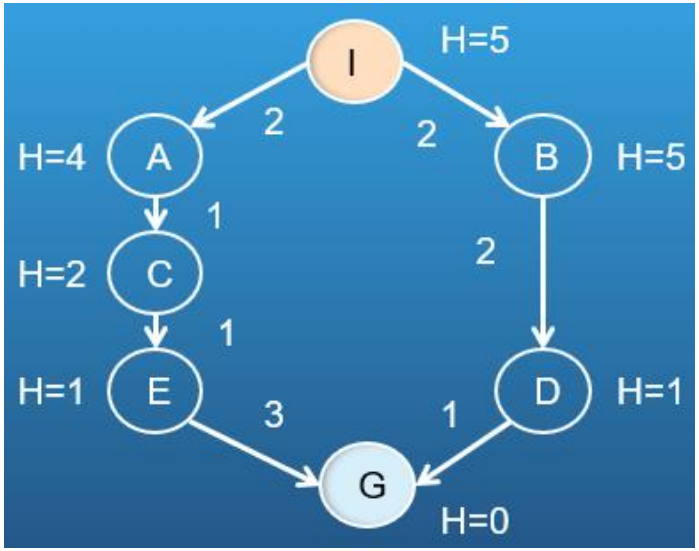
FUNCION BUSQUEDA-OPTIMA()
1. Hacer ABIERTOS la cola formada por el nodo inicial (es decir, el nodo cuyo estado es *ESTADO-INICIAL*, cuyo camino es vacío y cuyo costo es 0); Hacer CERRADOS vacío
2. Mientras que ABIERTOS no esté vacía
   2.1 Hacer ACTUAL el primer nodo de ABIERTOS
   2.2 Hacer ABIERTOS el resto de ABIERTOS
   2.3 Poner el nodo ACTUAL en CERRADOS
   2.4 Si ES-ESTADO-FINAL(ESTADO(ACTUAL)),
      2.4.1 devolver el nodo ACTUAL y terminar.
      2.4.2 en caso contrario,
         2.4.2.1 Hacer NUEVOS-SUCESORES la lista de nodos de SUCESORES(ACTUAL) para los que no existe ni en ABIERTOS ni en CERRADOS un nodo con el mismo estado y menor costo
         2.4.2.2 Hacer ABIERTOS el resultado de incluir NUEVOS-SUCESO -RES en ABIERTOS y ordenar todo en orden creciente de los costos de los caminos de los nodos
3. Devolver FALLO.

Propiedades de la búsqueda óptima

- Complejidad:
  - r: factor de ramificación.
  - p: profundidad de la solución.
  - Complejidad en espacio: $O(r^p)$
  - Complejidad en tiempo: $O(r^p)$
- Es completa.
- Es óptima.
- Salvo en espacios de estados pequeños, en la práctica esta búsqueda no es posible, debido a la cantidad de tiempo y espacio consumidos.

Heurísticas para soluciones óptimas, I

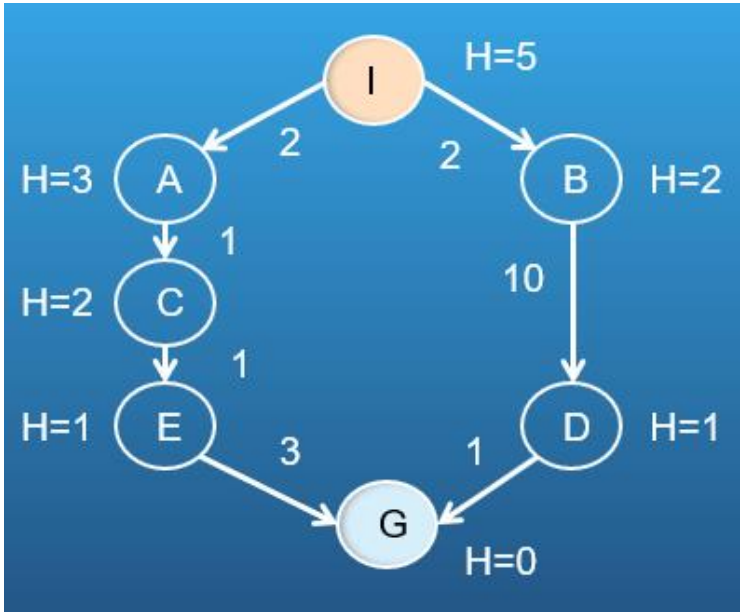
- ¿Podemos usar búsqueda por primero el mejor para acelerar la búsqueda y aún estar seguro de obtener soluciones óptimas? En general, **NO**.
- Ejemplo 1:



- Solución encontrada por primero el mejor: I-A-C-E-G (sub-óptima)
- Causa: la heurística *sobrestima* el costo real.

Heurísticas para soluciones óptimas, II

- Ejemplo 2:



- Solución encontrada por primero el mejor: I-B-D-G (sub-óptima)
- causa: no se ha tomado en cuenta *los costos* de los caminos ya recorridos.

Búsqueda A*

- Objetivo de la búsqueda A*:
  - conseguir buena soluciones (óptimas).
  - Ganar eficiencia (podando el árbol de búsqueda).

- Idea: asignar a cada nodo *n* un valor $f(n) = g(n) + h(n)$,
  - $g(n)$: costo del camino hasta *n*
  - $h(n)$: heurística del nodo, *estimación* del costo *de un camino óptimo* desde *n* hasta un estado final
  - $f(n)$: estimación del costo total de una solución óptima *que pasa por n*

- Seleccionar siempre el nodo con menor valor de *f*
  - Ordenando la cola de ABIERTOS en orden creciente respecto a *f*.

- La implementación de nodos con costo y considerando la heurística, es como sigue:

Implementación

- Nodo de búsqueda: estado + camino + costo del camino + costo-mas-heurística
- Funciones de acceso: ESTADO(NODO), CAMINO(NODO), COSTO-CAMINO(NODO) y COSTO-MAS-HEURISTICA(NODO).
- Sucesores de un nodo con costo y heurística

```
FUNCION SUCESOR(NODO,OPERADOR)
1. Hacer ESTADO-SUCESOR igual a APLICA(OPERADOR,ESTADO(NODO))
2. Si ESTADO-SUCESOR = NO-APLICABLE
   devolver NO-APLICABLE
 en caso contrario,
   devolver un nodo cuyo estado es ESTADO-SUCESOR, cuyo camino es el
  resultado de añadir OPERADOR a CAMINO(NODO) y cuyo costo es
   COSTO(CAMINO(ESTADO)) + COSTO-DE-APLICAR-OPERADOR(ESTADO(NODO),
   OPERADOR) y cuyo costo-mas-heurística es el costo anterior mas HEURISTICA(
   ESTADO(NODO))
```

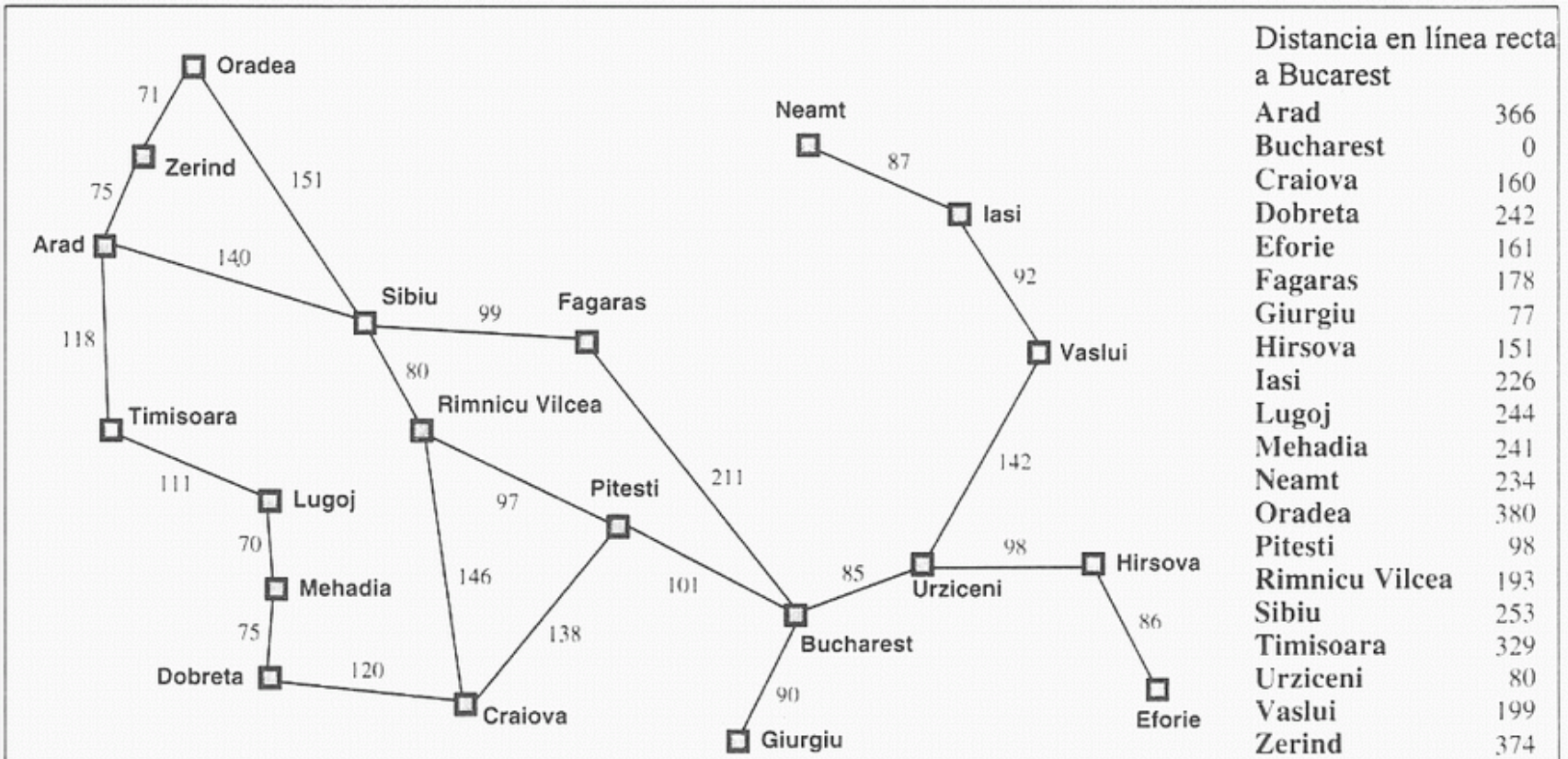
- La función SUCESORES(NODO), es como en temas anteriores.

Algoritmo A*

FUNCION BUSQUEDA-A*()

1. Hacer ABIERTOS la cola formada por el nodo inicial (es decir, el nodo cuyo estado es *ESTADO-INICIAL*, cuyo camino es vacío, cuyo costo es 0, y cuyo costo-mas-heurística es HEURISTICA(*ESTADO-INICIAL*));
   Hacer CERRADOS vacío
2. Mientras que ABIERTOS no esté vacía,
   2.1 Hacer ACTUAL el primer nodo de ABIERTOS
   2.2 Hacer ABIERTOS el resto de ABIERTOS
   2.3 Poner el nodo ACTUAL en CERRADOS.
   2.4 Si ES-ESTADO-FINAL(ESTADO(ACTUAL)),
      2.4.1 devolver el nodo ACTUAL y terminar.
      2.4.2 en caso contrario,
         2.4.2.1 Hacer NUEVOS-SUCESORES la lista de nodos de SUCESORES (ACTUAL) para los que no existe ni en ABIERTOS ni en CERRADOS nodo con el mismo estado y menor costo
         2.4.2.2 Hacer ABIERTOS el resultado de incluir NUEVOS-SUCESORES en ABIERTOS y ordenar todo en orden creciente del costo-mas-heurística de los nodos
3. Devolver FALLO

Ejemplo, mapa de Rumania

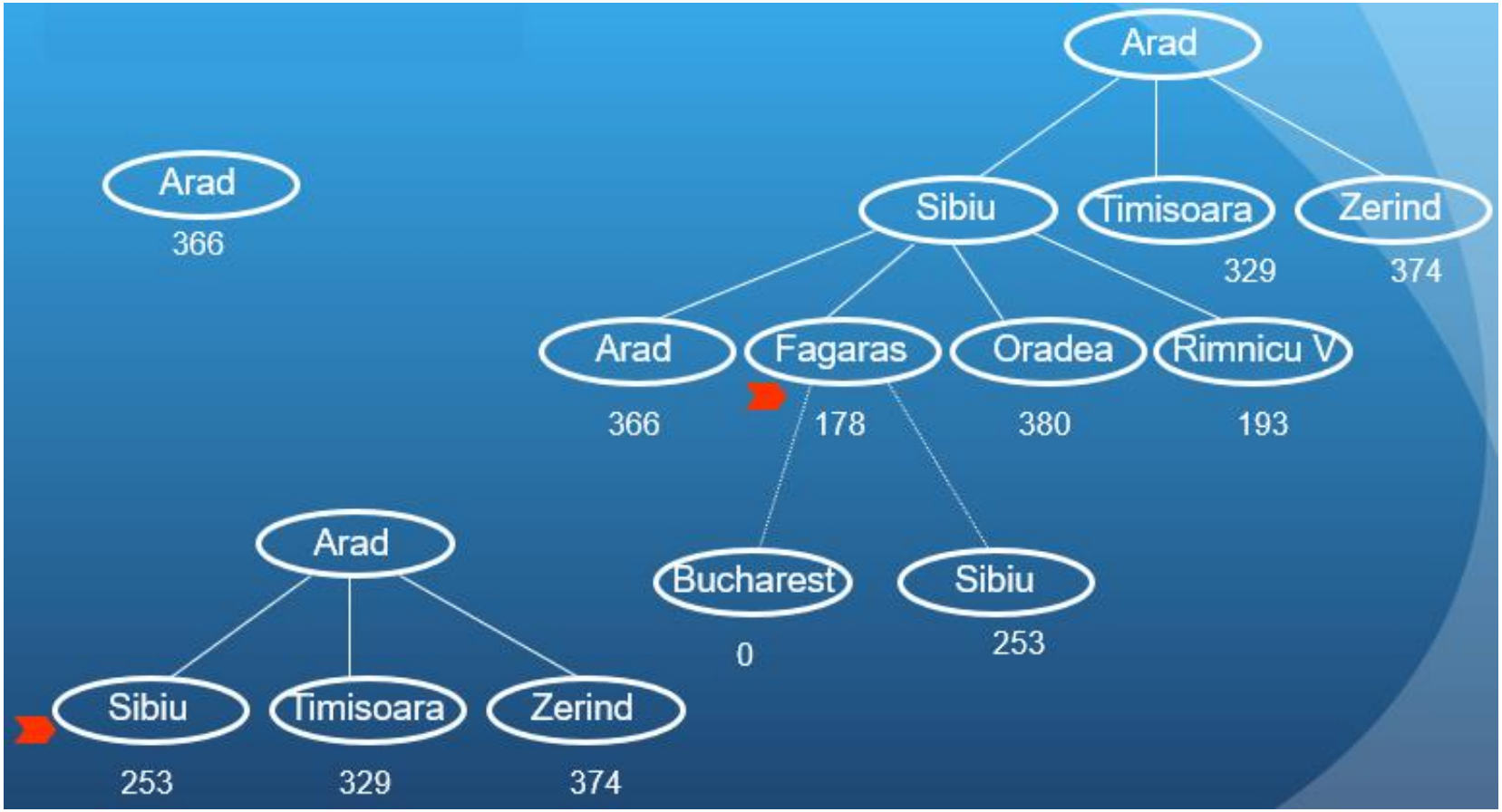


Distancia en línea recta a Bucarest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Mapa de Rumania, en donde se indica la extensión de las carreteras en kilómetros así como las distancias en línea recta a Bucarest.

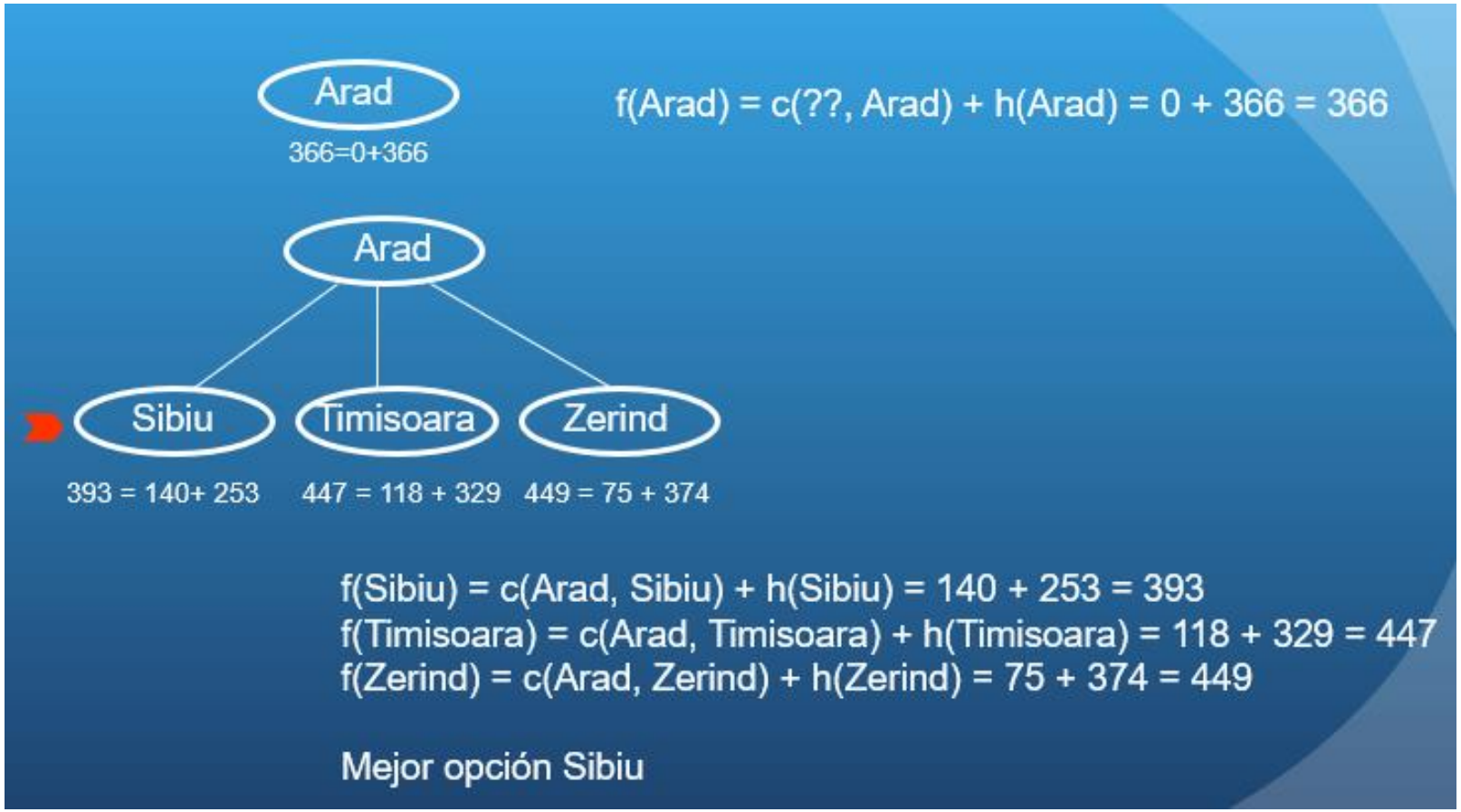
Ejemplo, búsqueda primero el mejor

- Ir de Arad a Bucharest

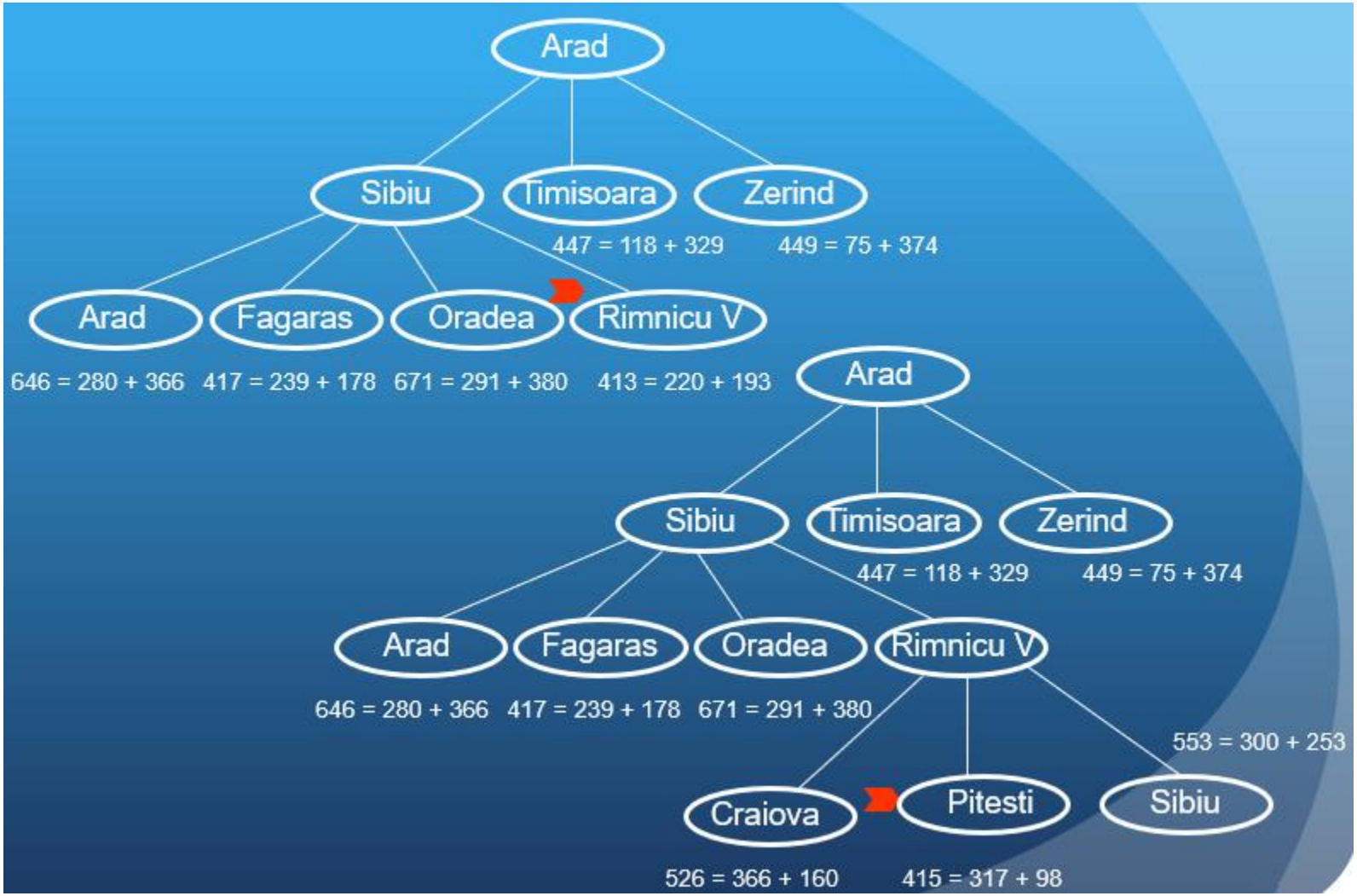


Ejemplo con A*, I

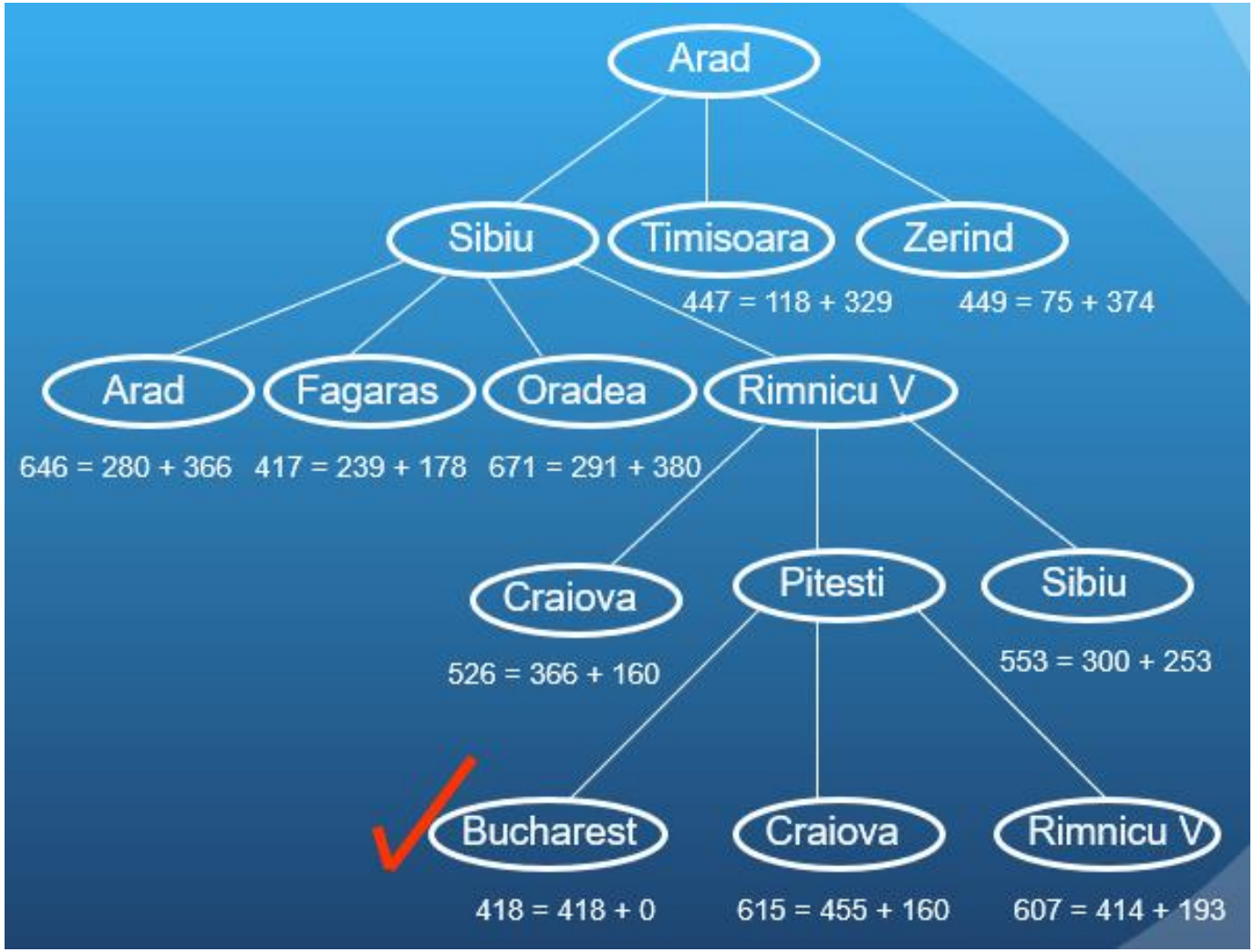
- Ir de Arad a Bucharest



Ejemplo con A*, II



Ejemplo con A*, III



Propiedades de A*, I

- Sea *h*(n)* costo de un camino óptimo desde *n* hasta un estado final
  - *f*(n) = g(n) + h*(n)* costo total de una solución óptima *que pasa por n*
- En la práctica, no conocemos *h**
  - Usamos una función heurística *h* que estima *h**

- Posibilidades:
  - *h=0*, búsqueda óptima, no hay poda(no hay reducción del árbol de búsqueda)
  - *h=h**, estimación perfecta, no hay búsqueda
  - $0 \leq h \leq h^*$, *heurística admisible*, solución óptima garantizada
  - Para algún nodo *n*, *h(n) > h*(n)*, no se puede garantizar que la solución encontrada sea óptima

- Usando una heurística admisible:
  - Es completa.
  - Encuentra siempre una solución óptima.

Propiedades de A*, II

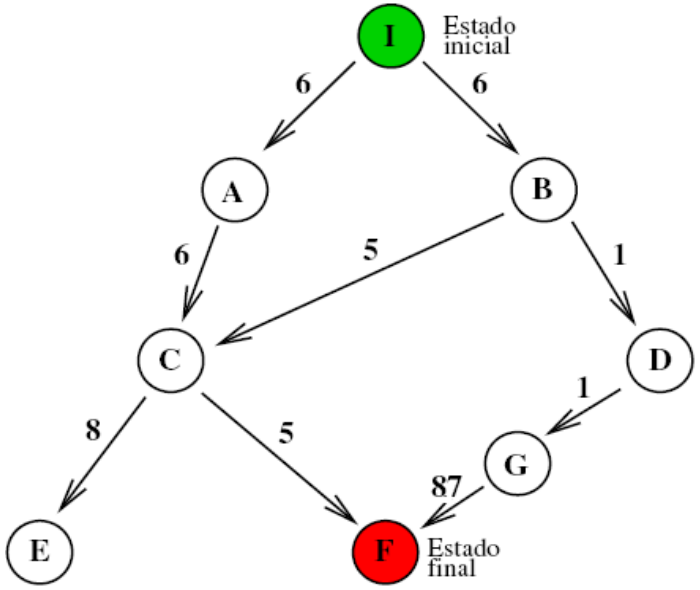
- Complejidad:
  - *r:* factor de ramificación.
  - *p*: profundidad de la solución.
  - Complejidad en espacio: $O(r^p)$
  - Complejidad en tiempo: $O(r^p)$
  - En la práctica, el costo en tiempo y espacio depende de la calidad de la heurística usada.

Invención de heurísticas

- Estimación del costo restante hasta la solución óptima
- Comparación de heurísticas:
  - Si $0 \leq h_1 \leq h_2 \leq h^*$, entonces h_2 está más informada que h_1 y mejora la eficiencia
- Técnicas para encontrar heurísticas
  - Relajación del problema
  - Combinación de heurísticas admisibles
  - Uso de información estadística
- Evaluación eficiente de la función heurística

Ejemplo de búsquedas heurísticas, I

- El siguiente grafo representa un problema de espacio de estados. Los nodos del grafo son los estados del problema, las aristas conectan estados con sus sucesores y el valor numérico de cada arista representa el costo de pasar de un estado a su sucesor:
- El estado inicial del problema es I y el único estado final es F. Se consideran las funciones heurísticas H_1 y H_2 dadas en la tabla:



Estado	H_1	H_2
I	16	20
A	8	8
B	10	6
C	3	12
D	2	2
E	9	9
F	0	0
G	1	1

Ejemplo de búsquedas heurísticas, II

a) Construir los árboles de búsqueda generados por los algoritmos de búsqueda en profundidad, anchura y búsqueda primero el mejor. Indicar en cada caso, el orden en el que se analizan los nodos y la solución obtenida junto con su costo.

b) Análogamente, con el algoritmo de búsqueda A*, usando las dos heurísticas H_1 y H_2 definidas anteriormente. Son admisibles? Justificar las respuestas.