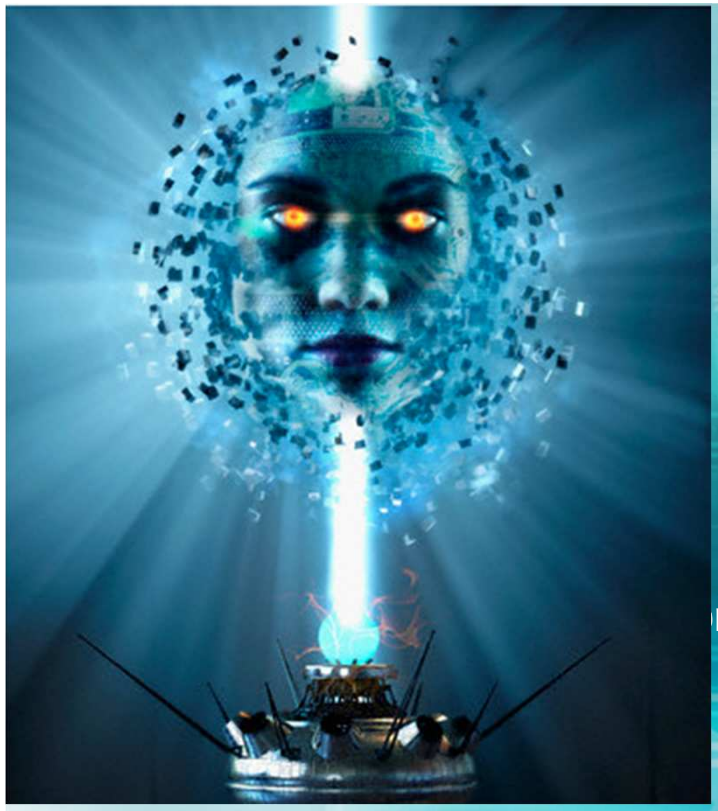


mpany slogan ”

Resolución de Problemas Mediante Búsqueda

Ing. Jesús Eduard López Quentasi

Búsqueda en Amplitud

La búsqueda en amplitud puede estar orientada a recorrer el árbol o grafo correspondiente por niveles. Para conseguirlo se utiliza una estructura tipo cola (FIFO) en la que se van introduciendo los nodos que son generados este tipo de exploración recibe el nombre de búsqueda en amplitud y garantiza la obtención de la solución de menor coste (óptima), si es que esta existe.

Algoritmo de Búsqueda en Amplitud

Algoritmo de Búsqueda en amplitud.

Crear una lista de nodos llamada abierta y asignarle el nodo raíz, que representa el estado inicial del problema planteado.

Hasta que **ABIERTA** éste vacía o se encuentre una meta, realizar las siguientes acciones:

Extraer el primer nodo de **ABIERTA** y llamarlo m.

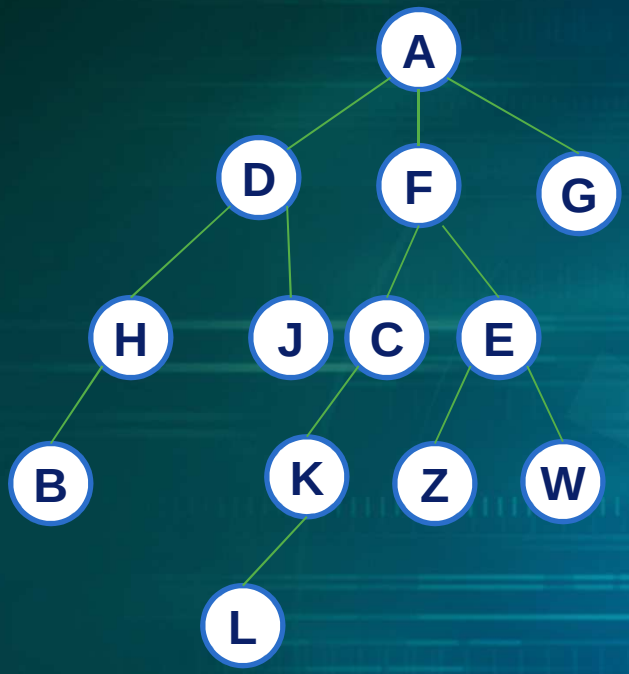
Expandir m (generar todos sus sucesores). Para cada operador aplicable y cada forma de aplicación.

Aplicar el operador a m, obtener un nuevo estado y crear un puntero que permita saber que su predecesor es m.

Si el nuevo estado generado es meta, salir del proceso iterativo iniciado en 2,2 y devolver dicho estado.

Incluir el nuevo estado al final de ABIERTA, (una vez completado este proceso para todos los sucesores de m cuando no se haya encontrado antes una meta se continua el proceso iterativo en el paso 2).

Ejemplo



Búsqueda en Profundidad

La búsqueda en profundidad se centra en expandir un único camino de la raíz; en el caso de llegar a un "callejón sin salida" se retrocede hasta el nodo más cercano desde donde se puede tomar una rama alternativa para poder seguir avanzando. Para poder llevar a cabo este tipo de búsqueda debe utilizarse una estructura de tipo pila (LIFO) que vaya almacenando los nodos generados, suele establecerse, por otra parte el llamado límite de exploración, que marca la máxima longitud que puede alcanzar cualquier camino desde la raíz durante el proceso de búsqueda,

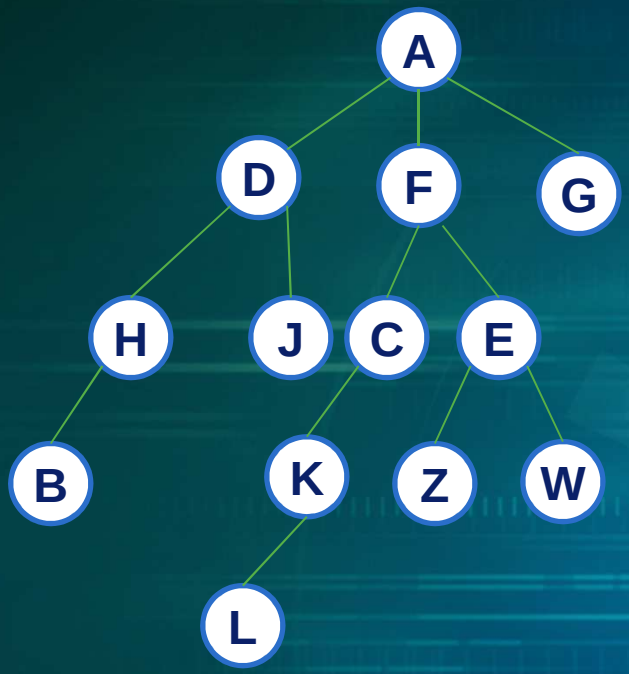
Una ligera variante de la búsqueda en profundidad es la búsqueda con retroceso, en ésta última, en lugar de generar todos los sucesores al expandir un nodo, se genera un único sucesor en cada paso.

Algoritmo de Búsqueda en Profundidad

Algoritmo de Búsqueda en profundidad.

1. Crear una lista de nodos llamada abierta y asignarle el nodo raíz, que representa el estado inicial del problema planteado.
2. Hasta que se encuentre una meta o se devuelva un fallo, realizar las siguientes acciones:
(1) Si **ABIERTA** está vacía terminar con fallo en caso contrario continuar.
(2) Extraer el primer nodo de **ABIERTA** y denominarlo m
(3) Si la profundidad de m es igual a lp (límite de profundidad), regresar a 2; caso contrario, continuar.
(4) Expandir m creando punteros hacia ese nodo desde todos sus sucesores, de forma que pueda saberse cuál es su predecesor. Introducir dichos sucesores al principio de abierta siguiendo un orden arbitrario (La Falta de orden refleja el carácter no informado de este procedimiento).
(4.1). Si algún sucesor de m es meta, abandonar el proceso iterativo señalado en 2, devolviendo el camino de la solución, que se obtiene recorriendo los punteros de sus antepasados.
(4.2). Si algún sucesor de m se encuentra en un callejón sin salida, eliminarlo de **ABIERTA** (Se continua el proceso iterativo en el paso 2).

Ejemplo



Búsqueda de costo uniforme

Otra variante es la búsqueda bidireccional, donde se parte de un estado inicial desde el que se busca la meta y de un estado meta desde donde se busca el estado inicial; con la única condición de que una de las dos búsquedas anteriores sea en amplitud, se garantiza que si existe algún camino solución, en algún momento las dos búsquedas pasaran por un estado común desde el que se podrá reconstruir dicho camino.

En el caso de la **búsqueda de costo uniforme** se modifica la estrategia de búsqueda por amplitud en el sentido de expandir siempre el nodo de menor costo en el margen (medido por el costo de la ruta g(n)) en vez del nodo de menor profundidad. No es difícil darse cuenta de que la búsqueda por amplitud no es sino una búsqueda de costo uniforme en la que g(n) = Profundidad (n).
Si se cumplen ciertas condiciones, es seguro que la primera solución encontrada será la más barata, puesto que si hubiera una ruta más barata que fuese solución, ya se habría expandido.

Búqueda de costo uniforme

Mediante la búsqueda por costo uniforme se puede encontrar la solución más barata, siempre y cuando se satisfaga un requisito muy sencillo: el costo de la ruta nunca debe ir disminuyendo conforme avanzamos por la ruta. En otras palabras, es importante que:

$$g(\text{Sucesor}(n)) \geq g(n)$$

en todos los nodos.

La restricción de que el costo no vaya disminuyendo tiene sentido si se considera como costo de la ruta de un nodo a la suma de los costos de los operadores que configuran la ruta el costo de todos los operadores no es negativo, el costo de una ruta nunca ira disminuyendo conforme se avanza, y mediante la búsqueda de costo uniforme será posible encontrar la ruta más barata si tener que explorar el árbol de búsqueda en su totalidad

Búsqueda Bidireccional.

La búsqueda bidireccional es, básicamente, una búsqueda simultánea que avanza a partir del estado inicial y que retrocede a partir de la meta y que se detiene cuando ambas búsquedas se encuentran en algún punto intermedio, En el caso de problemas cuyo factor de ramificación de ramificación es b en ambas direcciones, la búsqueda bidireccional puede ser muy útil. Si como de costumbre, se supone que existe una solución cuya profundidad es d, entonces la solución estará a O(2bd/2)=(Od/2) pasos, puesto que en las búsquedas hacia adelante y hacia atrás solo recorre la mitad del trayecto. Un ejemplo Concreto: si b=10 y d = 6, una búsqueda preferente por amplitud produce 1, 111, 111, nodos en tanto que una búsqueda bidireccional tiene éxito cuando cada una de las direcciones de búsqueda están a profundidad 3, en cuyo caso se generan 2, 222 nodos, en teoría esto es fabuloso, pero antes de poder implantar el algoritmo hay que resolver varias cuestiones.

Búsqueda Bidireccional.

Lo más importante es: ¿Qué significa buscar hacia atrás a partir de la meta? Se definen los predecesores de un nodo n como todos aquellos nodos cuyo sucesor es n. La búsqueda hacia atrás implica la sucesiva generación de predecesores a partir del nodo meta.

Si todos los operadores son reversibles, los conjuntos de predecesor y sucesor son idénticos; en algunos problemas, sin embargo, el cálculo de los predecesores puede resultar muy difícil.

Algoritmo de Búsqueda Bidireccional.

Procedimiento Bidireccional (Estado-inicial Estado-final)

1. ABIERTA-I=Estado-inicial, ABIERTA-M=Estado-final, EXITO=Falso
2. Hasta que alguna de ABIERTA-I o ABIERTA-M estén vacías O EXITO

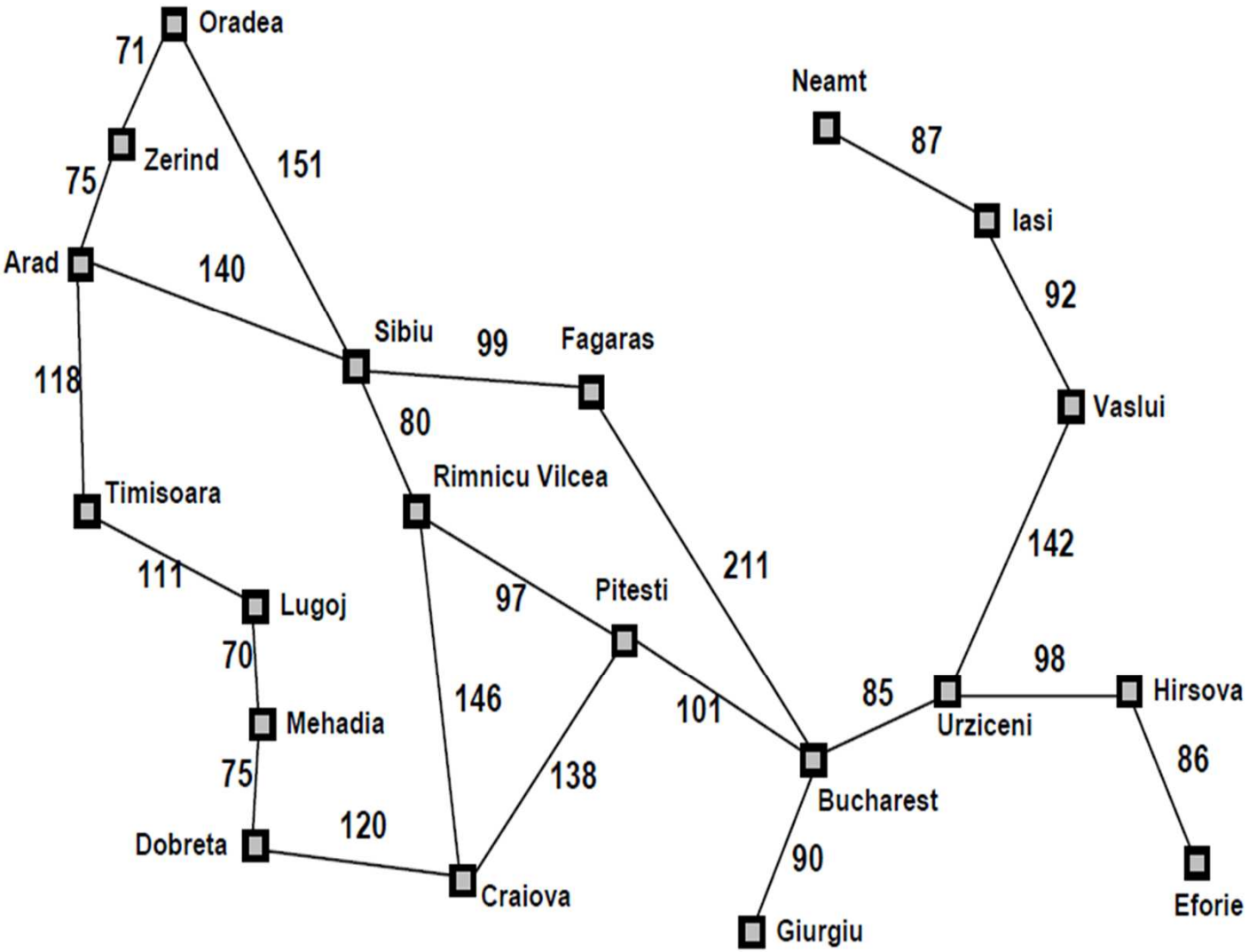
   Quitar de ABIERTA-I el primer nodo, N
   Generar sucesores de N, introducirlos en ABIERTA-I
   SI algún sucesor de N equipara con algún elemento de ABIERTA-M
   ENTONCES EXITO=Verdadero
   SI NO Quitar de ABIERTA-M el primer nodo M
   Generar sucesores de M, introducirlos en ABIERTA-M
   SI algún sucesor de M equipara con algún elemento de ABIERTA-I
   ENTONCES EXITO=Verdadero

3. Si EXITO
   Entonces Solución=camino desde nodo del Estado-inicial al nodo del Estado-final por los punteros
   Si no, Solución=fracaso

Búsquedas Informadas o Heurísticas

- Las estrategias de búsqueda informada o usando un conocimiento específico del problema pueden resolver problemas eficientemente.
- Si analizamos por ejemplo una búsqueda primero el mejor. A partir del algoritmo de búsqueda general se introduce conocimiento específico del problema al insertar los nodos sucesores en la cola mediante una función de evaluación (medida de lo deseable) o lo indeseable de expandir un nodo.
- Por tanto se expande primero el nodo expandido que satisface la función de evaluación (medida deseable) de expandir un nodo.

Ejemplo.



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

Variantes basadas en distintos tipos de funciones de evaluación

- Búsqueda voraz
- Búsqueda A*

Utilizan una estimación del costo del estado actual al objetivo más cercano y tratan de minimizarlo.

Búsqueda Voraz.

La función de evaluación es h(n) (función heurística)
Estimación del costo del nodo al objetivo.

Expande primero el nodo no expandido que supone acercarse más al objetivo.

Para el problema de hallar rutas en Rumania h(n)= distancia en línea recta de n a Bucharest

Ejemplo busqueda Voraz.

Variantes basadas en distintos tipos de funciones de evaluación

- **Propiedades de la búsqueda Voraz:**
- Completa No: puede caer en caminos infinitos, con bucles
- Completa en espacios finitos evitando estados repetidos.
- Tiempo: $O(b^m)$(se deben recorrer todos los nodos)
- Espacio: $O(b^m)$(se deben almacenar todos los nodos)
- Optima No.

Búsqueda A*

La idea es combinar la búsqueda voraz que minimiza el coste al objetivo h(n) y la búsqueda de costo uniforme g(n) que minimiza el coste acumulado. La función de evaluación será

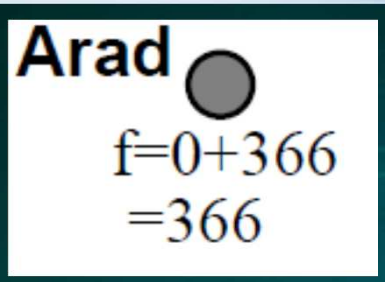
$$f(n) = h(n) + g(n)$$

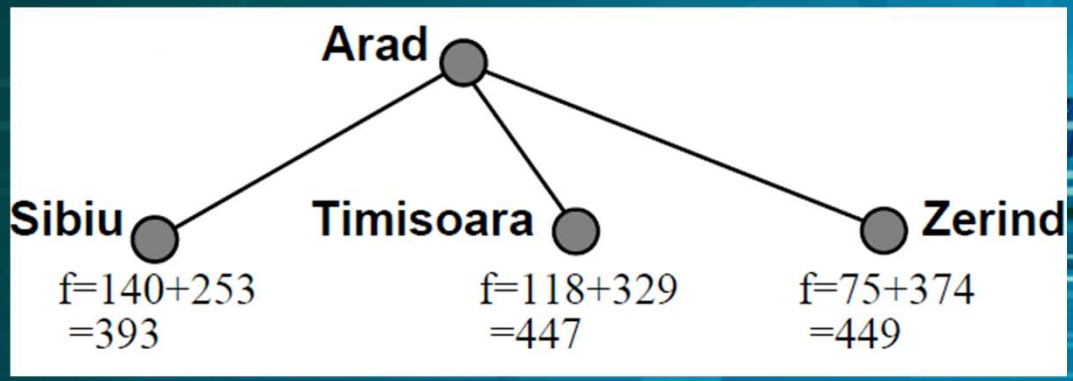
f(n) que es la estimación del coste total del camino más barato al objetivo a través del nodo n expande primero el nodo no expandido que conlleva un camino con menos estimación de costo.

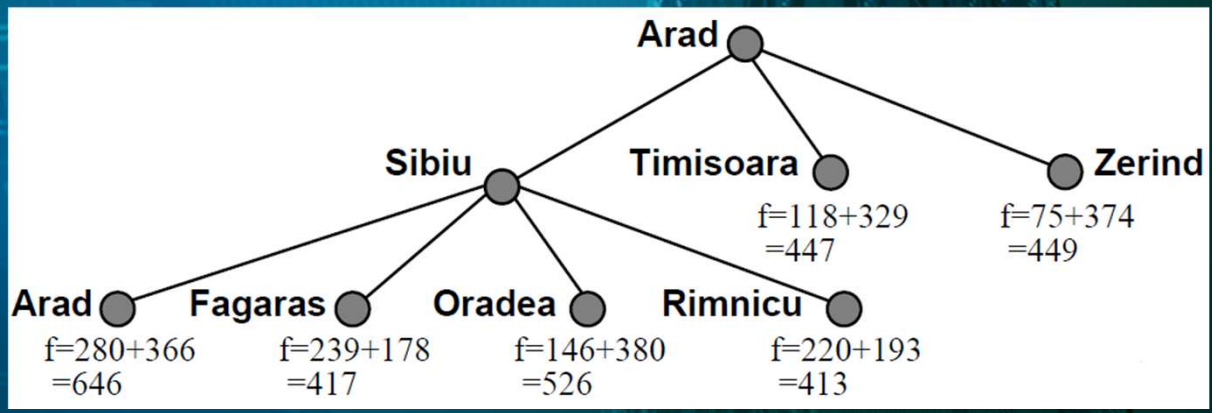
La búsqueda A* es completa y optima.

- La estimación del costo del nodo al objetivo que no supera el costo real.

Ejemplo búsqueda A*.







Comparación de algoritmos de búsqueda

	Algoritmo	Estrategia (Orden de expansión)
NO INFORMADA	Búsqueda en Amplitud Búsqueda en profundidad Búsqueda de costo uniforme	FIFO LIFO g(n)
INFORMADA	Búsqueda Voraz Búsqueda A*	f(n)=h(n) f(n)=g(n)+h(n)

mpany slogan ”

GRACIAS !