INGENIERÍA EN SISTEMAS Y COMUNICACIONES

UDA: INTELIGENCIA ARTIFICIAL

TEMA: RESOLUCIÓN DE PROBLEMAS ESTRATEGIAS DE BÚSQUEDA Y CONTROL

ELABORÓ: DR. EN C. HÉCTOR RAFAEL OROZCO AGUIRRE
CU UAEM VM

Programa de Estudio Por Competencias
Inteligencia Artificial

1. IDENTIFICACIÓN DEL CURSO

ESPACIO ACADÉMICO: CENTRO UNIVERSITARIO UAEM VALLE DE MÉXICO								
PROGRAMA EDUCATIVO: INGENIERIA EN SISTEMAS Y COMUNICACIONES					**Área de docencia:** INGENIERIA APLICADA			
Aprobación por los H. H. Consejos Académico y de Gobierno		**Fecha:**			**Programa elaborado por:** MARICELA QUINTANA LOPEZ, SATURNINO JOB MORALES ESCOBAR		**Fecha de elaboración:** ENERO 2012	
Clave	**Horas de teoría**	**Horas de práctica**	**Total de horas**	**Créditos**	**Tipo de Unidad de Aprendizaje**	**Carácter de la Unidad de Aprendizaje**	**Núcleo de formación**	**Modalidad**
L32310	2	2	4	6	CURSO	OPTATIVA	INTEGRAL	PRESENCIAL
Prerrequisitos (Conocimientos Previos) MATEMÁTICAS DISCRESTAS,LÓGICA MATEMÁTICA			**Unidad de aprendizaje antecedente** NINGUNA			**Unidad de aprendizaje consecuente** NINGUNA		
Programas educativos en los que se imparte: INGENIERIA EN SISTEMAS Y COMUNICACIONES								

Resolución de Problemas Generales

Definiciones de resolución de problemas:

- Espacio del problema
- Resolución de problemas
- Espacio de estados
- Cambio de estado
- Estructura de espacio de estado
- Solución de problemas
- Descripción del problema; ejemplos de la definición del problema.

Resolución de Problemas Generales

La resolución de problemas es fundamental para la mayoría de las aplicaciones de IA. De hecho, la capacidad de resolver problemas suele usarse como una medida de la inteligencia tanto para el ser humano como para la computadora.

Resolución de Problemas Generales

Hay principalmente dos clases de problemas:

- Una primera clase puede ser resuelta usando algún tipo de procedimiento determinista, cuyo éxito esté garantizado. A este procedimiento se le llama de computación.
- En el mundo real, muy pocos problemas se prestan a soluciones sencillas. La mayoría de los problemas del mundo real sólo pueden resolverse mediante la búsqueda de una solución. AI se ocupa de este tipo de resolución de problemas.

Resolución de Problemas Generales

- **La resolución de problemas es un proceso** de generación de soluciones a partir de datos observados.

- un problema se caracteriza por un conjunto de objetivos,

- un conjunto de objetos, y

- un conjunto de operaciones.

Estos pueden ser imprecisos y pueden evolucionar durante la resolución de problemas.

Resolución de Problemas Generales

- **Espacio del problema** es un espacio abstracto.

- Un espacio del problema abarca todos los ***estados válidos*** que se pueden generar por la aplicación de cualquier combinación de ***los operadores*** en cualquier combinación de ***objetos***.

- El espacio del problema puede contener una o más ***soluciones***.

La solución es una combinación de las operaciones y los objetos que logran las metas.

Resolución de Problemas Generales

- **Buscar** refiere a la búsqueda de una solución en un problema de espacio.

- Buscar procede con diferentes tipos de *estrategias de control de búsqueda.*

- La *búsqueda primero en profundidad y primero en amplitud* son dos estrategias de *búsqueda comunes.*

Resolución de Problemas Generales

1. Resolución de problemas generales

- La resolución de problemas es un proceso de generación de soluciones
- General de Problem Solver (GPS) fue un programa informático creado en el año 1957 por *Simon* y *Newell* para construir una máquina que resuelve problemas universales

Resolución de Problemas Generales

1. Resolución de problemas generales

1.1 Definición del problema:

a) Definir un ***espacio de estado*** que contiene todas las posibles configuraciones de los objetos relevantes.

b) Especifique uno o más estados, que describen las posibles situaciones, desde que el proceso de resolución de problemas puede comenzar. Estos estados se llaman ***estados iniciales.***

Resolución de Problemas Generales

1. Resolución de problemas generales

1.1 Definición del problema:

c) Especifique uno o más estados que serían solución aceptable para el problema. Estos estados se llaman ***estados meta***.

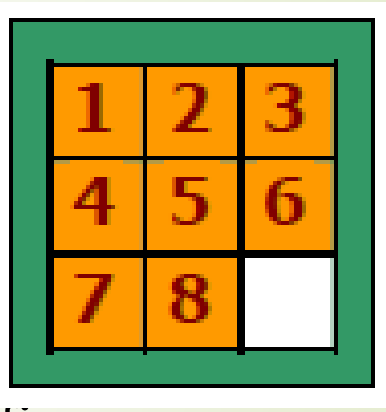
d) Especificar un conjunto de reglas que describen las acciones (operadores) disponible.

Resolución de Problemas Generales

1. Resolución de problemas generales

Ejemplo 1:

Un juego de 8-Puzzle (rompecabezas)



- Espacio de estados.- 8 fichas en el tablero
- Estado inicial.- cualquier configuración
- Estado meta.- fichas en un orden especitico
- Acción.- mover el blanco (left, right, up, down)
- Solución.- secuencia óptima de los operadores

Resolución de Problemas Generales

1. Resolución de problemas generales

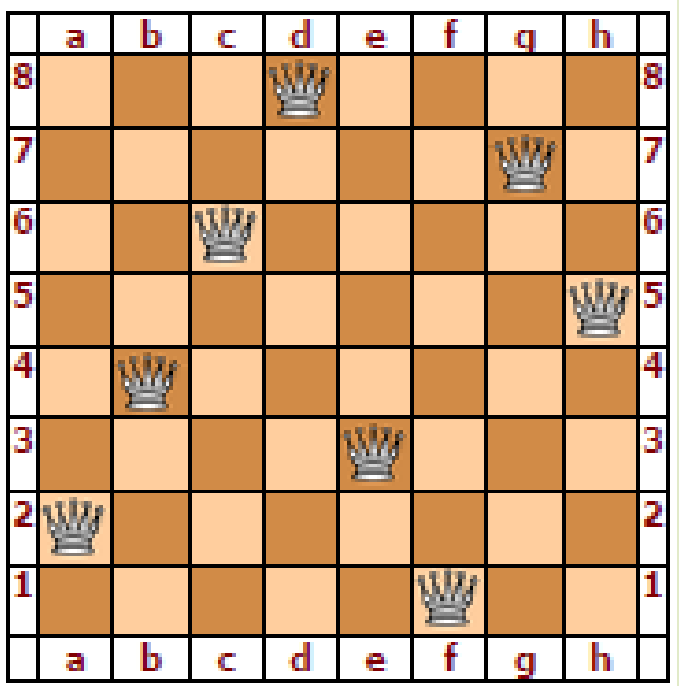
Ejemplo 2:

Un juego de n - reinas rompecabezas, n = 8

- Espacio de estados.-

Las configuraciones de **n = 8** reinas en el tablero con un solo reina por fila y columna

- Estado inicial.- Tablero sin reinas
- Estado meta.- 8 reinas sin que se puedan atacar
- Acción
- Solución



Estrategias de Búsqueda y Control

Una estrategia se define escogiendo el orden en el que los nodos se expanden.

Las estrategias de búsqueda se evalúan en las siguientes dimensiones:

- I*ntegridad*
- *La complejidad de tiempo*
- *La complejidad espacial*
- *Optimalidad.*

Estrategias de Búsqueda y Control

Búsqueda de términos relacionados

- **El rendimiento y la complejidad del algoritmo**
  - Rendimiento de un algoritmo requiere de factores internos y externos

Factores Internos	Factores Externos
Tiempo requerido, para ejecutarse	**Tamaño** de entrada de un algoritmo
Espacio (memoria) requerida para ejecutarse	**Velocidad** de la computadores
	Calidad del compilador

  - Complejidad, mide los factores internos (tiempo)

Estrategias de Búsqueda y Control

- **Complejidad computacional**

  Una medida de los recursos en términos de tiempo y espacio.

- Si **A** es un algoritmo que resuelve un problema de decisión **f** entonces el tiempo de ejecución **A** es el número de pasos dados en la entrada de longitud **n.**
- **Complejidad de Tiempo.- T (n)** de un problema de decisión **f** es el tiempo de ejecución del "Mejor" algoritmo **A** para **f.**
- **Complejidad espacial.- S (n)** de un problema de decisión **f** es la cantidad de memoria utilizada por el "mejor"algoritmo **A** para **f.**

Estrategias de Búsqueda y Control

- **Notación “O” mayúscula (“Big-O”)**

Es la medida teórica de la ejecución de un algoritmo, generalmente indica el ***tiempo*** o la ***memoria*** necesaria, dado el problema de tamaño **n,** que es por lo general el número de elementos.

- **Notación Big-O**

se utiliza para dar una aproximación al tiempo de ejecución-la eficiencia de un algoritmo, la letra **"O"** es para orden de magnitud de operaciones o el espacio en tiempo de ejecución.

Estrategias de Búsqueda y Control

- **El Big-O de un algoritmo A**

  - Si un algoritmo **A** requiere un tiempo proporcional a **f(n),** entonces el algoritmo **A** se dice que es de orden

  f (n), Y que se denota como **O (f (n)).**

- Si el algoritmo **A** requiere un tiempo proporcional a

  N^2 , Entonces el orden del algoritmo se dice que es

  $O(n^2)$.

- Si el algoritmo **A** requiere un tiempo proporcional a **n,** entonces el orden del algoritmo se dice que es

  O (n)

Estrategias de Búsqueda y Control

La función **f (n)** se llama **la función de la tasa de crecimiento de un algoritmo**

- Si un algoritmo tiene complejidad rendimiento **O (n)**, esto significa que el tiempo de ejecución **t** debe ser directamente proporcional a **n,** es decir, **t ∞ n** o **t = kn**

  donde **k** es la constante de proporcionalidad.

- Del mismo modo, para los algoritmos de rendimiento que tiene complejidad **O (log_2(n))**, **O (log N)**,

  O(N log N), **O (2^N)** y así sucesivamente.

Estrategias de Búsqueda y Control

Ejemplo 1

Array. Determinar el Big-O de un algoritmo;

Calcular la suma de los **n** elementos en una matriz de enteros a**[0. . . . N-1]**

Line no	Instructions	No of execution steps
line 1	sum = 0	1
line 2	for (i = 0; i < n; i++)	n + 1
line 3	sum += a[i]	n
line 4	print sum	1
	Total	2n + 3

Estrategias de Búsqueda y Control

Para el polinomio **(2 * n + 3)** el Big-O está dominado por el primer termino **N,** mientras que el número de elementos de la matriz se hace muy grande. También en la determinación de la **Big-O**

- Haciendo caso omiso de las constantes **2** y **3**, el algoritmo es de orden **n.**
- Así que el **Big-O** del algoritmo es **O (n).**
- En otras palabras, el tiempo de ejecución de este algoritmo aumenta más o menos como el tamaño de los datos de entrada **n,** por ejemplo una matriz de tamaño **n.**

Estrategias de Búsqueda y Control

Ejemplo 2

Array a**[0. . . . . n-1] [0. . . . . N-1]**, Encontrar el elemento más grande.

Line no	Instructions	No of executions
line 1	max = a[0][0]	1
line 2	for (row = 0; row < n; row++)	n + 1
line 3	for (col = 0; col < n; col++)	n * (n+1)
line 4	if (a[row][col] > max) max = a[row][col].	n * (n)
line 5	print max	1
	Total	$2n^2 + 2n + 3$

Estrategias de Búsqueda y Control

- Haciendo caso omiso de las constantes **2**, **2** y **3**, el algoritmo es de orden **n^2.**
- Así que el **Big-O** del algoritmo es **O (n^2).**
- En otras palabras, el tiempo de ejecución de este algoritmo aumenta más o menos como el tamaño de los datos de entrada **n^2,** por ejemplo una matriz de tamaño **n x n.**

Estrategias de Búsqueda y Control

Ejemplo 3

Polinomio en n con grado k

- Número de pasos necesarios para llevar a cabo un algoritmo expresado como

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n^1 + a_0$$

Entonces **f (n)** es un polinomio en **n** con grado **k** y **f (n)** ∈**O(nk).**

- Para obtener el orden de una función polinomial, se usa término que es de grado más alto y no tener en cuenta las constantes y los términos que son de grados inferiores.

  El Big-O del algoritmo es **O (nk).**

- En otras palabras, en tiempo de ejecución de este algoritmo aumenta exponencialmente.

Estrategias de Búsqueda y Control

Variación de la tasa de crecimiento con número de entrada y tiempo

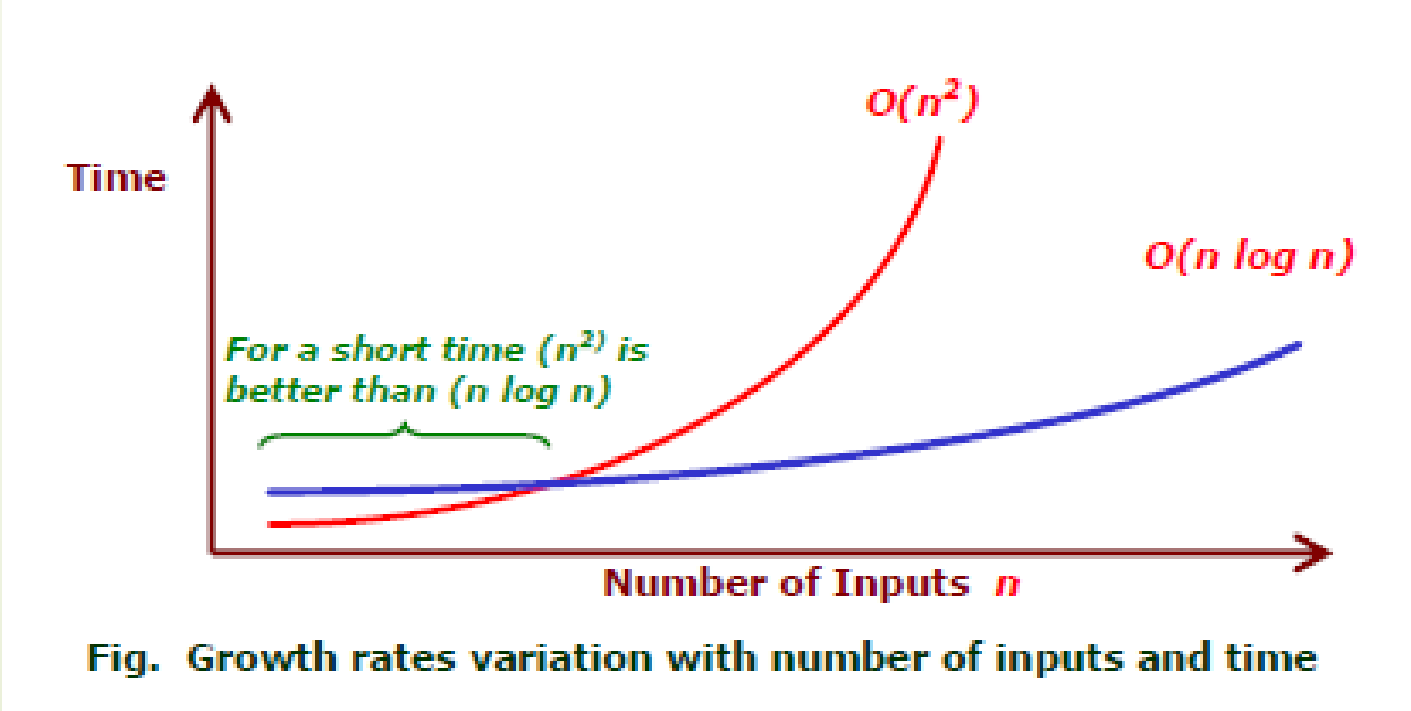


Fig. Growth rates variation with number of inputs and time

Estrategias de Búsqueda y Control

- **Estructura de Árbol**

El árbol es una forma de organizar los objetos, relacionados de manera jerárquica.

Árbol es un tipo de estructura de datos donde:

- Cada ***elementos*** está unido a uno o más elementos directamente debajo de ella.
- Las conexiones entre elementos se denominan ***ramas***.
- árbol se llama a menudo ***árboles invertidos*** porque su *raíz* está en la parte superior.

Estrategias de Búsqueda y Control

- Los elementos que no tienen elementos por debajo de ellos se llaman ***hojas***.
- Un ***árbol binario*** es un tipo especial, cada elemento tiene dos ramas por debajo de ella.

Ejemplo:

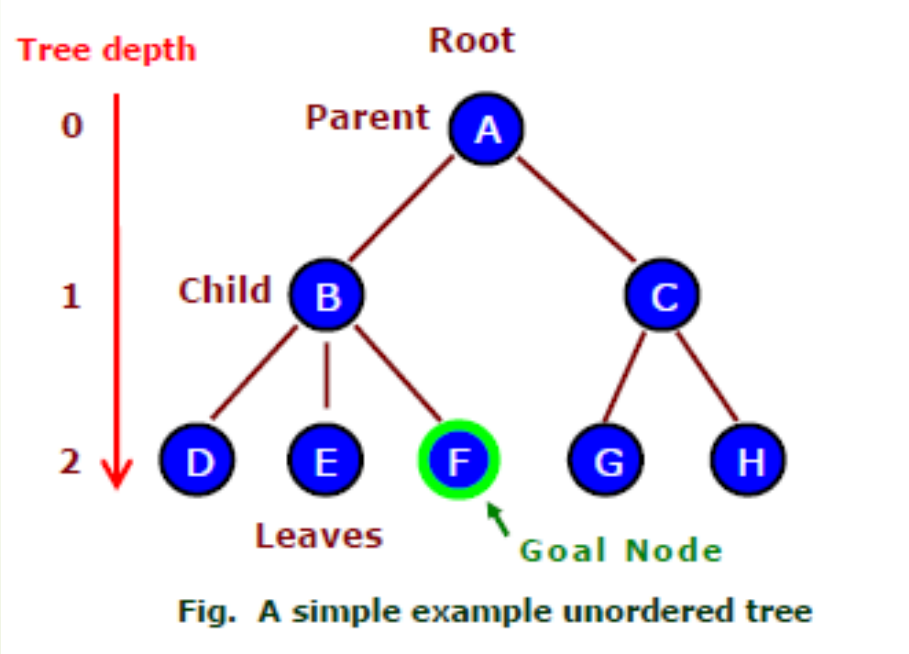
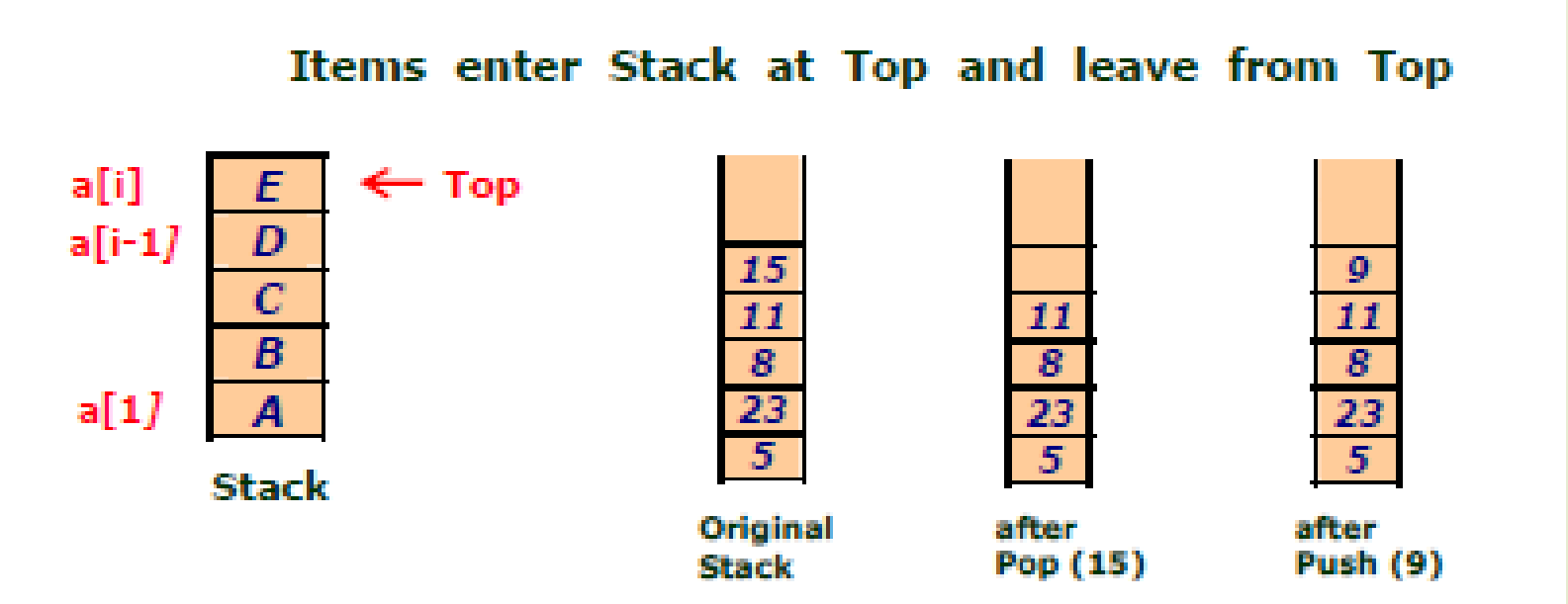


Fig. A simple example unordered tree

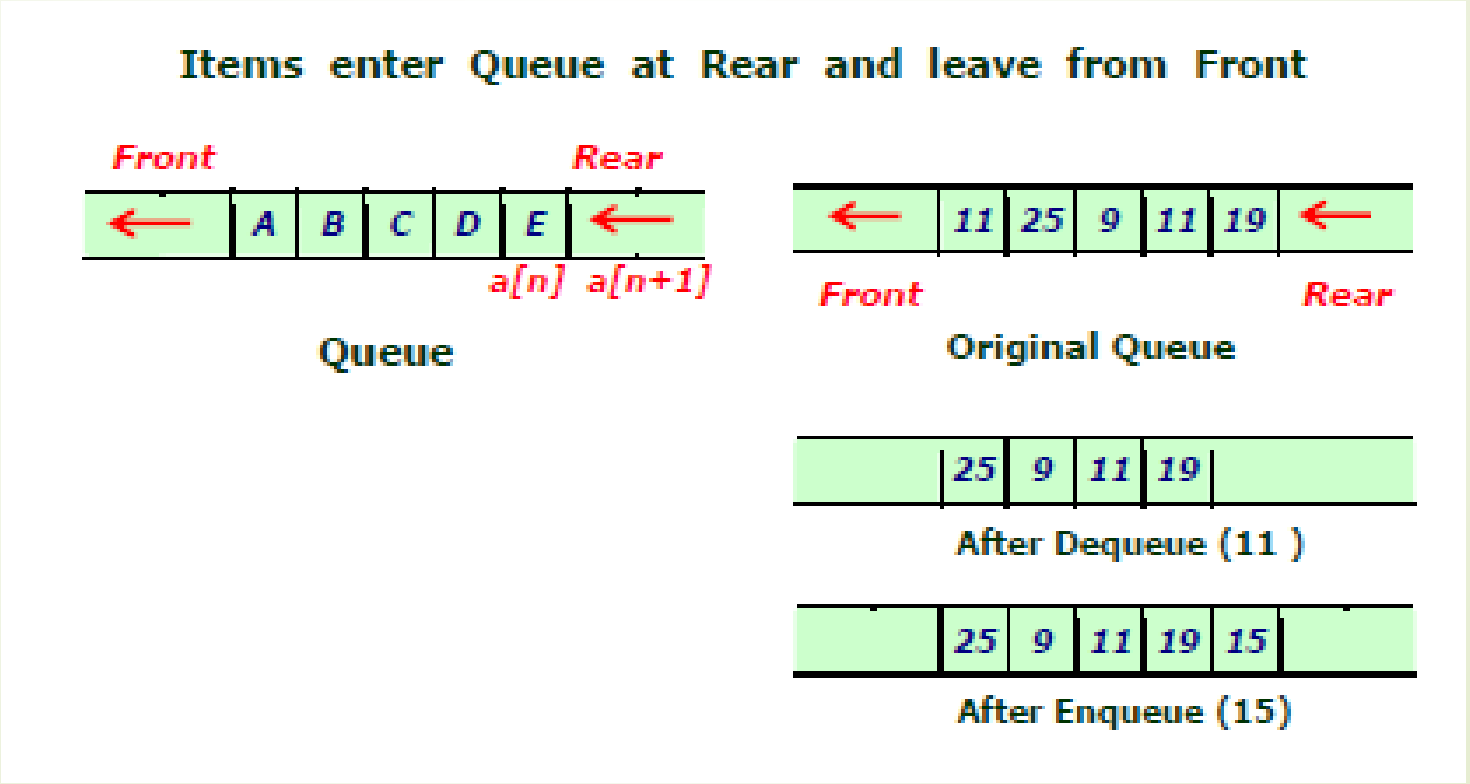
Estrategias de Búsqueda y Control

- **Pilas y Colas (Stacks and Queues)**
- **Pila.- Lista ordenada que trabaja como Last-In First-Out (LIFO)**



Estrategias de Búsqueda y Control

- **Colas.- Lista ordenada que trabaja como primero en entrar, primero en salir (FIFO)**



Estrategias de Búsqueda y Control

- **Algoritmos de búsqueda:**

Muchos algoritmos de búsqueda tradicionales se utilizan en aplicaciones de IA. Para problemas complejos, los algoritmos tradicionales no son capaces de encontrar la solución dentro de un cierto limite de tiempo y espacio. En consecuencia, muchas técnicas especiales se desarrollan, utilizando **funciones heurísticas**

- Los algoritmos que utilizan funciones heurísticas se llaman **algoritmos heurísticos.**
- Los algoritmos heurísticos *no* son *realmente inteligentes,* sino que parecen ser inteligente porque lograr un mejor rendimiento.

Estrategias de Búsqueda y Control

- **Algoritmos de búsqueda:**

- Los algoritmos heurísticos son *más eficientes,* ya que se aprovechan de retroalimentación de los datos para dirigir la ruta de búsqueda.

- Algoritmos de búsqueda **desinformados** o **algoritmos de búsqueda fuerza bruta**, a través del espacio de búsqueda, consiste en enumerar sistemáticamente todos los posibles candidatos para la solución de un problema, con el fin de verificar si dicho candidato satisface la solución al mismo.

Estrategias de Búsqueda y Control

- **Algoritmos de búsqueda:**
  - Algoritmos de **búsqueda informada** usan funciones heurísticas, que son específicos para el problema, los aplican para orientar la búsqueda a través del espacio de búsqueda para tratar de reducir la cantidad de tiempo invertido en la búsqueda.

Una buena heurística puede hacer una búsqueda informada drásticamente superar el rendimiento de cualquier Búsqueda desinformada. Por ejemplo, el problema del viajero (TSP) donde el objetivo es encontrar una ***buena solución*** en lugar de encontrar la ***mejor solución***.

Estrategias de Búsqueda y Control

- **Algoritmos de búsqueda:**

Algunos de los algoritmos de búsqueda inteligentes destacados se presentan a continuación.

1. **Búsqueda de Generar y prueba**
2. **Búsqueda primero el mejor**
3. **Búsqueda Greedy**
4. **Búsqueda A ***
5. **Búsqueda de restricciones**
6. **Medios y fines de análisis**

Representación jerárquica de Algoritmos de búsqueda

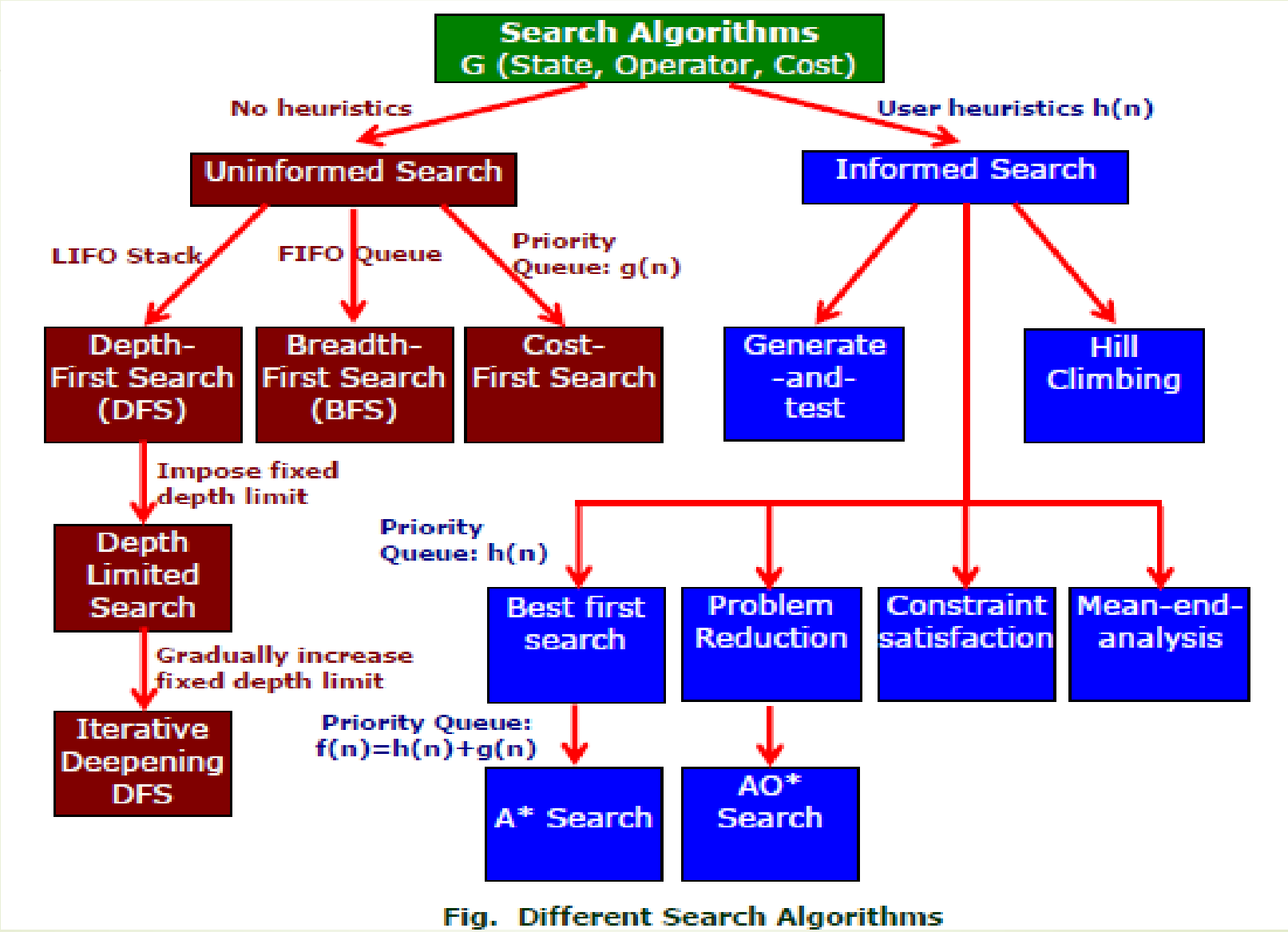


Fig. Different Search Algorithms

Estrategias de Búsqueda y Control

- Espacio de búsqueda

Un conjunto de todos los estados, que se puede llegar, constituye un espacio de *búsqueda*.

Esto se obtiene mediante la aplicación de una combinación de los operadores para definir su conectividad.

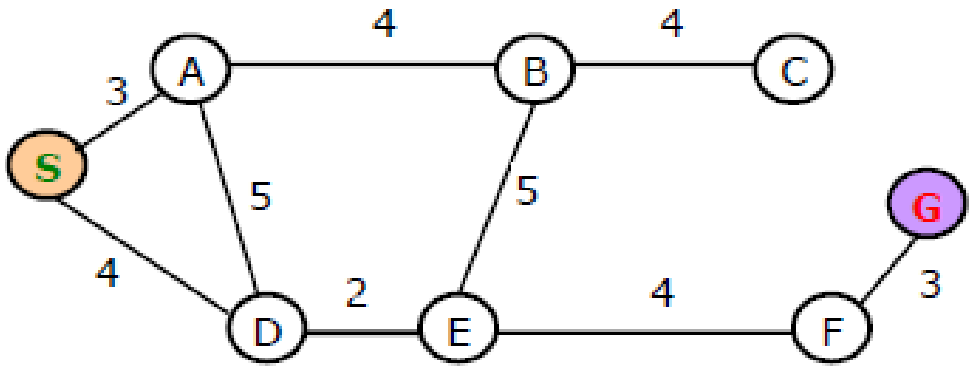


Fig. Search Space

Estrategias de Búsqueda y Control

- **Estrategias de control**
  - ✓ **Búsqueda hacia adelante**
  - ✓ **Búsqueda hacia atrás**
  - ✓ **Búsqueda hacia adelante y hacia atrás**
  - ✓ **Búsqueda sistemática**
  - ✓ **Búsqueda Heurística**

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda hacia adelante**

En este sentido, las estrategias de control para la exploración de búsqueda procede hacia adelante de un estado inicial a una solución; los métodos se denominan ***datos dirigida*** .

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda hacia atrás**

En este sentido, las estrategias de control para la exploración de búsqueda procede hacia atrás desde una meta o estado final hacia uno u otro un problema sub solucionable o el estado inicial; los métodos se llaman ***meta dirigida***.

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda hacia adelante y hacia atrás**

Aquí, las estrategias de control para explorar la búsqueda es una *mezcla de las estrategias* hacia adelante y hacia atrás.

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda sistemática**

Cuando el espacio de búsqueda es pequeña, de forma sistemática (pero método ciego) se puede utilizar para explorar todo el espacio de búsqueda.

Uno de estos métodos de búsqueda es ***primero en profundidad*** y el otro es ***primero en amplitud***.

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda Heurística**
  - Muchas búsqueda depende del conocimiento del dominio del problema.
  - La búsqueda guiada se llaman *búsqueda heurística* y los métodos utilizados se denominan *heurística* .

Nota: Una búsqueda heurística no siempre puede encontrar la mejor solución, pero está garantizado para encontrar una buena solución en un plazo razonable.

Estrategias de Búsqueda y Control

- **Estrategias de control**
- ✓ **Búsqueda Heurística**

Algoritmos de búsqueda Heurística:

- En primer lugar, generar una posible solución que puede ser o bien un punto en el espacio del problema o de un camino desde el estado inicial.

- A continuación, compruebe si esta posible solución es una solución real comparando el estado alcanzado con el conjunto de estados de la meta.

- Por último, si se trata de una verdadera solución, volver, otra repetición de la primera vez.

3. Búsquedas exhaustivas

- Una búsqueda se dice ser **exhaustiva** si se garantiza la búsqueda sobre todos los estados alcanzables (resultados) antes de terminar con el fracaso.

- Una representación gráfica de todos los posibles estados alcanzables y las rutas por donde pueden ser visitados es llamado un **árbol de decisión**.

Breadth – First Search (BFS)

- La búsqueda **primero en amplitud** es una estrategia sencilla en la que se expanden todos los nodos a una profundidad en árbol de búsqueda antes de expandir cualquier nodo del próximo nivel.
- Se implementa con un algoritmo **FIFO(cola**), asegurando que los primeros nodos visitados serán los primeros expandidos.

Breadth – First Search (BFS)

- Los requisitos de **memoria** representan un problema más grande para la búsqueda primero en amplitud que el tiempo de **ejecución**.
- Los problemas de búsqueda de **complejidad exponencial** no pueden resolverse por métodos sin información, salvo casos pequeños.
- Esta estrategia a menudo no es factible cuando el **espacio** de búsqueda es grande.

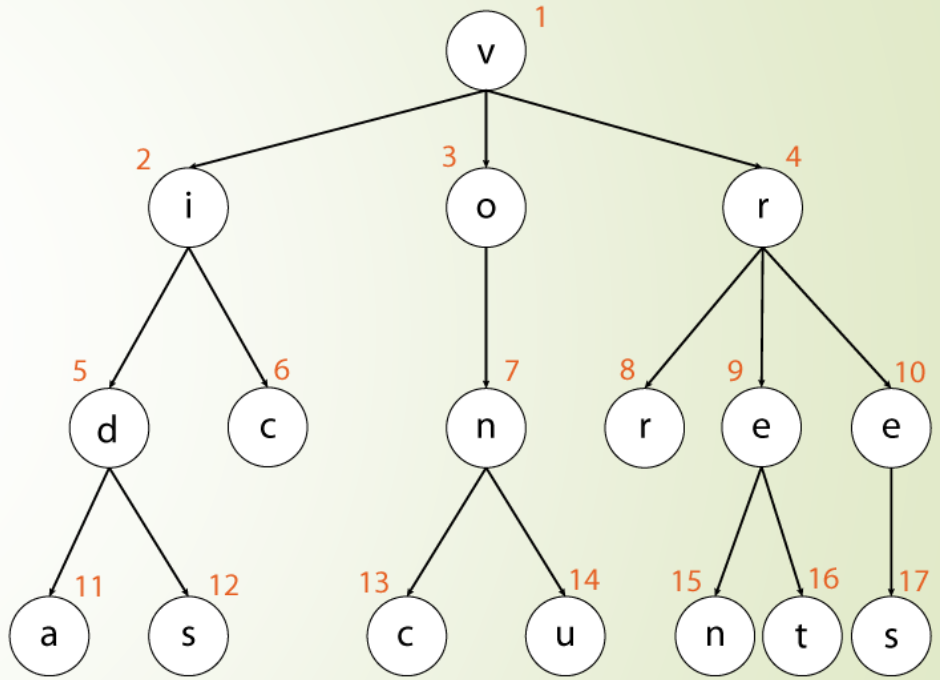
Breadth – First Search (BFS)

- La búsqueda genera todos los nodos de un particular nivel antes de proceder con el segundo nivel del árbol.
- La búsqueda prueba sistemáticamente cada nodo que es alcanzable desde el nodo padre antes de expandir cualquiera de los nodos hijos de estos nodos.
- El régimen de control garantiza que el espacio de movimientos posibles es sistemáticamente examinado, esta búsqueda requiere recursos de memoria considerables.
- El espacio de búsqueda es bastante grande y la solución puede estar a miles de pasos del nodo de inicio.
- La búsqueda termina cuando la solución es encontrada y el algoritmo regresa true.

Breadth – First Search (BFS)

- ALGORITMO:

Colocar el nodo raíz en la COLA
MIENTRAS (COLA no este vacía) {
    Sacamos el primer nodo de la COLA
    SI (el nodo es un nodo meta)
        ENTONCES regresa satisfactorio;
    SI NO
        Poner todos los hijos del nodo en la COLA;
} regresa falla;



Si objetivo = u

v	i	o	r	d	c	n	r	e	e	a	s	c	u

Breadth – First Search (BFS)

Ventajas de la búsqueda en anchura:

- Es **completo**: siempre encuentra la solución si existe.
- Es **óptimo** si el coste de cada rama es constante: en Inteligencia Artificial puede que cada nodo sea un estado de un problema, y que unas ramas tengan un coste diferente a las demás.

Inconvenientes de la búsqueda en anchura:

- **Complejidad** exponencial en espacio y tiempo (incluso peor la del espacio que la del tiempo).

Depth – First Search (DFS)

- La búsqueda **primero en profundidad** siempre expande el nodo más profundo del árbol de búsqueda.

- Se implementa con un algoritmo LIFO(pila), en dónde se aplica la búsqueda primero en profundidad con una función recursiva que se llama en cada uno de sus hijos.

Depth – First Search (DFS)

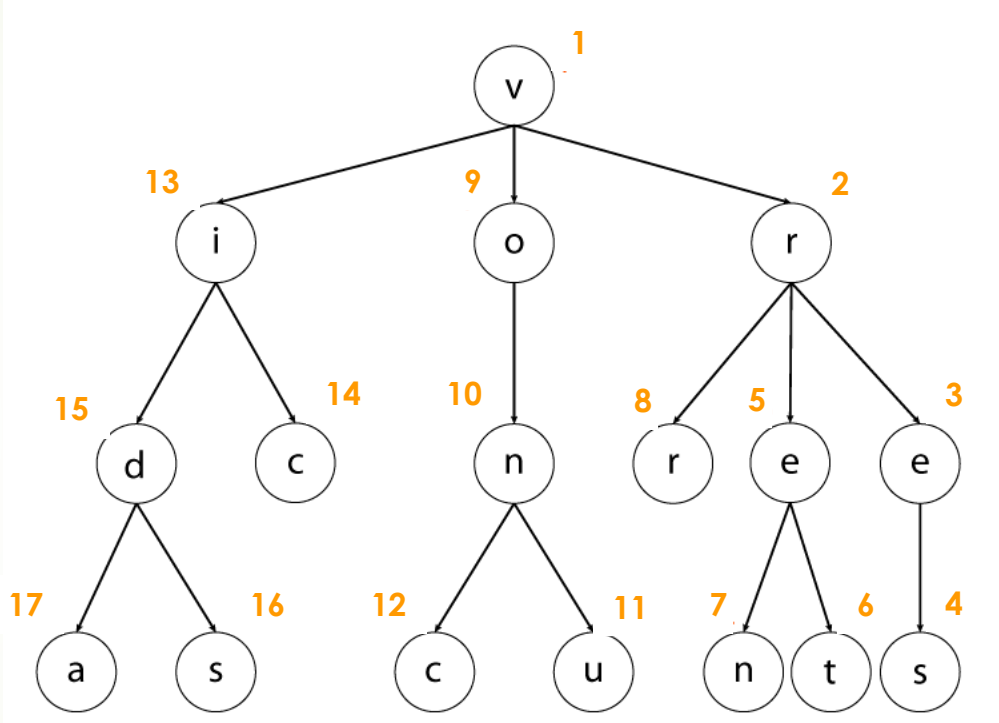
- La búsqueda procede sistemáticamente hacia una profundidad **d**, antes de que otro camino sea considerado.

- Si la máxima profundidad del árbol es tres, tenemos que: si este limite se alcanzado y la solución no ha sido encontrada, entonces la búsqueda regresa al nivel anterior (**backtraking**) y explora cualquier alternativa restante de este nivel y así sucesivamente.

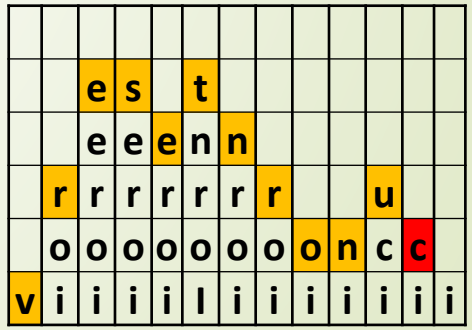
Depth – First Search (DFS)

- ALGORITMO:

Colocar el nodo raíz en la PILA
MIENTRAS (PILA no este vacía) {
    Sacamos el nodo cabeza de la PILA
    SI (el nodo es un nodo meta)
        ENTONCES regresa satisfactorio;
    SI NO
        Poner todos los hijos del nodo en la PILA
} regresa falla;



Si objetivo = c



Depth – First Search (DFS)

Ventajas de la búsqueda en profundidad:

- Es **completo**: si no existen ciclos repetidos.
- Tiene menor **complejidad** en espacio que la búsqueda en anchura, porque solo mantenemos en memoria un camino simultáneamente.

Inconvenientes de la búsqueda en profundidad:

- **No** es **óptima**.
- Puede **no** encontrar la solución, aunque exista, si hay caminos infinitos. Por esta razón **no** es completa.

Comparación DFS y BFS

Depth - First Search	Breadth– First Search
Colocar el nodo raíz en la **PILA** MIENTRAS (**PILA** no este vacía) { Sacamos el nodo cabeza de la **PILA** SI (el nodo es un nodo meta) ENTONCES regresa satisfactorio; SI NO Poner todos los hijos del nodo en la **PILA** } regresa falla;	Colocar el nodo raíz en la **COLA** MIENTRAS (**COLA** no este vacía) { Sacamos el primer nodo de la **COLA** SI (el nodo es un nodo meta) ENTONCES regresa satisfactorio; SI NO Poner todos los hijos del nodo en la **COLA**; } regresa falla;
Un árbol grande, podría tomar excesivo tiempo de **ejecución** para encontrar un nodo meta.	Un árbol grande, podría requerir excesivo uso de **memoria** para encontrar un nodo meta.
Cuando tiene éxito, el nodo meta encontrado **no necesariamente** es la profundidad mínima.	Cuando tiene éxito, encuentra una **profundidad mínima** (la más cercana a la raíz) nodo objetivo.

… búsquedas exhaustivas

En realidad **no hay una búsqueda no informada mejor o peor**, porque dependen mucho de la posición de nuestra *solución* dentro del árbol (entendemos *solución* por *lo que estamos buscando*).

¿Cómo solucionamos esto?

La respuesta es usar búsquedas **informadas** (usando **heurísticas**).

4. Técnicas de búsqueda heurística

Para problemas complejos, los algoritmos tradicionales, presentados anteriormente, no son capaces de encontrar la solución dentro de ciertos límites de tiempo y espacio. En consecuencia, muchas de las técnicas son desarrolladas utilizando las **funciones heurísticas**.

- La Búsqueda Heurística es la que utiliza el conocimiento específico del problema más allá de la definición del problema en sí mismo.
- La Búsqueda Heurística es considerada una de las técnicas débiles, pero puede ser efectiva si se aplica correctamente; esta técnica requiere información específica del dominio.

Características de búsqueda heurística

- Las Heurísticas es conocimiento acerca del dominio, él cual ayuda a buscar y razonar en su propio dominio.

- La Búsqueda Heurística incorpora conocimiento del dominio para mejorar la eficiencia, por ejemplo, en la Búsqueda Blind (ciega).

- La Heurística es una función que, cuando es aplicada a un estado, regresa un valor como mérito estimado de estado, con respecto a la meta.

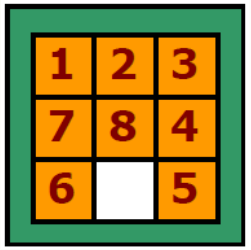
- Las Funciones Heurísticas son la forma más común de transmitir el conocimiento adicional del problema al algoritmo de búsqueda.

Búsqueda Heurística comparada con otras búsquedas.

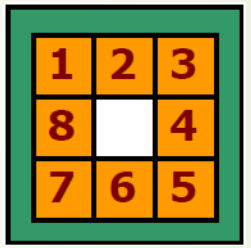
Brute force / Blind search	Heuristic search
Solo tiene conocimiento acerca de los nodos ya explorados.	Estima la "distancia" al estado meta.
No tiene conocimiento de que tan lejos esta un nodo del estado meta.	Guía los procesos de búsqueda hacia el estado meta.
	Prefiere estados (nodos) que vayan cerca y no lejos del estado meta.

Ejemplo: 8 - Puzzle

- **Espacio de estados**: Configuración de 8 cuadros sobre el tablero.
- **Estado inicial**:



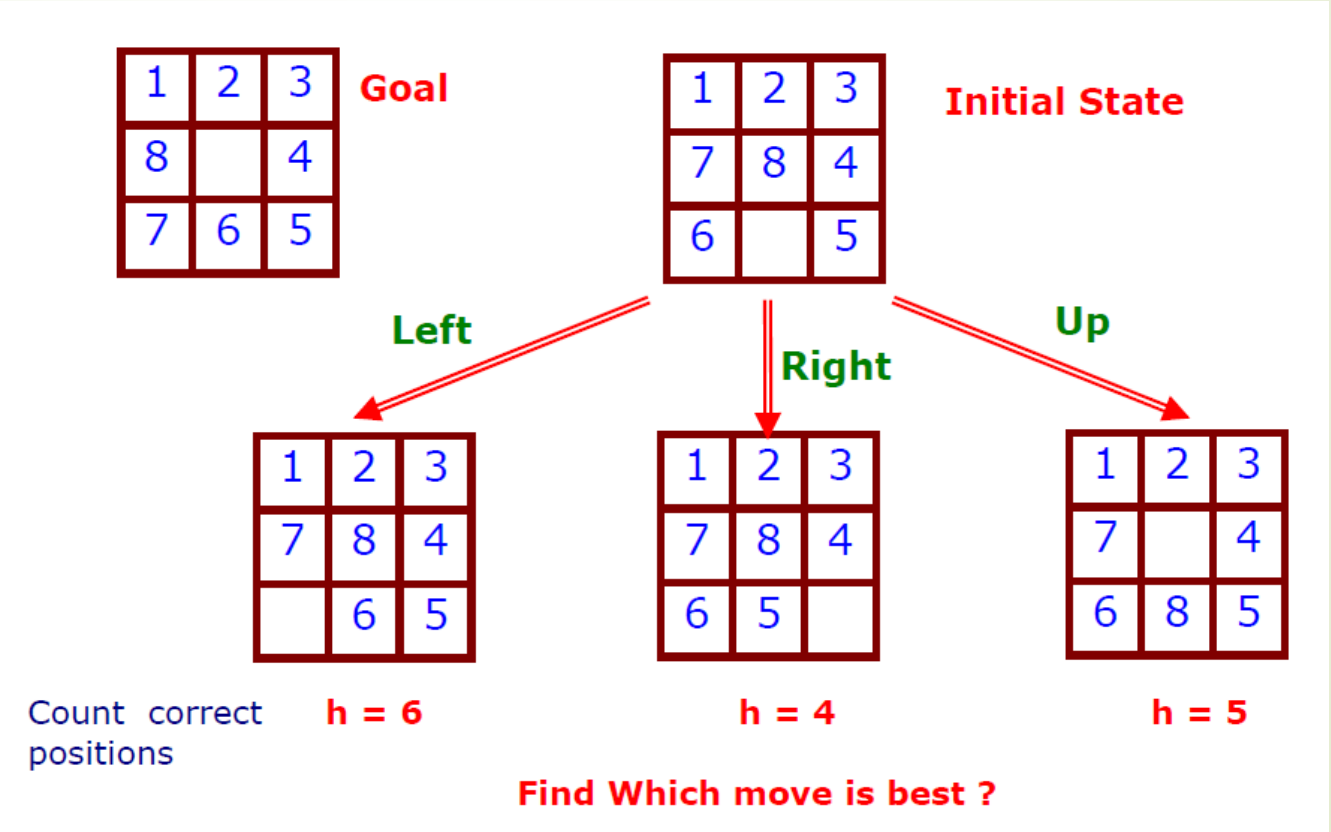
Estado meta:



- **Solución**: Secuencia optima de operadores.
- **Acción**: Mover espacio en blanco
  - Condición: El movimiento es dentro del tablero
  - Direcciones: Left, Right, Up Down
- **Problema**:
  - ¿ Qué movimiento es el mejor ?
  - ¿ Qué heurísticas pueden decidir?

… continuando con el ejemplo del 8 - Puzzle:

Tres movimientos posibles:



… continuando con el ejemplo del 8 - Puzzle:

Aplicar la heurística: Se consideran tres diferentes enfoques

- Contar la posición **correcta** de cada cuadro, comparado al estado meta.
- Contar la posición **incorrecta** de cada cuadro, comparado al estado meta.
- Contar que tan lejos esta cada cuadro, de su posición correcta.
- Enfoques:

Enfoques	Left	Right	Up
1. Contar posiciones **correctas.**	6	4	5
2. Contar posiciones **incorrectas.**	2	4	3
3. Contar que tan **lejos** estan.	2	4	4

Tipos de algoritmos de búsqueda heurística:

- Generate - And - Test
- Hill climbing
  - Simple
  - Steepest - Ascent Hill climbing
  - Simulated Anealing
- Best First Search
  - OR Graph
  - A* (A - Star) Algorithm
  - Agendas
- Problem Reduction
  - AND – OR_Graph
  - AO* (AO – Start) Algorithm
- **Constraint Satisfaction**
- Mean – end Analysis

5. Problemas y modelos de satisfacción de restricciones: características

- Las restricciones surgen en la mayoría de las áreas de la actividad humana.
- Las restricciones son un medio natural para que la gente exprese problemas en muchos campos.
- Muchos problemas reales en Inteligencia Artificial pueden ser modelados como Problemas de Satisfacción de Restricciones (CSPs) y son resueltos a través de búsquedas.

Ejemplo de restricciones:

- La suma de los tres ángulos de un triángulo es 180 grados.

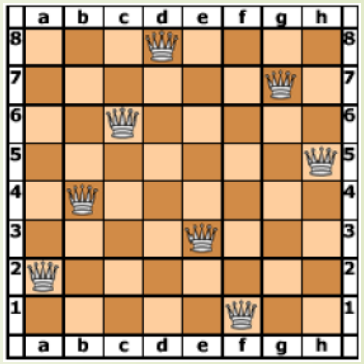
Problemas de Satisfacción de Restricciones

- La restricción es una relación lógica entre variables.
- La satisfacción de restricciones es un proceso de encontrar una solución a un conjunto de restricciones.
- Encontrar una solución es evaluando estas variables que satisfacen todas las restricciones.

Ejemplos de Problemas de Satisfacción de Restricciones

- **Latin Square**: ¿Cómo podemos llenar un tablero **n x n** con **n** diferentes símbolos tal que cada símbolo aparece solo una vez en cada renglón y en cada columna?

- **El problema de las ocho reinas:** ¿Cómo puedes poner 8 reinas en un tablero de ajedrez de 8 X 8, de tal forma que ninguna reina pueda atacar a ninguna otra?



- **Sudoku Problem**: ¿Cómo puedes llenar unas celdas de 9 X 9 de tal manera que cada renglón, columna y cada uno de las cajas de 3X3 contengan los números del 1 al 9 ?

	2	6				8	1	
3			7		8			6
4				5				7
	5		1		7		9	
		3	9		5	1		
	4		3		2		5	
1				3				2
5			2		4			9
	3	8				4	6	

7	**2**	**6**	4	9	3	**8**	**1**	5
3	1	5	**7**	2	**8**	9	4	**6**
4	8	9	6	**5**	1	2	3	**7**
8	**5**	2	**1**	4	**7**	6	**9**	3
6	7	**3**	**9**	8	**5**	**1**	2	4
9	**4**	1	**3**	6	**2**	7	**5**	8
1	9	4	8	**3**	6	5	7	**2**
5	6	7	**2**	1	**4**	3	8	**9**
2	**3**	**8**	5	7	9	**4**	**6**	1

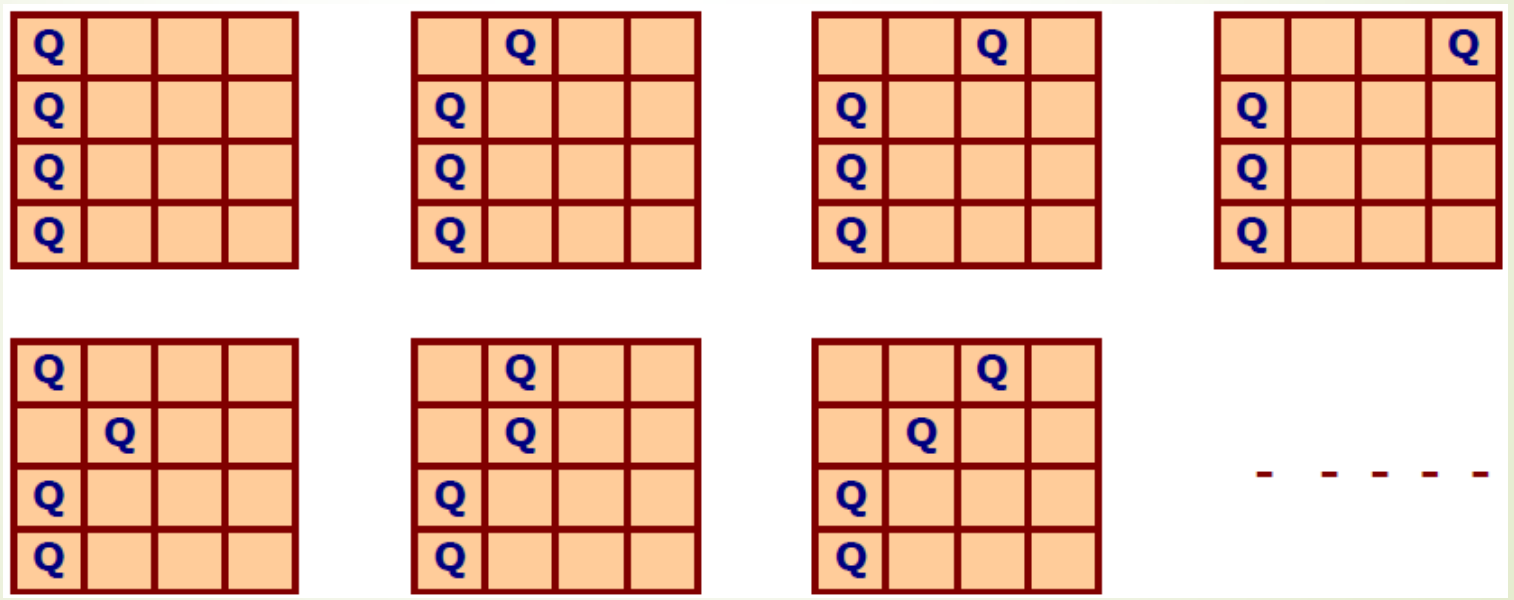
Modelos de Satisfacción de Restricciones

- *Tres enfoques o modelos basados en computadora se muestran a continuación:*
  - Generate and Test (GT)
  - Backtracking (BT)
  - Constraint Satisfaction Problems (CSPs)

Generate and Test (GT)

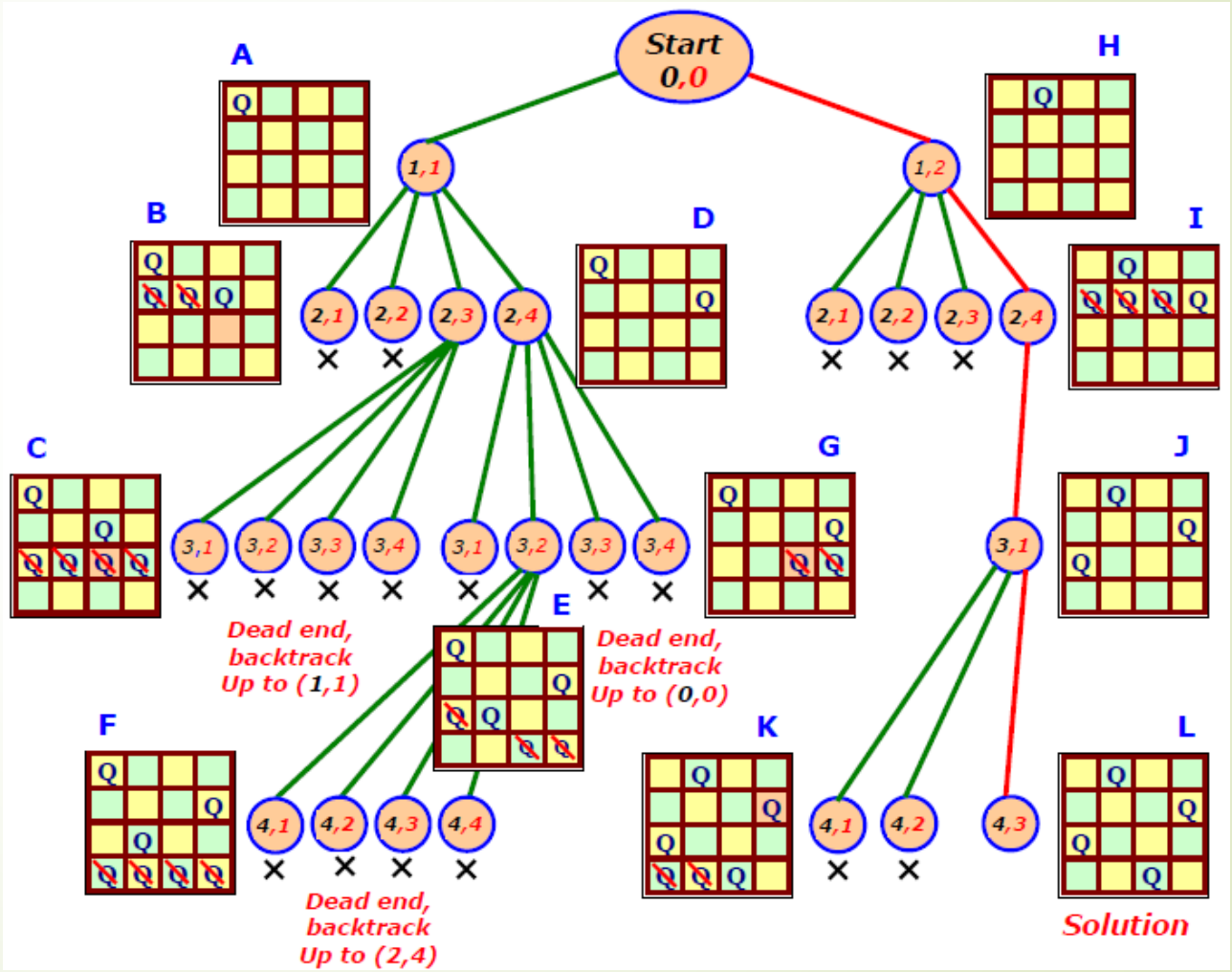
- Una posible solución es tratar sistemáticamente cada posición de la reina hasta que encontremos una solución.

Este proceso es conocido como “Generate and Test”.



Backtracking (BT)

- Este método esta basado en la examinar sistemáticamente las posibles soluciones.
- Regularmente se utiliza una función recursiva.



Referencias

"Artificial Intelligence", by Elaine Rich and Kevin Knight, (2006), McGraw Hill companies Inc., Chapter 1-22, page 1-613.

"Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, (2002), Prentice Hall, Chapter 1-27, page 1-1057.

"Computational Intelligence: A Logical Approach", by David Poole, Alan Mackworth, and Randy Goebel, (1998), Oxford University Press, Chapter 1-12, page 1-608.

"Artificial Intelligence: Structures and Strategies for Complex Problem Solving", by George F. Luger, (2002), Addison-Wesley, Chapter 1- 16, page 1-743.

"AI: A New Synthesis", by Nils J. Nilsson, (1998), Morgan Kaufmann Inc., Chapter 1-25, Page 1-493.

"Artificial Intelligence: Theory and Practice", by Thomas Dean, (1994), Addison-Wesley, Chapter 1-10, Page 1-650.

Related documents from open source, mainly internet. An exhaustive list is being prepared for inclusion at a later date.

Guión explicativo

- Esta presentación tiene como fin lo siguiente:
  - Resolución de Problemas Generales
  - Estrategias de Búsqueda y Control
  - Búsquedas Exhaustivas
  - Técnicas de Búsqueda Heurística
  - Los Problemas de Restricción de Satisfacción

Guión explicativo

- El contenido de esta presentación contiene temas de interés contenidos en la Unidad de Aprendizaje Inteligencia Artificial.

- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 24 horas, además de realizar preguntas a la clase sobre el contenido mostrado.