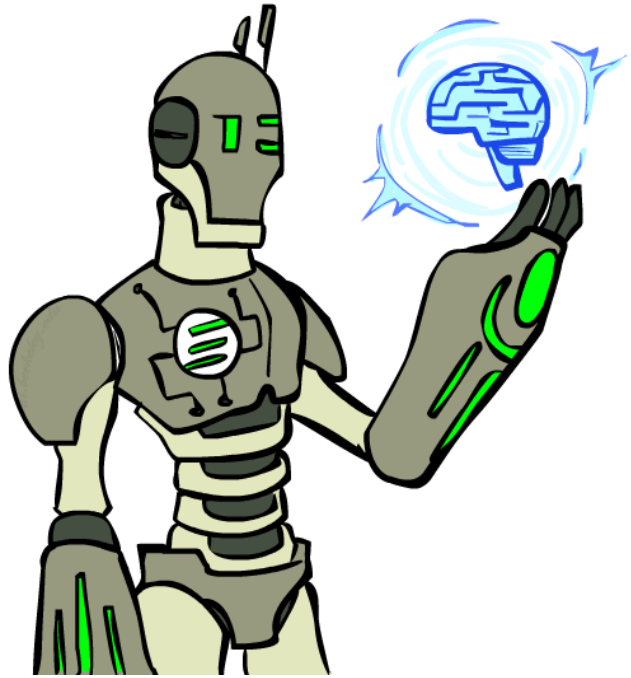
Curso de Inteligencia Artificial CC421
Clase 2

Agentes en la Inteligencia Artificial

Objetivo de la Unidad de aprendizaje

I. Aprender a representar el conocimiento y definir un problema en términos de espacio de estados y resolverlo

II. Aplicar diferentes técnicas de búsqueda



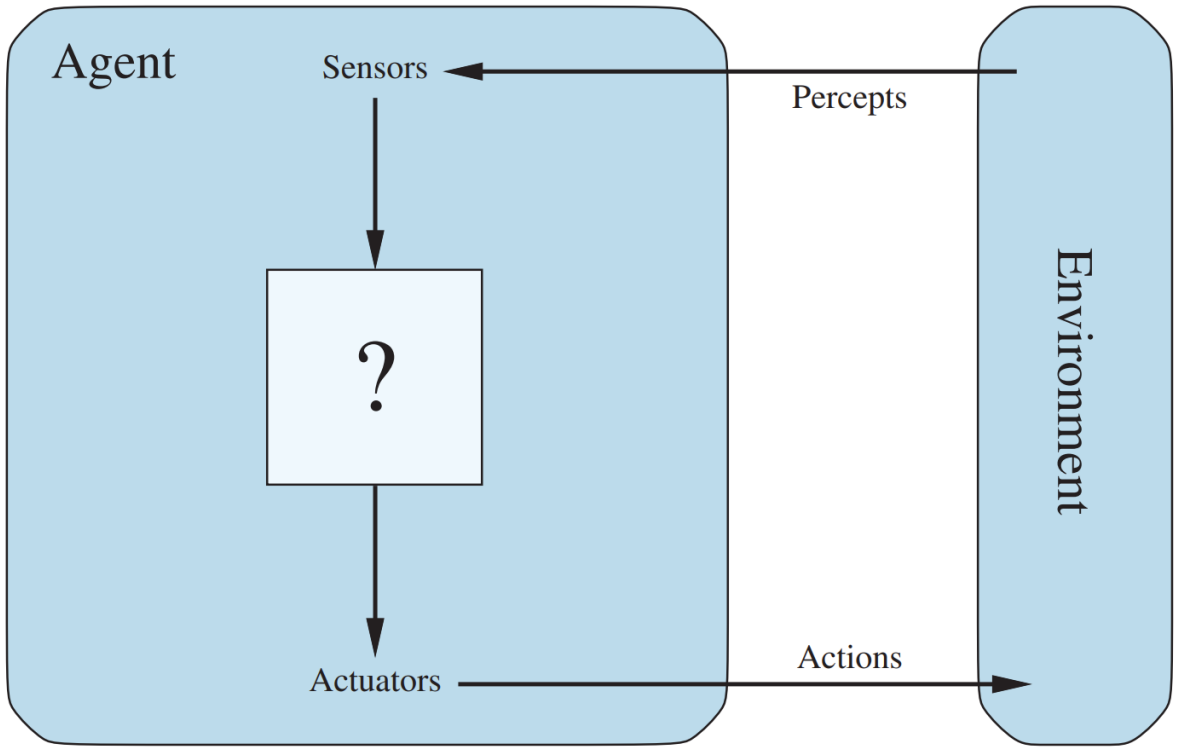
Clasificación de definiciones de IA

	Basado en humanos	Racionalidad Ideal
Basado en razonamiento	**Sistemas que piensan como humanos**	**Sistemas que piensan racionalmente**
Basado en comportamiento	**Sistemas que actúan como humanos**	**Sistemas que actúan racionalmente**

1 [Russell y Norvig, *"Inteligencia Artificial: Un enfoque moderno"*]

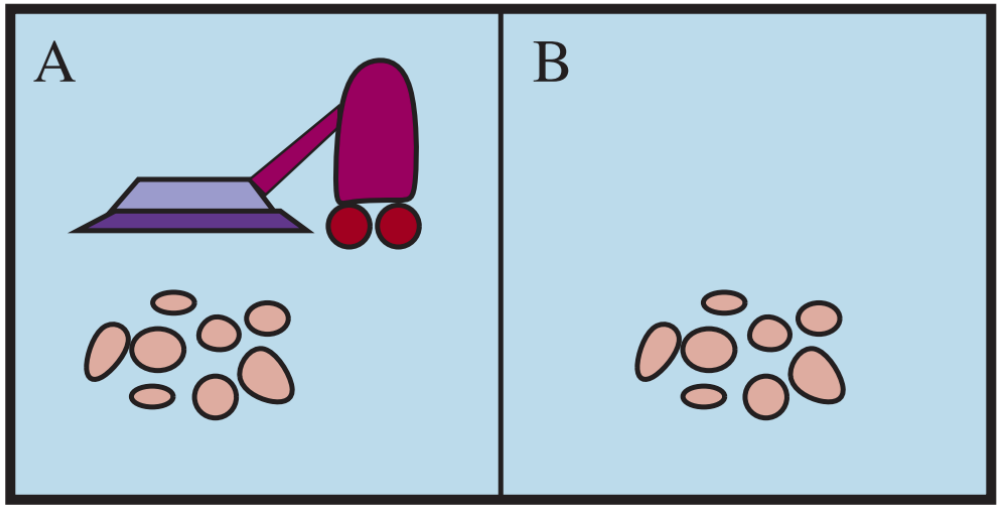
¿Qué es un agente?

Sistemas que perciben su entorno mediante sensores, procesan sus percepciones y responden o actúan en su entorno de manera racional usando actuadores para lograr sus objetivos. Un agente puede ser una entidad física o virtual.



¿Qué es un agente?

Ejemplo: El mundo de la aspiradora.



- ¿Quién es el agente?
  - La aspiradora
- ¿Cuál es el entorno?
  - Las dos cuadriculas A y B
- ¿Qué información toma de su entorno?
  - La aspiradora puede percibir en que cuadrante esta y si hay suciedad en él.
- ¿Qué acción o acciones ejecuta?
  - Puede elegir moverse a la derecha, izquierda, aspirar la suciedad o no hacer nada

¿Qué es un agente?

Una posible lista de acciones a la percepción de la aspiradora es la siguiente

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

Estructura de un agente

Agente = Arquitectura + programa

Esta conformado por el computador/microprocesador, los sensores físicos y los actuadores, en otras palabras " El hardware que lo compone"

El trabajo de la IA es crear/diseñar el programa del agente.

Programa para el agente reactivo simple (aspiradora) en el entorno definido por las dos cuadrículas.

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

¿Agentes en IA?

- Ejemplos:
  - Asistentes virtuales Siri, Cortana o Alexa.
  - Los bots de chat que ofrecen atención al cliente o información sobre productos o servicios en páginas web o aplicaciones móviles.
  - Los sistemas de recomendación que sugieren productos, contenidos o servicios en función de las preferencias o el historial de los usuarios, como los que usan Amazon, Netflix o Spotify.
  - Los asistentes de voz que permiten controlar dispositivos inteligentes como luces, termostatos o cámaras de seguridad mediante comandos de voz, como los que integran Google Home o Amazon Echo.

¿Qué significa actuar de forma racional?

Racionalidad: Un sistema es racional si hace «lo correcto», en función de su conocimiento.

- Según Herbert Simon
  - El Principio de la Racionalidad Restringida alega que la racionalidad óptima ideal NO es el buen éxito perfecto.
  - Ningún ser humano apela a una racionalidad mayor a la que es necesaria para sus fines prácticos.
  - Las limitaciones de un agente
    - con los SENSORES que tiene
    - con los EFECTORES que tiene y
    - con la POTENCIA COMPUTACIONAL
      - disponible y
      - (en algunos casos) óptima económica
    - conducen a que la racionalidad ideal sea
      - imposible e
      - impráctica.

https://youtu.be/lRNp9SS9G-U

Tarea

- Definir los conceptos de:
  - Omnisciencia, aprendizaje y autonomía (mostrar ejemplo de cada uno).
- Propiedades y tipos de entorno/ambiente
  - Totalmente observable, parciamente observable, agente único vs multiagente, determinista, estocástico, episódico, secuencial, estático, dinámico, discreto continuo.
- Tipos de agentes (características, diagramas, limitaciones, ejemplo)
  - Agentes reactivos simples, Deliberativo, híbrido, Agentes reactivos basados en modelos, Agentes basados en objetivos, Agentes basados en utilidad, Agentes que aprenden.

Representación del problema como espacio de estados

Resolución del Problema

- La resolución de problemas es una capacidad que consideramos **inteligente**
- Somos capaces de resolver problemas variados
  - Encontrar el camino en un laberinto
  - Resolver un crucigrama
  - Jugar a un juego
  - Diagnosticar una enfermedad
  - Decidir si invertir en bolsa
- El objetivo es que un programa también sea capaz de resolverlos

Resolución del Problema

- Deseamos definir cualquier tipo de problema de manera que se pueda resolver automáticamente

- Necesitamos:
  - Una representación común para todos los problemas
  - Algoritmos que usen alguna estrategia para resolver problemas definidos en esa representación común

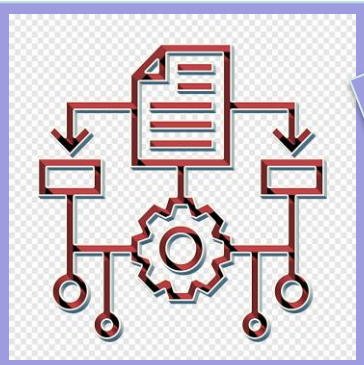
Método de solución del problema

El Problema

Abstracción

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \end{array}$$

Representación como espacio de estados



Interpretación

Solución

Aplicación de los algoritmos de búsqueda de la solución

Implementación en un programa/software

Definición del problema

- Un **problema** puede definirse, formalmente, por cuatro componentes:
  - Un **estado inicial** o punto de partida
  - Una descripción de las posibles **acciones y restricciones** sobre el objetivo
  - Un **estado final** o objetivo a alcanzar
  - Una función **costo del camino** que asigna un costo numérico a cada camino (Elementos que son relevantes en el problema definidos por el tipo de dominio).

Representación del problema

- Existen diferentes formas de representar problemas para resolverlos de manera automática
- Representaciones generales
  - **Espacio de estados:** un problema se divide en un conjunto de pasos de resolución desde el inicio hasta el objetivo
  - **Reducción a subproblemas:** un problema se puede descomponer en una jerarquía de subproblemas
- Representaciones para problemas específicos
  - **Resolución de juegos**
  - **Satisfacción de restricciones**

Representación del problema

Estados

- Podemos definir un problema por los elementos que intervienen y sus relaciones
- En cada instante de la resolución de un problema esos elementos tendrán unas características y relaciones específicas
- Denominaremos **Estado** a la representación de los elementos que describen el problema en un momento
- Distinguiremos dos estado especiales el **Estado Inicial** (punto de partida) y el **Estado Final** (objetivo del problema)
- ¿Que incluir en el estado?

Modificación del estado

Operadores

Representan un conjunto finito de acciones básicas que transforman unos estados en otros

- Para poder movernos entre los diferentes estados necesitamos operadores de transformación
- **Operador:** Función de transformación sobre la representación de un estado que lo convierte en otro estado
- Los operadores definen una relación de accesibilidad entre estados

Resumen

Descripción del problema

- Definir el conjunto de estados del problema (explícita o implícitamente)
- Especificar el estado inicial
- Especificar el estado final o las condiciones que cumple
- Especificar los operadores de cambio de estado (condiciones de aplicabilidad y función de transformación)
- Especificar el tipo de solución:
  - La secuencia de operadores o el estado final
  - Una solución cualquiera, la mejor (definición de coste), . . .

Resumen

Solución del problema

- **Solución:** Secuencia de pasos que llevan del estado inicial al final (secuencia de operadores) o también el estado final
- **Tipos de solución:** una cualquiera, la mejor, todas
- **Coste de una solución:** Gasto en recursos de la aplicación de los operadores a los estados. Puede ser importante o no según el problema y que tipo de solución busquemos

Problema del puzzle-8

Planteamiento del problema

En un tablero cuadrado se disponen 8 bloques cuadrados y enumerados dejando un hueco del mismo tamaño del los bloques, de tal forma que cualquiera de los bloques adyacentes al hueco puedan deslizarse hacia el. El juego consiste en pasar de un ordenamiento inicial a uno final mediante el deslizamiento de los bloques. Consideremos los siguientes casos:

Estado inicial

Estado final

Problema del puzzle-8

Representación de estados

- ✓ Espacio de estados: Configuraciones de 8 fichas en el tablero
- ✓ Estado inicial: Cualquier configuración
- ✓ Estado final: Fichas en orden específico
- ✓ Solución: Qué pasos + El menor número

- Representación *vs.* Implementación
  - Lista: **(2 8 3 1 6 4 7 X 5)**, **(2 8 3 4 5 X 7 1 6)**
  - Matriz: **((2 8 3)(1 6 4)(7 X 5))**
  - Literales: **((primera-izquierda 2) (primera-centro 8) ...)**

- Número de estados: 9!/2 = 181440.

Problema del puzzle-8

Operadores

- Según los movimientos de los bloques: 32.
- Según los movimientos del hueco: 4.
- **Condiciones**: El movimiento está dentro del tablero
- **Transformación**: Intercambio entre el hueco y la ficha en la posición del Movimiento

Referencias de lectura

- **El modelado de problemas**

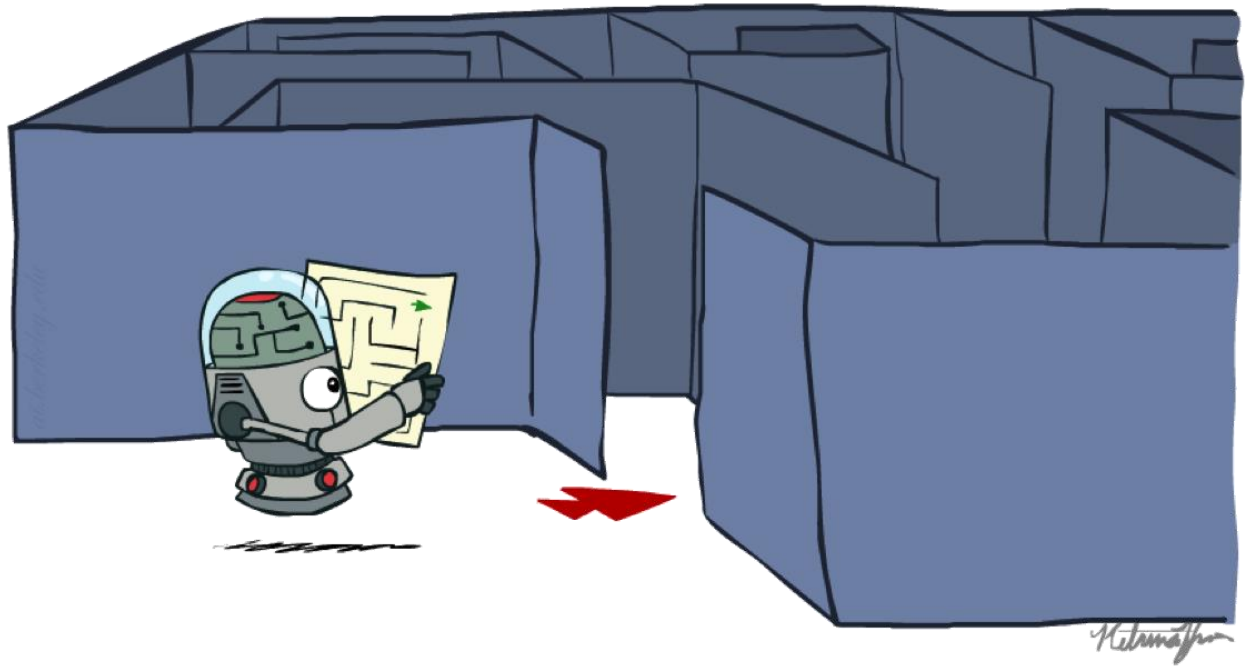
  http://www.cs.us.es/~fsancho/?e=94

- **Problemas búsqueda y planificación**

  http://www.cs.us.es/~fsancho/?e=33

Agentes que planifican con anticipación

Problemas de búsqueda



Las figuras fueron tomadas de las diapositivas de Dan Klein y Pieter Abbeel del curso de verano en la UC Berkeley]

Algoritmos de búsqueda

En IA se estudia **agentes constructores** que actúan de forma racional. La mayoría de las veces, estos agentes realizan algún tipo de búsqueda en segundo plano para lograr sus tareas.

Un problema de búsqueda consiste en:

- **Un espacio de estado** es un conjunto de todos los estados posibles en los que puede estar.
- **Un estado de inicio**. El estado desde donde comienza la búsqueda.
- **Un estado final (objetivo)**. Una función que mira el estado actual devuelve si es el estado objetivo o no.

Algoritmos de búsqueda

Características

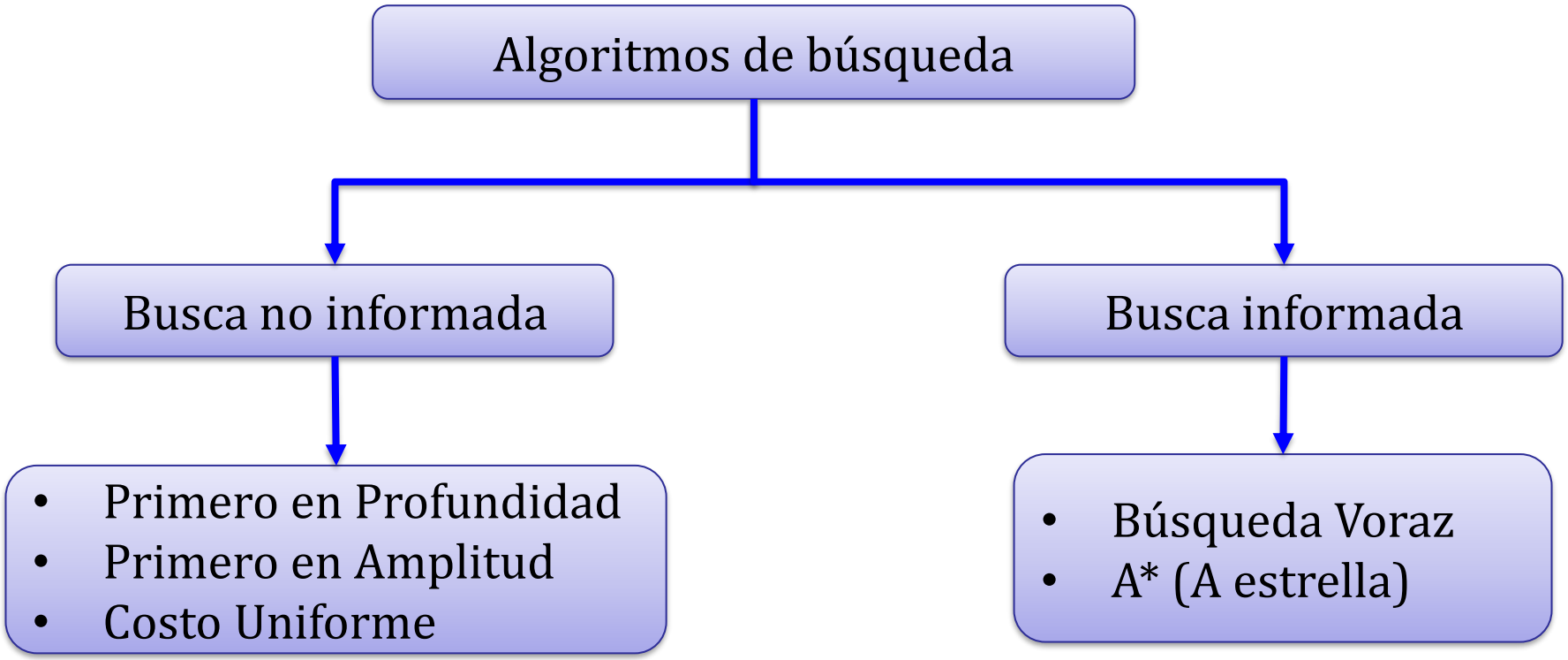
- **Completitud:** si existe solución, ¿la encuentra?
- **Optimización:** ¿obtiene la solución de menor número de pasos o de menor coste?
- **Complejidad en tiempo:** ¿cuánto tarda en encontrar una solución?
- **Complejidad en espacio:** ¿cuánta memoria necesitamos?

Son las que miden el rendimiento de la resolución del problema

Algoritmos de búsqueda

Tipos de algoritmos de búsqueda

Hay varios algoritmos de búsqueda muy eficientes y poderosos, los mas elementales son:



Búsqueda no informada

También conocida como Búsqueda a ciegas

- **No tienen en cuenta el coste de la solución** en la búsqueda
- Su funcionamiento es sistemático, siguen un orden de visitas y generación de nodos establecido por la estructura del espacio de búsqueda
- Anchura prioritaria, Profundidad prioritaria, Profundidad iterativa

Lectura: http://www.cs.us.es/~fsancho/?e=95

Búsqueda no informada

También conocida como Búsqueda a ciegas

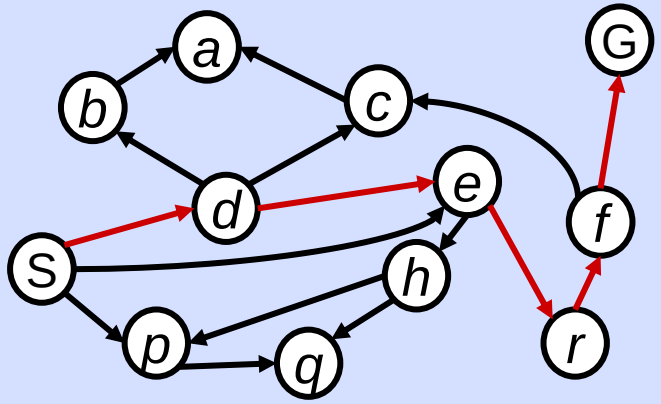
Deben presentar lo siguiente:

- Un gráfico del problema, que contiene el nodo inicial **S** y el nodo objetivo **G.**
- Una estrategia que describe la forma en que se recorrerá la gráfica para llegar a **G.**
- Una franja, que es una estructura de datos que se utiliza para almacenar todos los estados posibles (nodos) a los que puede ir desde los estados actuales.
- Un árbol, que resulta al atravesar el nodo objetivo.
- Un plan de solución, que la secuencia de nodos de **S** a **G.**

Búsqueda no informada

Grafos vs Árbol de búsqueda

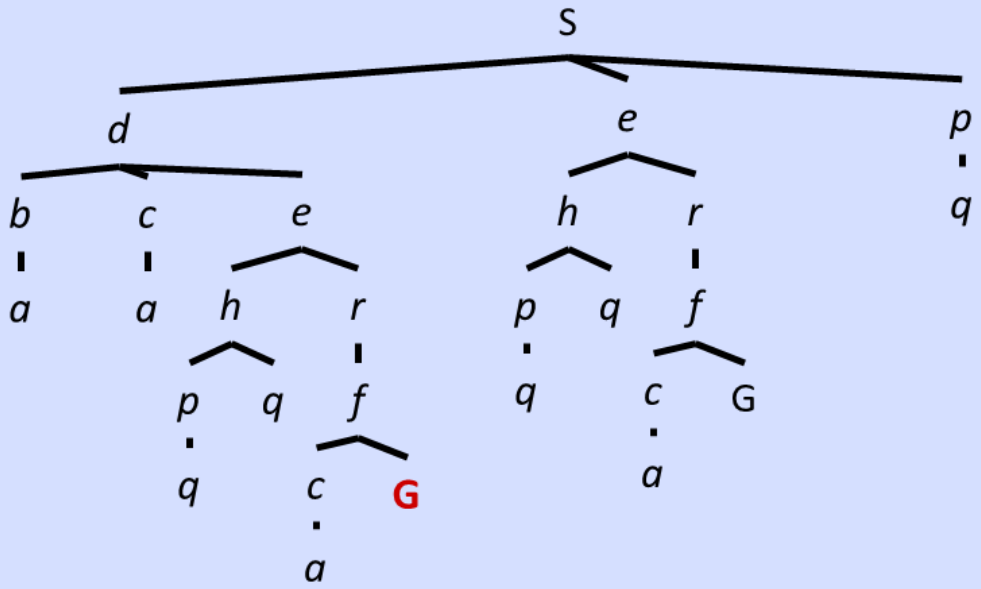
Grafo del espacio de estados



Cada NODO en el árbol de búsqueda es un camino completo en el grafo de espacio de estados.

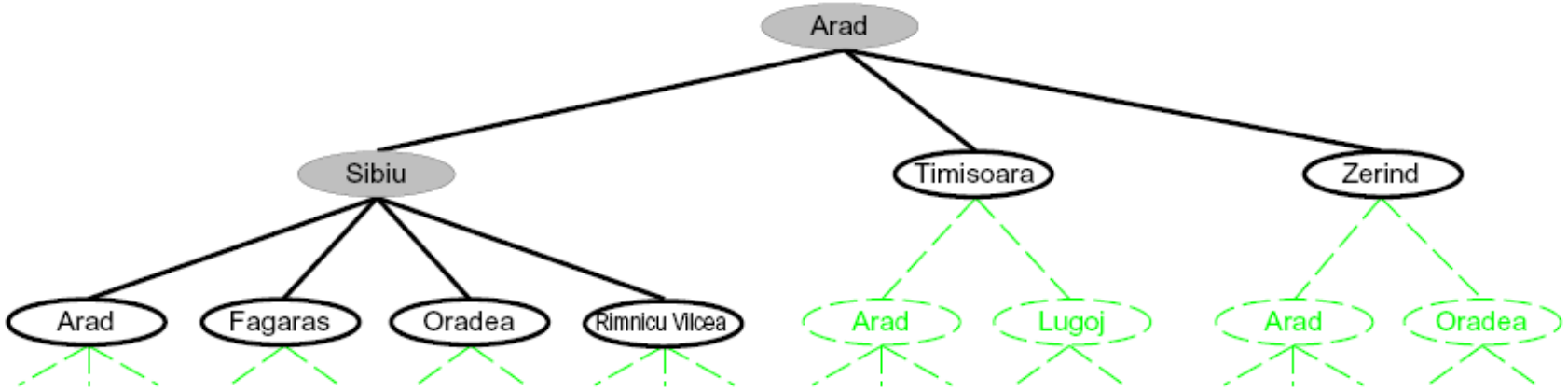
Construimos de acuerdo a lo que se necesite, es decir, lo menos posible.

Árbol de Búsqueda



Búsqueda no informada

Búsqueda con un árbol de búsqueda



- Búsqueda:
  - Ampliar las rutas potenciales (los nodos del árbol)
  - Mantener una franja de las rutas potenciales en consideración
  - Expandir la menor cantidad de nodos posibles

Búsqueda no informada

Búsqueda con un árbol de búsqueda

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

- Ideas importantes:
  - Franja
  - Expansión
  - Estrategia de Exploración
- La pregunta: ¿Que franja de los nodos debemos explorar?

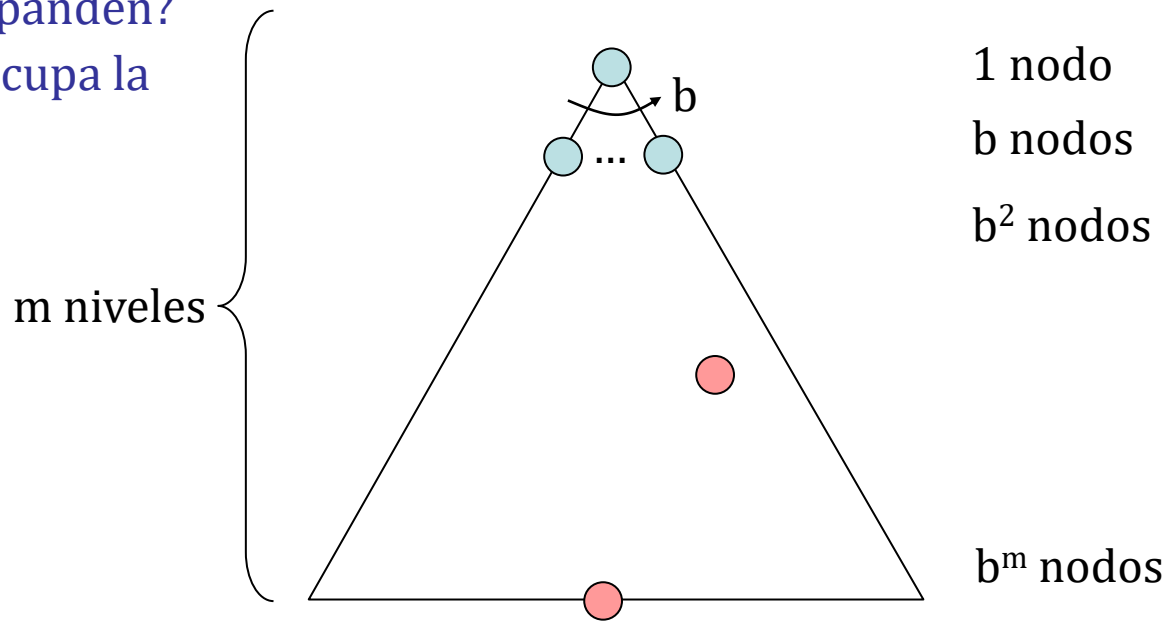
Búsqueda no informada

Características del algoritmo

- **Completitud**: ¿Garantiza encontrar la solución si esta existe?
- **Optimo**: ¿Garantiza encontrar la ruta de menor coste?
- **Complejidad en el tiempo**: ¿ Qué nodos se expanden?
- **Complejidad en el espacio**:¿Cuanto espacio ocupa la franja?

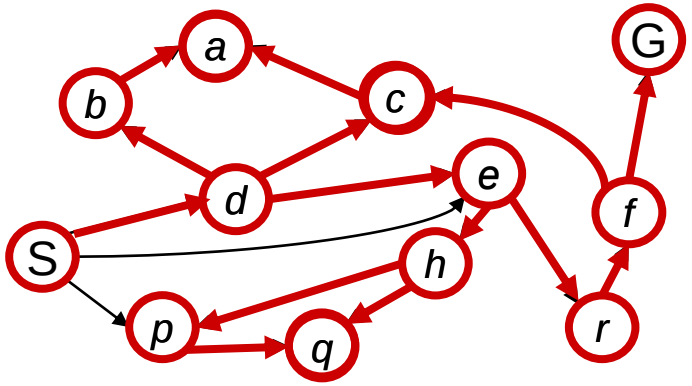
- Esquema del árbol de búsqueda:
  - b es el factor de ramificación
  - m es la profundidad máxima
  - Soluciones a diferentes profundidades

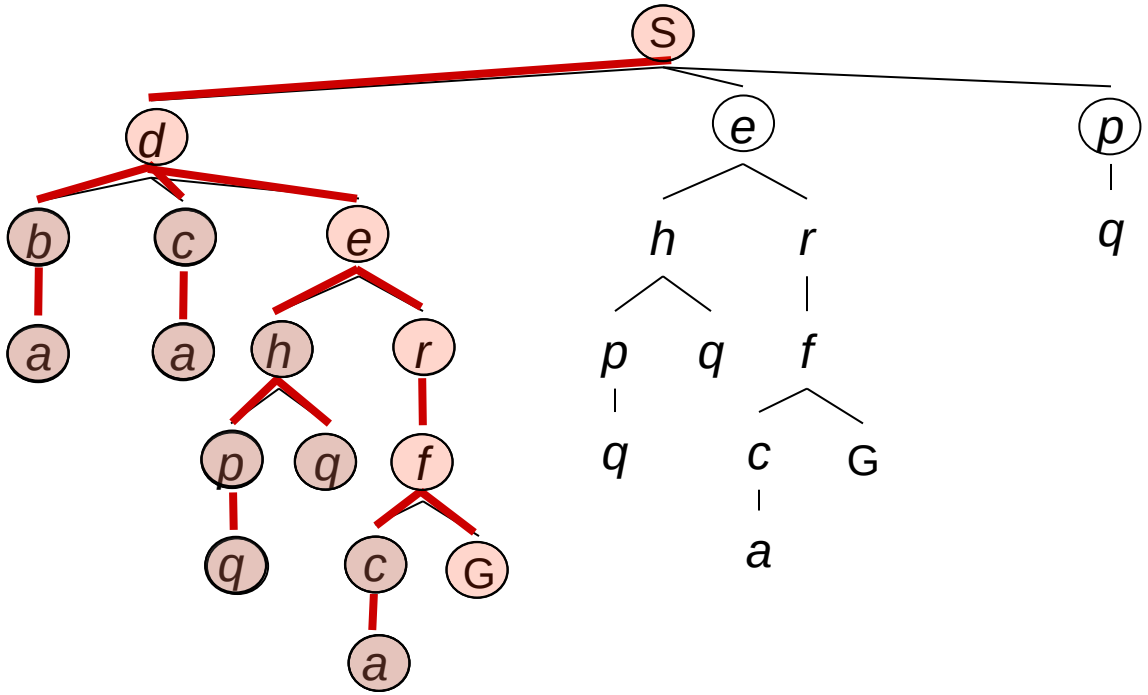
- ¿Numero de nodos en todo el árbol?
  - $1 + b + b^2 + ... b^m = O(b^m)$



Primero en Profundidad

Estrategia: expandir al nodo mas profundo

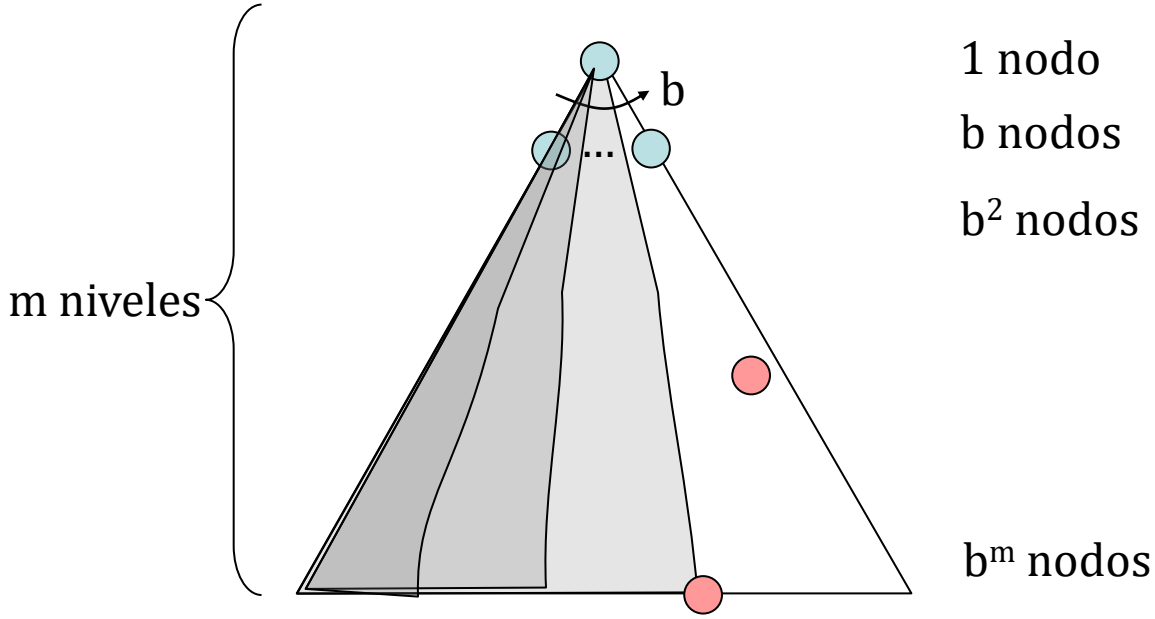




Primero en Profundidad

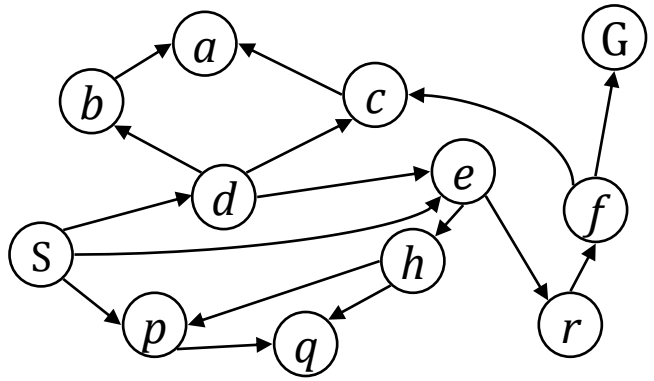
Propiedades

- ¿Que nodos se expanden?
  - Algunos dejan el prefijo del árbol.
  - Puede procesar todo el árbol!
  - Si es finito, toma tiempo $O(b^m)$
- ¿Cuanto espacio ocupa la franja?
  - Si solo tiene hermanos en dirección a la raíz, entonces O(bm)
- ¿Es completo?
  - m podría ser finito, solo hay que evitar los ciclos (usar la búsqueda en profundidad iterativa)
- ¿Es optimo?
  - No, encuentra la solución “mas a la izquierda”, independientemente de la profundidad del costo.

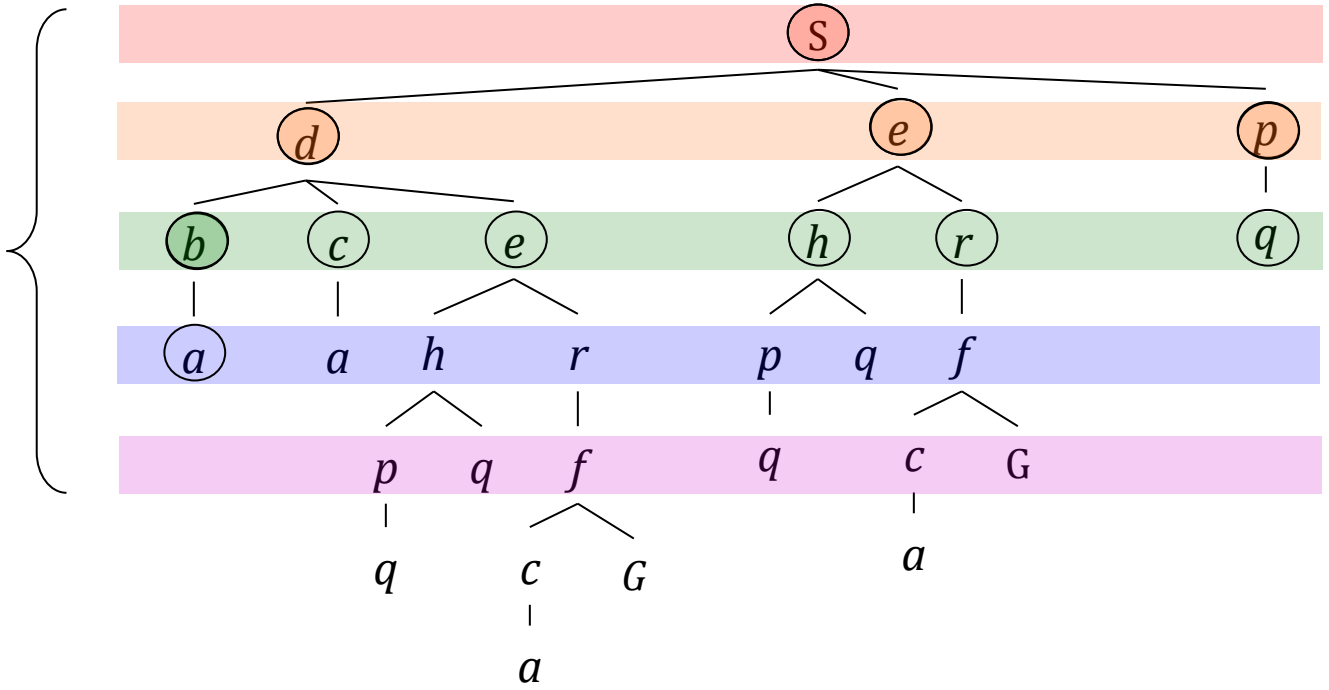


Primero en Amplitud

Estrategia: expande primero a un nodo menos profundo

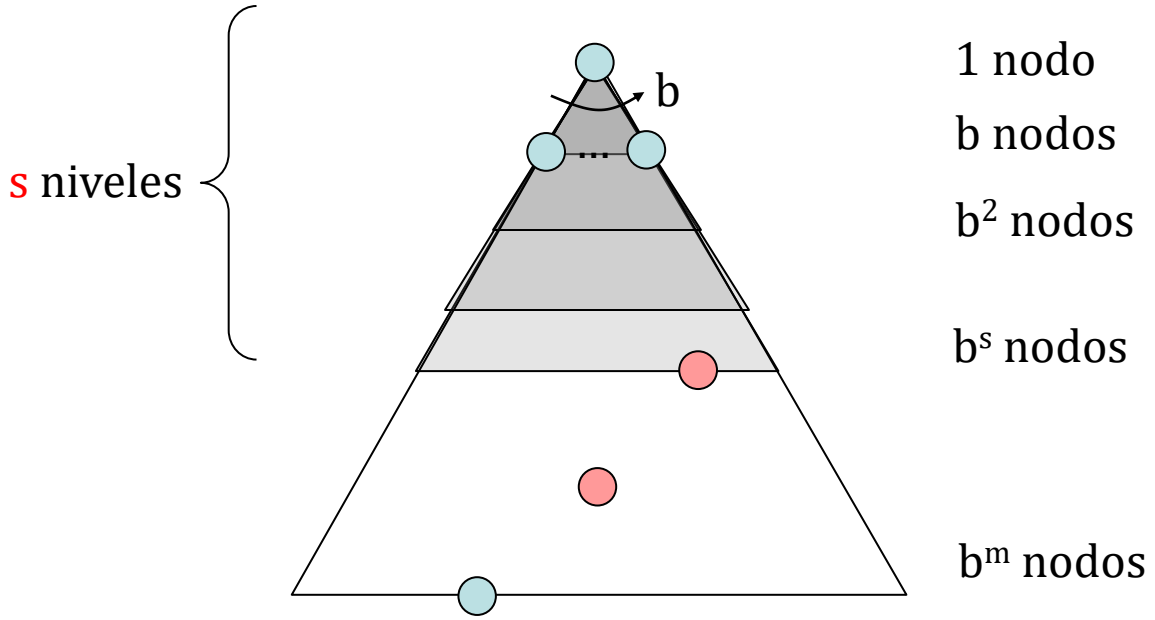


Búsqueda por niveles



Primero en Amplitud

- ¿Que nodos se expanden?
  - Procesa todos los nodos por encima de la solución
  - La profundidad de la solución es la mas superficial **s**
  - Toma tiempo $O(b^s)$
- ¿Cuanto espacio ocupa la franja?
  - El ultimo nivel es s, entonces $O(b^s)$
- ¿Es completo?
  - s debe ser finito si la solución existe, entonces si!
- ¿Es optimo?
  - Solo si todos los costos son 1



Preguntas

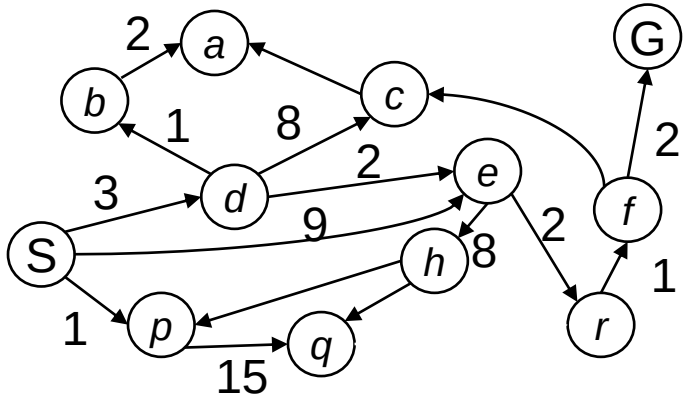
- ¿Cuando la búsqueda primero profundidad supera a la de primero en amplitud?

- ¿Cuando la búsqueda primero amplitud supera a la de primero en profundidad?

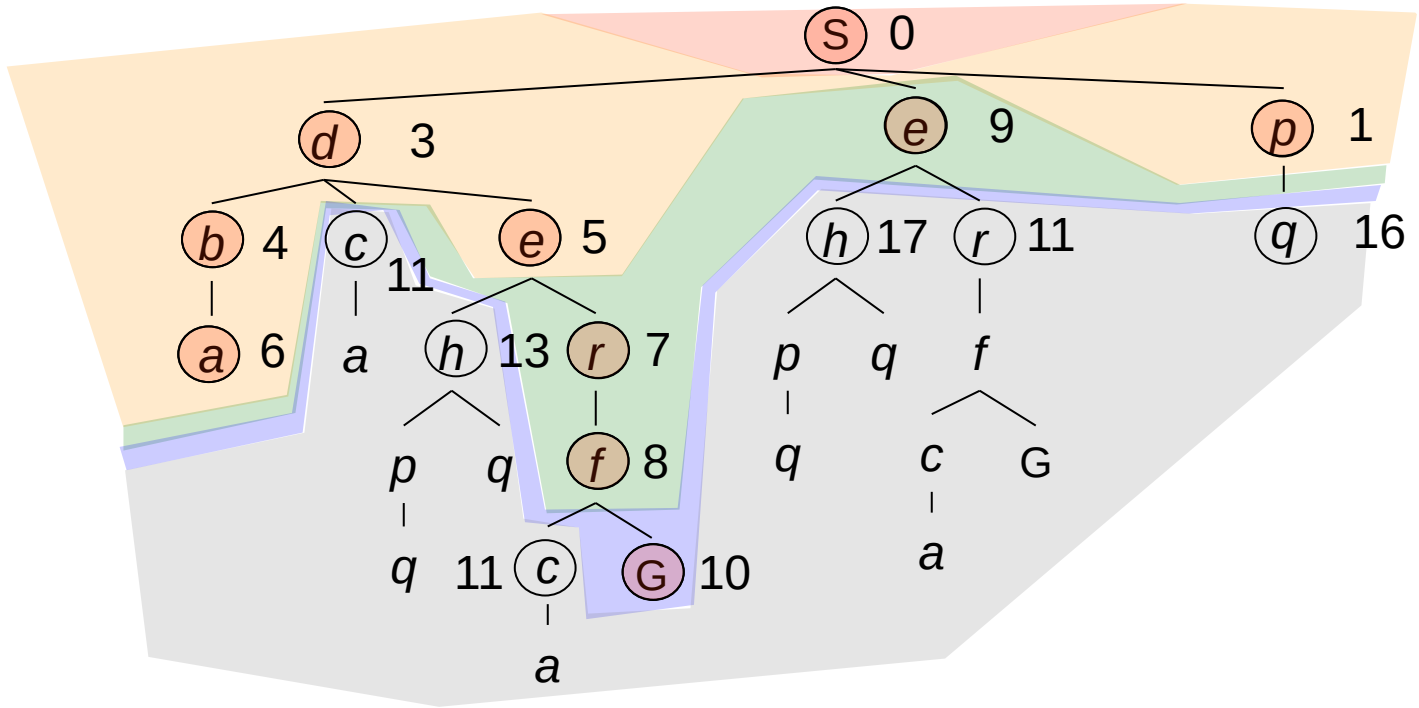
Búsqueda de Costo uniforme

Estrategia: Expandirse al nodo de menor costo

Prioriza el costo acumulativo

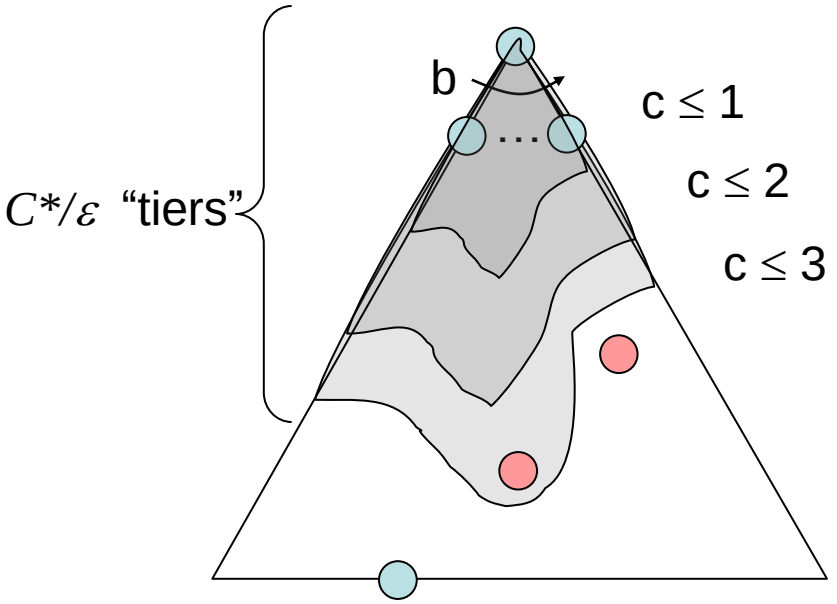


Costo de contorno



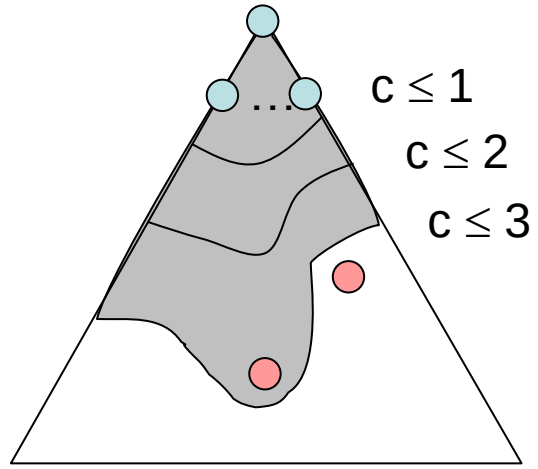
Búsqueda de Costo uniforme

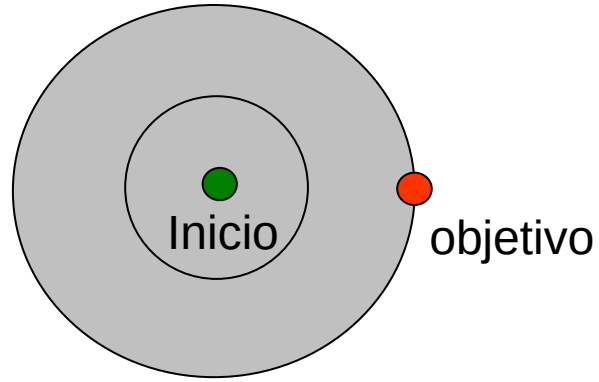
- ¿Que nodos expande este método?
  - Procesa/expande el nodo n con el camino de menor costo (no importa el numero de pasos)
  - Si todos los costos son iguales, es idéntico a la búsqueda primero en amplitud.
  - Si el coste de esa solución es C^* y las aristas cuestan al menos ε, entonces la profundidad efectiva es al menos C^*/ε
  - En el peor de los casos tarda un tiempo $O(b^{C^*/\varepsilon})$
- ¿Cuanto espacio ocupa la franja?
  - aproximadamente $O(b^{C^*/\varepsilon})$
- ¿Es completo?
  - Asumiendo que la mejor solución tiene un coste finito y el coste de las aristas son positivas y mínimas, Si!
- ¿Es optimo?
  - Si!



Búsqueda de Costo uniforme

- Recordar: Este método explora incrementando el coste del contorno
- Lo bueno: Es completo y optimo!
- Lo malo:
  - La opción de exploración es en todas las direcciones
  - No da información sobre la ubicación del objetivo
- Se puede mejorar!





Resumen de los métodos de búsqueda

Criterio	Primero en anchura	Costo uniforme	Primero en profundidad	Profundidad limitada	Profundidad iterativa	Bidireccional (si aplicable)
¿Completa?	Sí[a]	Sí[a, b]	No	No	Sí[a]	Sí[a, d]
Tiempo	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Espacio	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
¿Optimal?	Sí[c]	Sí	No	No	Sí[c]	Sí[c, d]

Evaluación de los métodos de búsqueda no informada (ciega). ***b*** es el factor de ramificación; ***d*** es la profundidad de la solución más superficial; ***m*** es la máxima profundidad del árbol de búsqueda; ***ℓ*** es el límite de profundidad. Los superíndice significan lo siguiente: ***a*** completa si ***b*** es finita; ***b*** completa si los costos son ≥ ϵ para ϵ positivo; ***c*** óptima si los costos son iguales; ***d*** si en ambas direcciones se utiliza la búsqueda primero en anchura.

Referencias

Leer el Capitulo 3 de: Artificial Intelligence - A Modern Approach [Stuart Russell, Peter Norvig] [3th Edition]

- Búsqueda primero en profundidad
  - https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_profundidad
  - https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/?ref=lbp
- Búsqueda primero en Amplitud
  - https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_anchura
  - https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/?ref=lbp
- Búsqueda de costo uniforme
  - https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/
  - https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Practical_optimizations_and_infinite_graphs