ISI437 - Inteligencia Artificial

Introducción y generalidades

Ing. Jose Eduardo Laruta Espejo

16 de febrero de 2020

Universidad La Salle - Bolivia

Contenido i

Agentes Planificadores

Agentes Reflejo

- Escoge una acción basado en el estado actual percibido (tal vez incluya algo de memoria).
- Puede tener memoria o un modelo del entorno en el estado actual.
- No considera futuras consecuencias de las acciones.
- Considera cómo el mundo está en un momento dado.

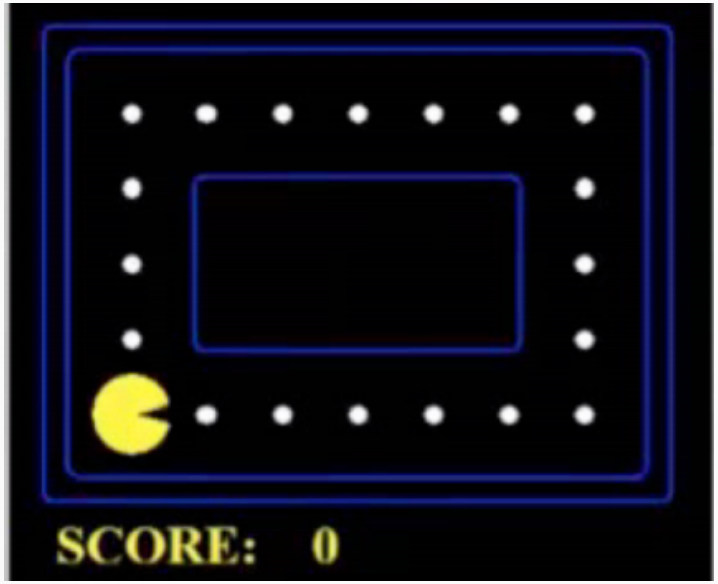
Agentes Reflejo

- Escoge una acción basado en el estado actual percibido (tal vez incluya algo de memoria).
- Puede tener memoria o un modelo del entorno en el estado actual.
- No considera futuras consecuencias de las acciones.
- Considera cómo el mundo está en un momento dado.

Puede un agente reflejo ser racional?

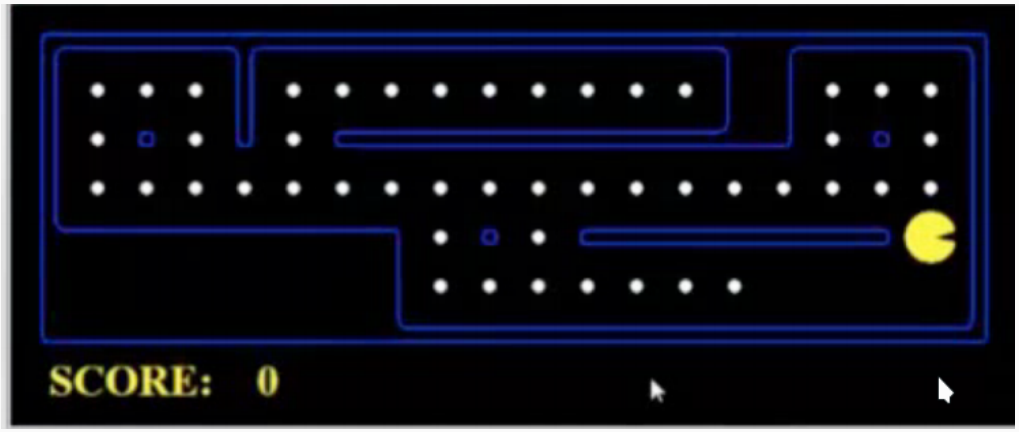
Agentes Reflejo

Considere a nuestro agente Pacman como un agente reflejo, la estrategia dicta a moverse en dirección de la comida más cercana.
¿Se puede considerar una solución óptima?



Agentes Reflejo 2

Considere a nuestro agente Pacman como un agente reflejo, la estrategia dicta a moverse en dirección de la comida más cercana.
¿Se puede considerar una solución óptima?



Agentes que planifican

- Se hacen la pregunta “que pasa si...”.
- Las decisiones se basan en las consecuencias de las acciones.
- Se necesita un modelo de cómo evoluciona el entorno en respuesta a las acciones.
- Se debe formular un objetivo.
- Considera cómo el entorno podría estar.

Planeamiento Óptimo vs Completo

ÓPTIMO

Se consigue el objetivo con costo mínimo.

COMPLETO

Cuando existe una solución, ésta es encontrada.

Planeamiento vs Replaneamiento

Planeamiento

Explora la secuencia de acciones completa y elige la mejor solución.

Replaneamiento

Define una estrategia a corto plazo y cuando consigue un objetivo secundario vuelve a calcular una nueva estrategia.

Problemas de búsqueda

Problemas de búsqueda

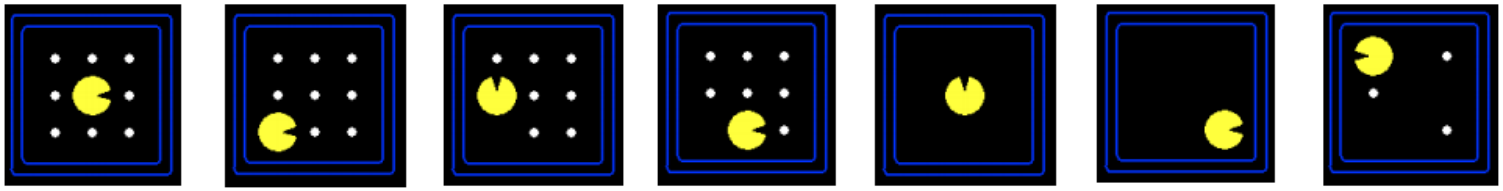
Un **problema de búsqueda** consiste en:

- Un espacio de estados.
- Una función sucesora.
- Un estado inicial y un objetivo.

Una **solución** es una secuencia de acciones (un plan) que transforma el estado inicial en el estado objetivo.

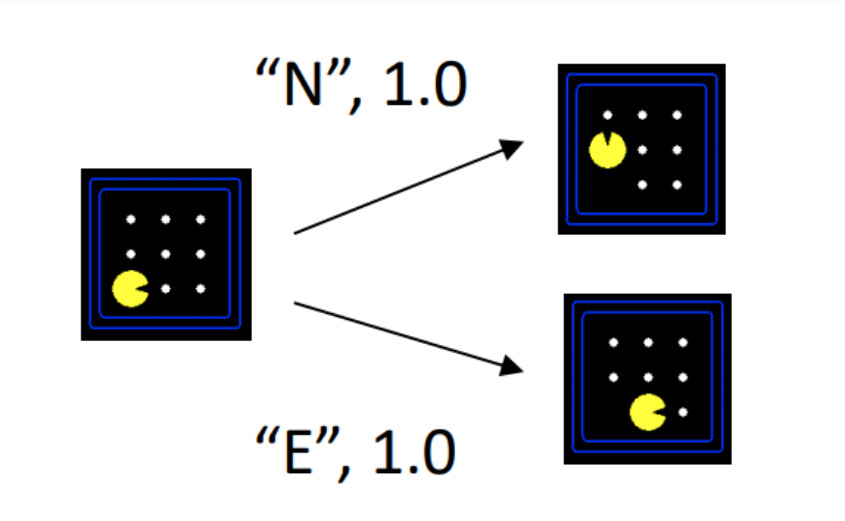
Espacio de estados

El **espacio de estados** es el conjunto de todas las posibles configuraciones del problema.



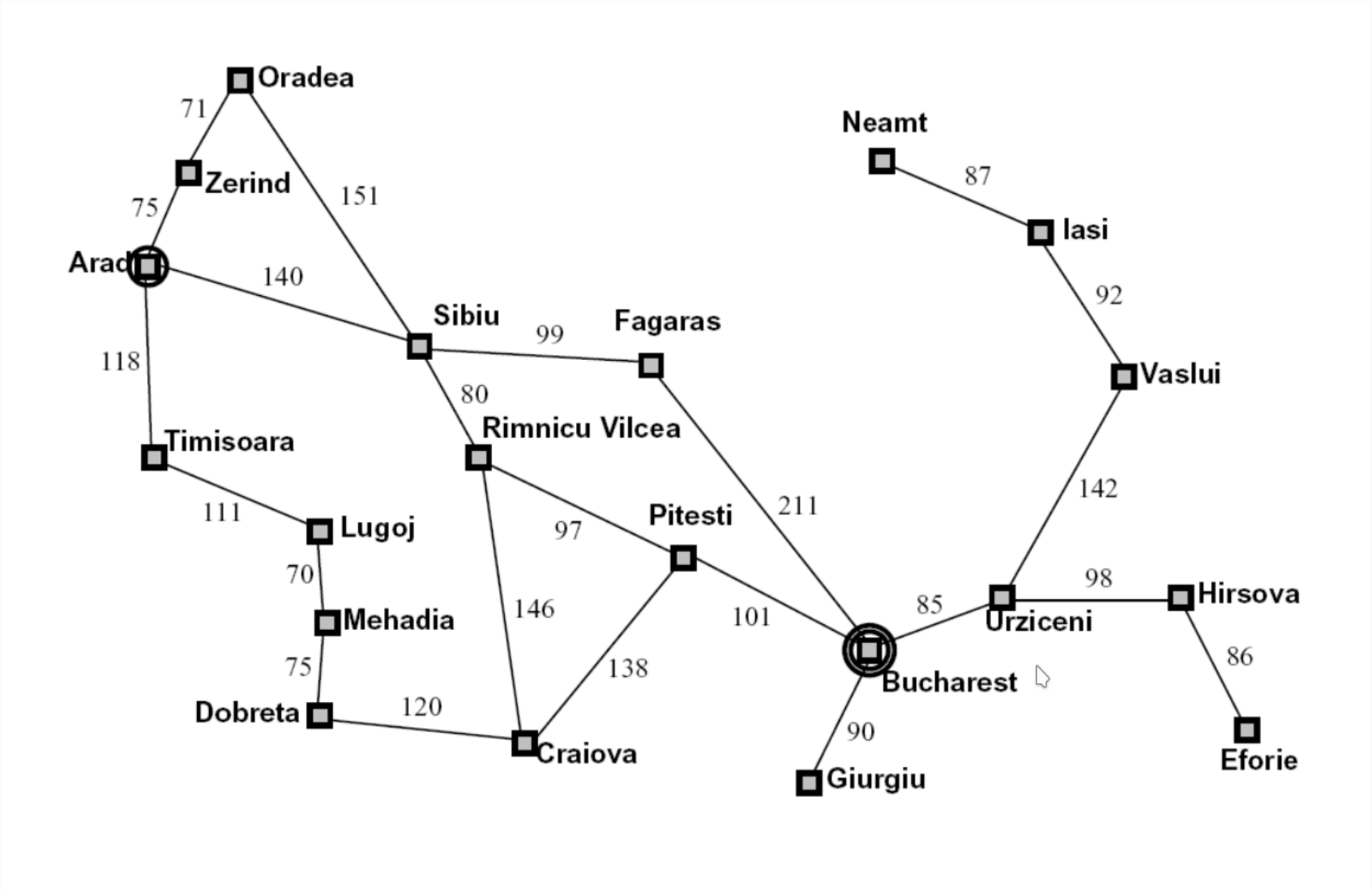
Función sucesora

La **función sucesora** define cuáles son las acciones disponibles para un estado dado y cuáles los estados consecuencia para dichas acciones.



Ejemplo

Ejemplo: Viajando por Rumania

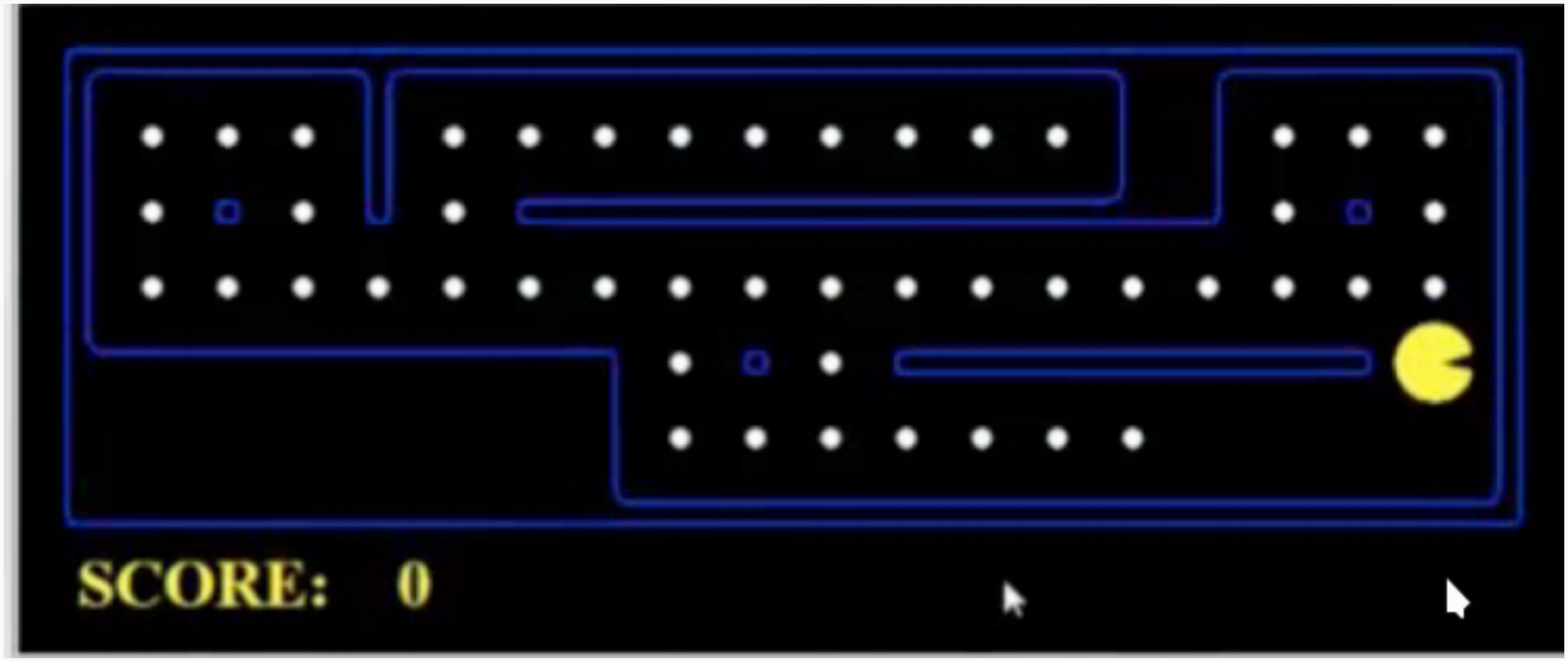


Ejemplo: Viajando por Rumania

- **Espacio de estados:** Ciudades
- **Función sucesora:** Carreteras, viaje a una ciudad adyacente con *costo* = *distancia*.
- **Estado Inicial:** Arad
- **Estado objetivo:** *estado* == *Bucharest*?.
- **Solución**

Ejemplo: Pacman

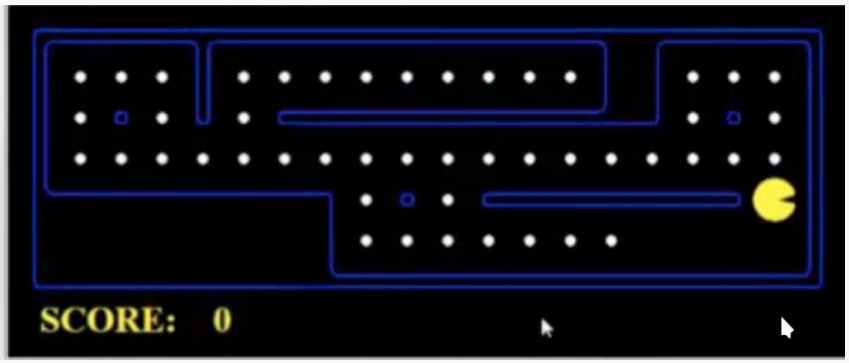
El estado del mundo incluye todos los detalles del entorno.



Ejemplo: Pacman

Problema: **Recorrido**

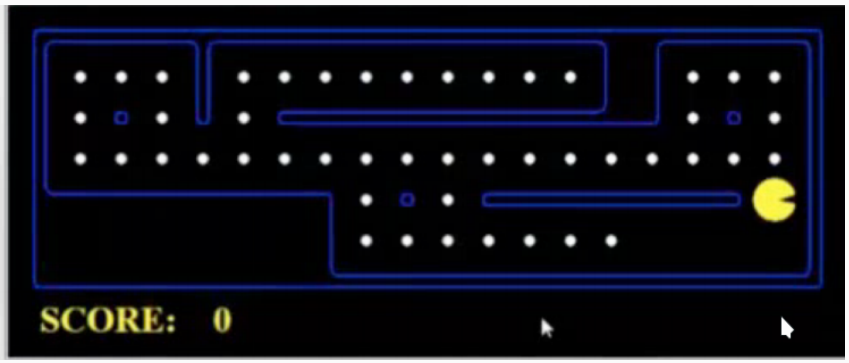
- Estados:
- Acciones:
- F. Sucesora:
- Objetivo:



Ejemplo: Pacman

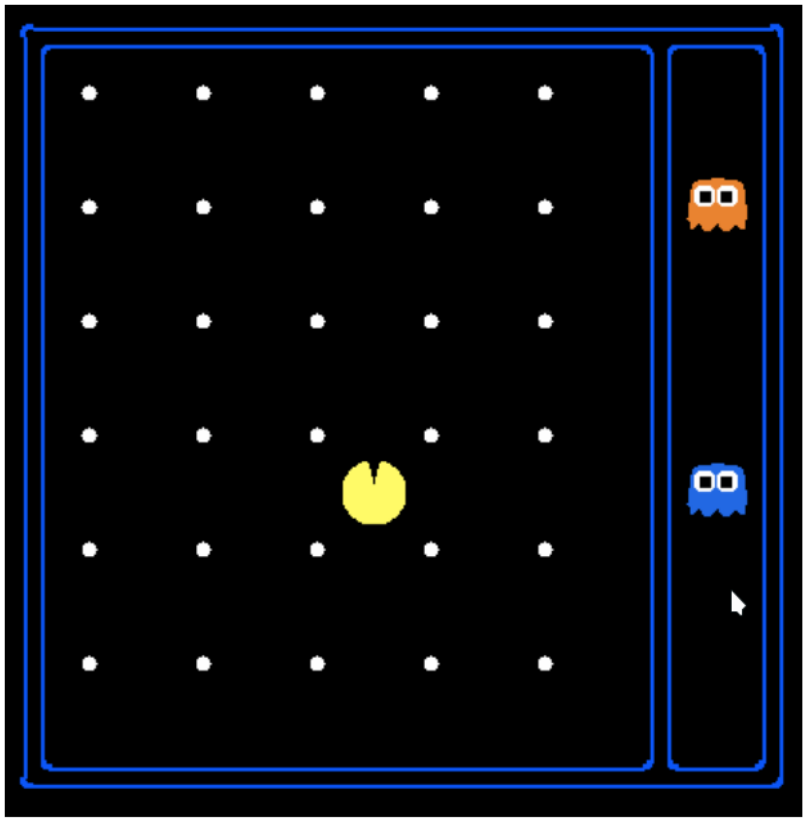
Problema: **Acabar toda la comida**

- Estados:
- Acciones:
- F. Sucesora:
- Objetivo:

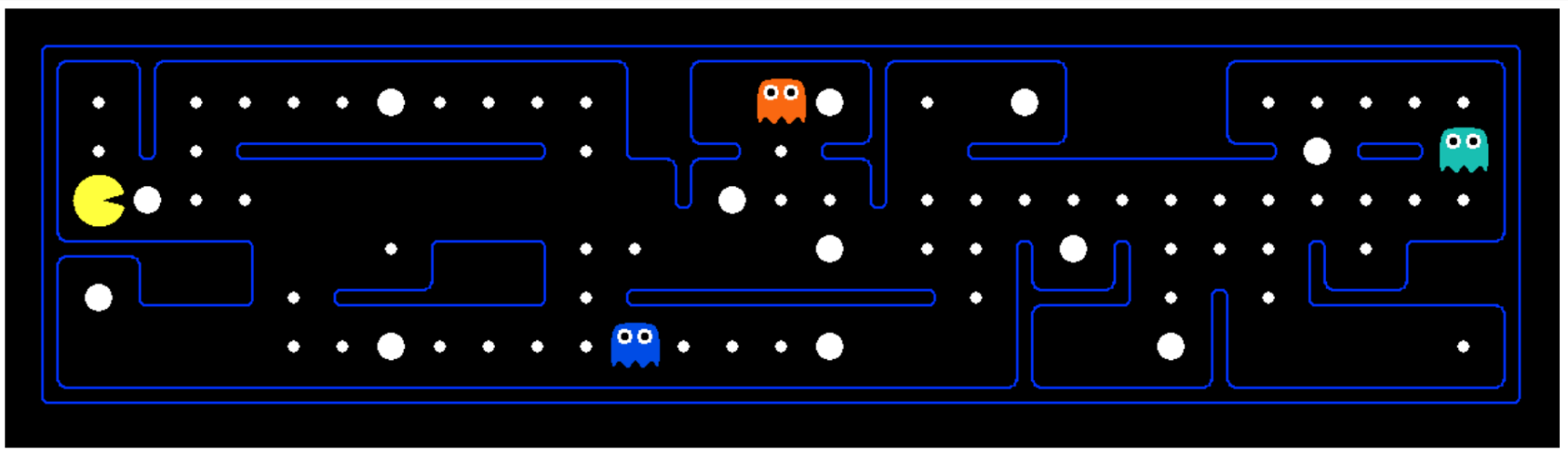


Dimensión del espacio de estados

- Estado del mundo:
  - Posiciones del agente: 120
  - Cantidad de puntos: 30
  - Posiciones de fantasmas: 12
  - Orientacion del agente: NSEW
- Cantidad:
  - Mundo: $120 \times (2^3 0) \times (12^2) \times 4$
  - Estados para recorrido: 120
  - Estados para comer: $120 \times 2^3 0$

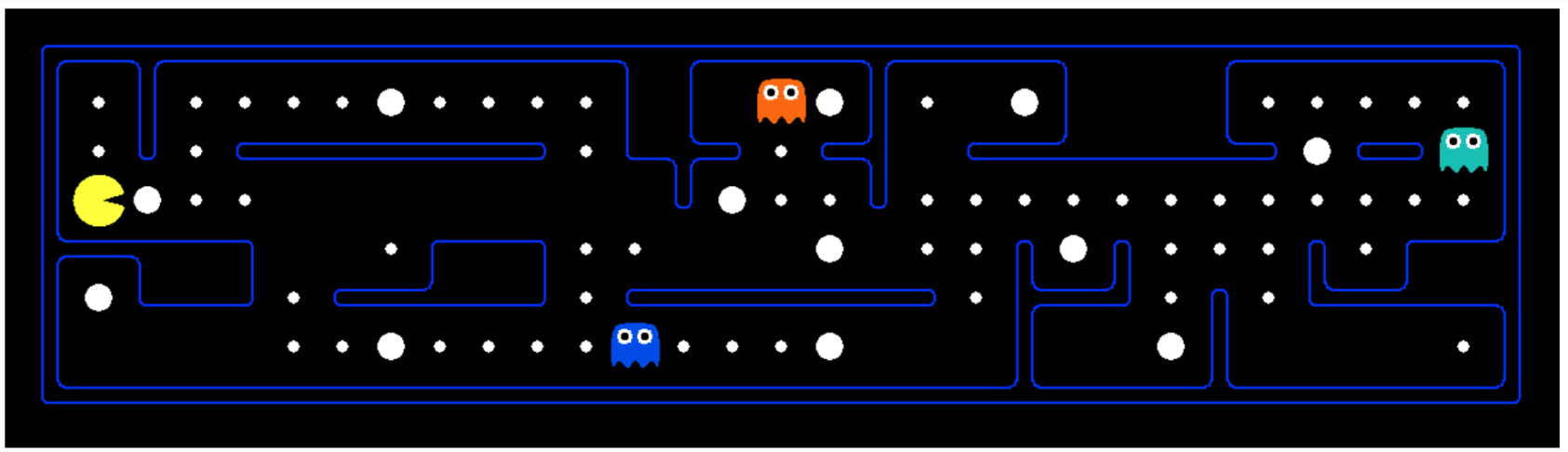


Ejercicio



Problema: Comer todos los puntos mientras se mantiene a los fantasmas asustados.

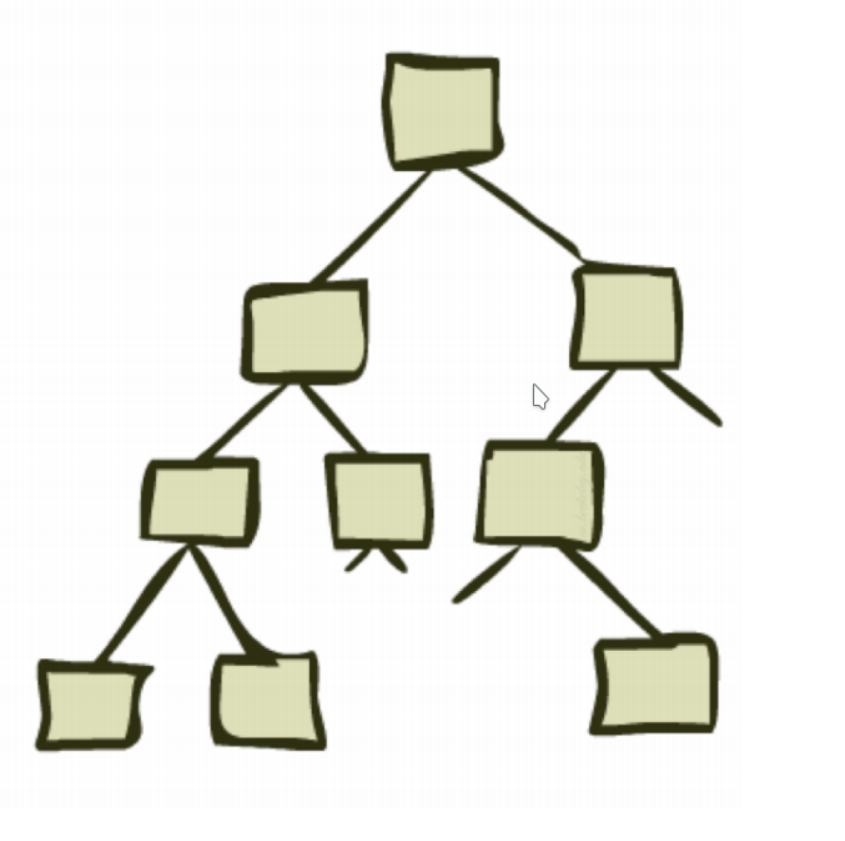
Ejercicio



Problema: Comer todos los puntos mientras se mantiene a los fantasmas asustados.

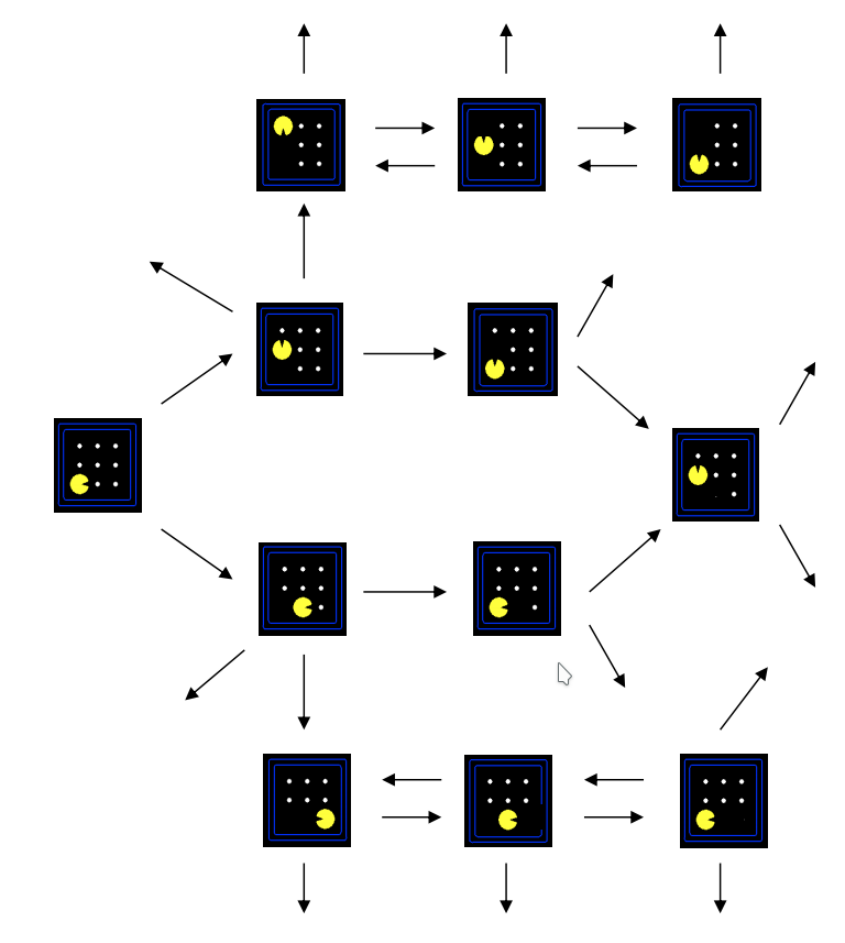
- Posición, puntos, puntos de poder, tiempo restante

Grafos de Espacio de estado



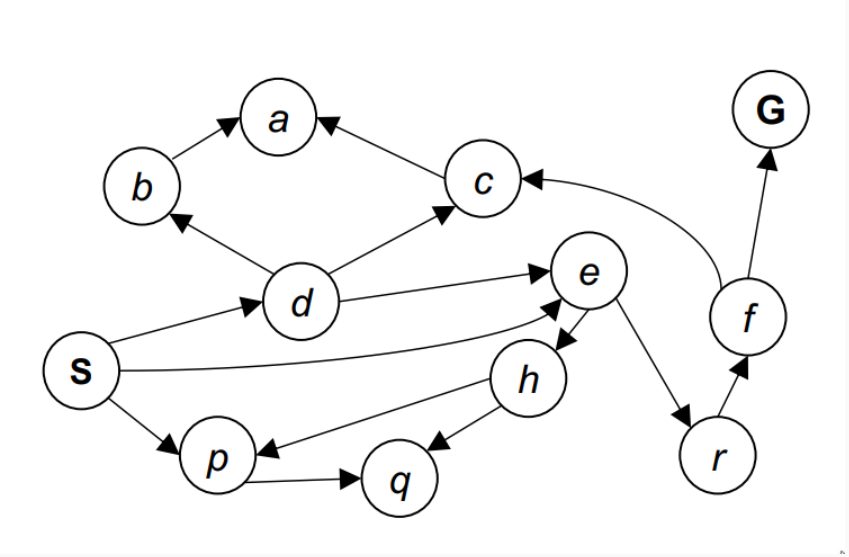
Grafos de Espacio de estado

- Representación matemática:
  - Los nodos son configuraciones abstractas.
  - Los arcos representan sucesores (resultados de acciones).
  - El objetivo es un conjunto de nodos objetivo (o uno solo).
- Cada estado solamente ocurre una vez.
- Es dificil construir este grafo completo en la memoria.

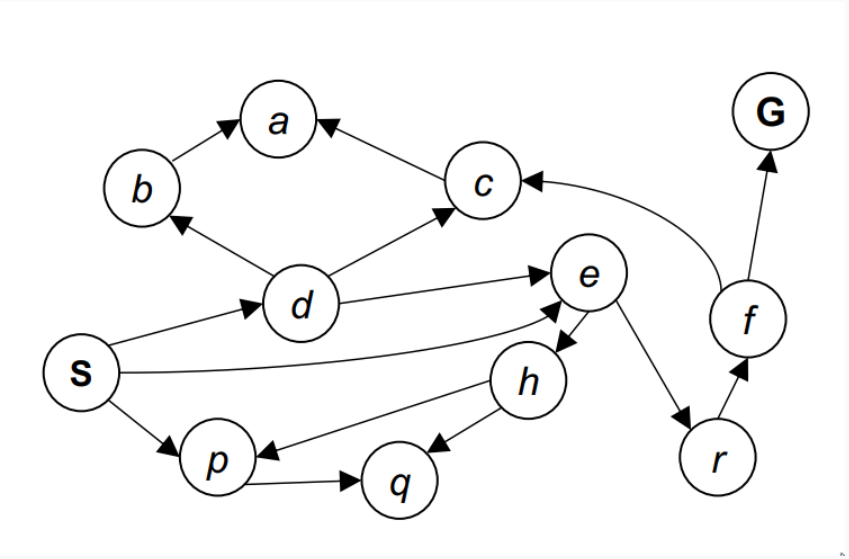


Grafos de Espacio de estado

- Representación matemática:
  - Los nodos son configuraciones abstractas.
  - Los arcos representan sucesores (resultados de acciones).
  - El objetivo es un conjunto de nodos objetivo (o uno solo).
- Cada estado solamente ocurre una vez.
- Es dificil construir este grafo completo en la memoria.



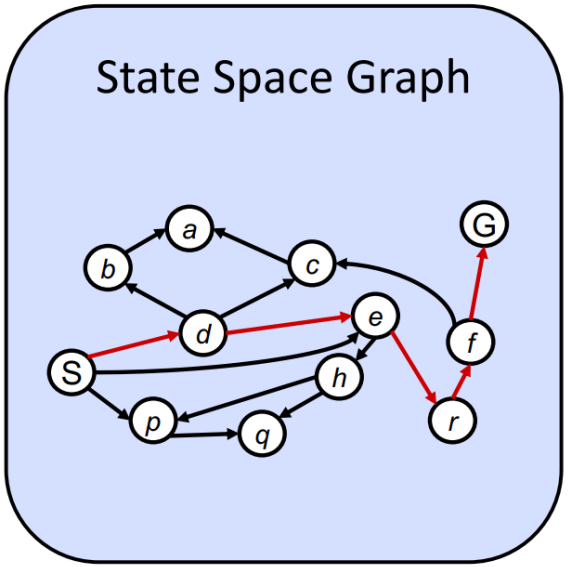
Árboles de búsqueda



Un árbol de búsqueda:

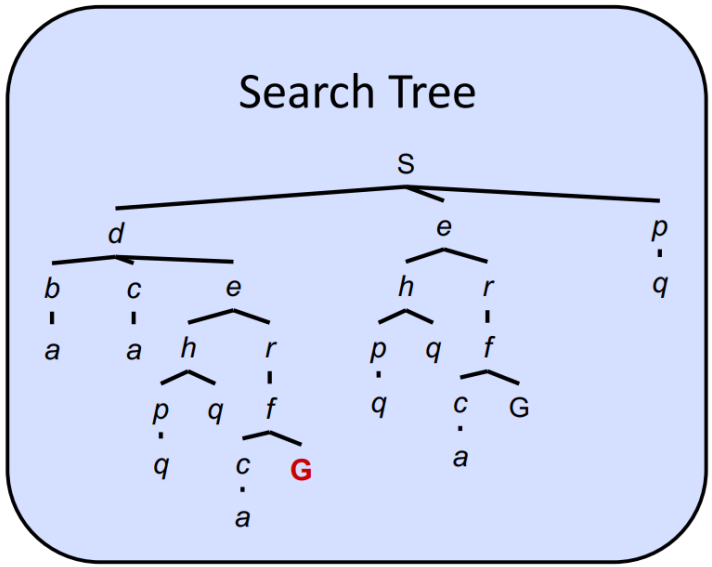
- Representa el "qué pasa si..." de los planes y sus resultados.
- El nodo inicial es el nodo raíz.
- Los nodos hijos corresponden a los sucesores.
- Los nodos muestran estados pero corresponden a planes.
- En la práctica no se construyen los árboles completos.

Árboles de búsqueda

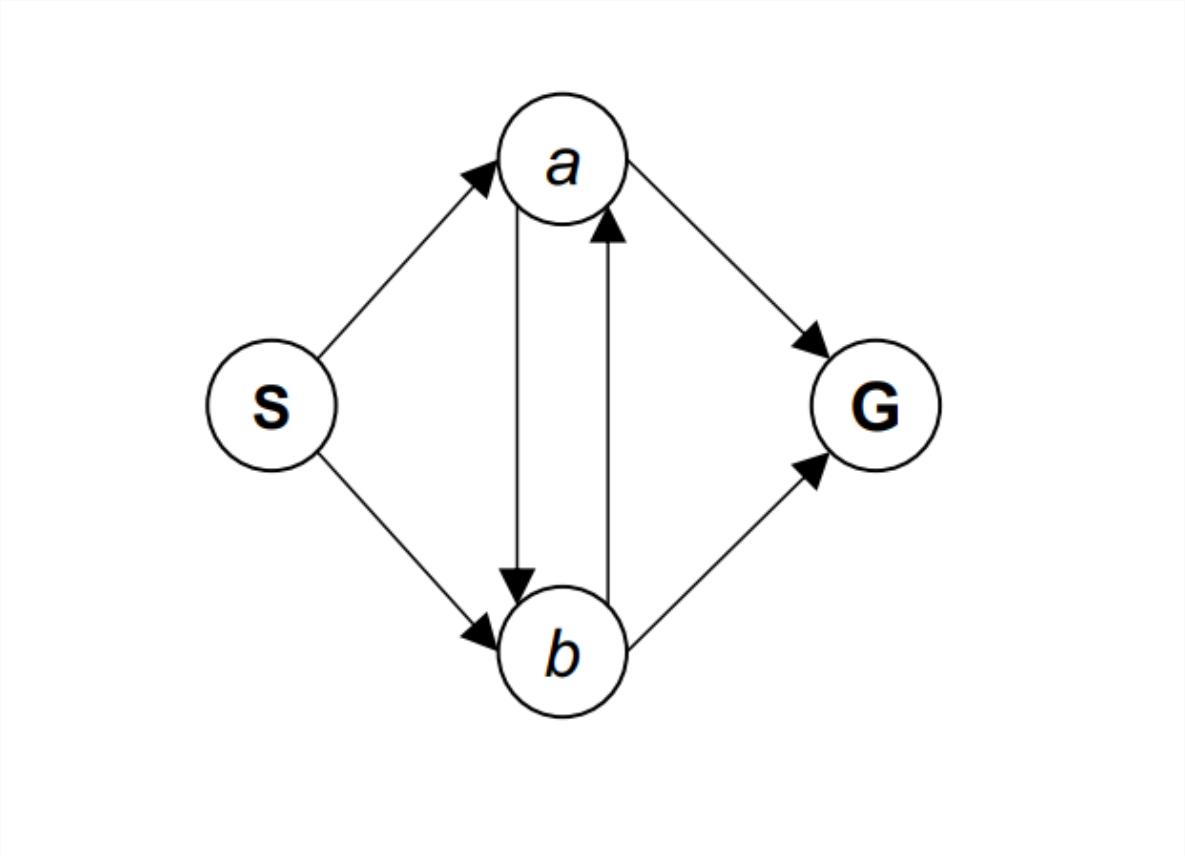


Each NODE in in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.

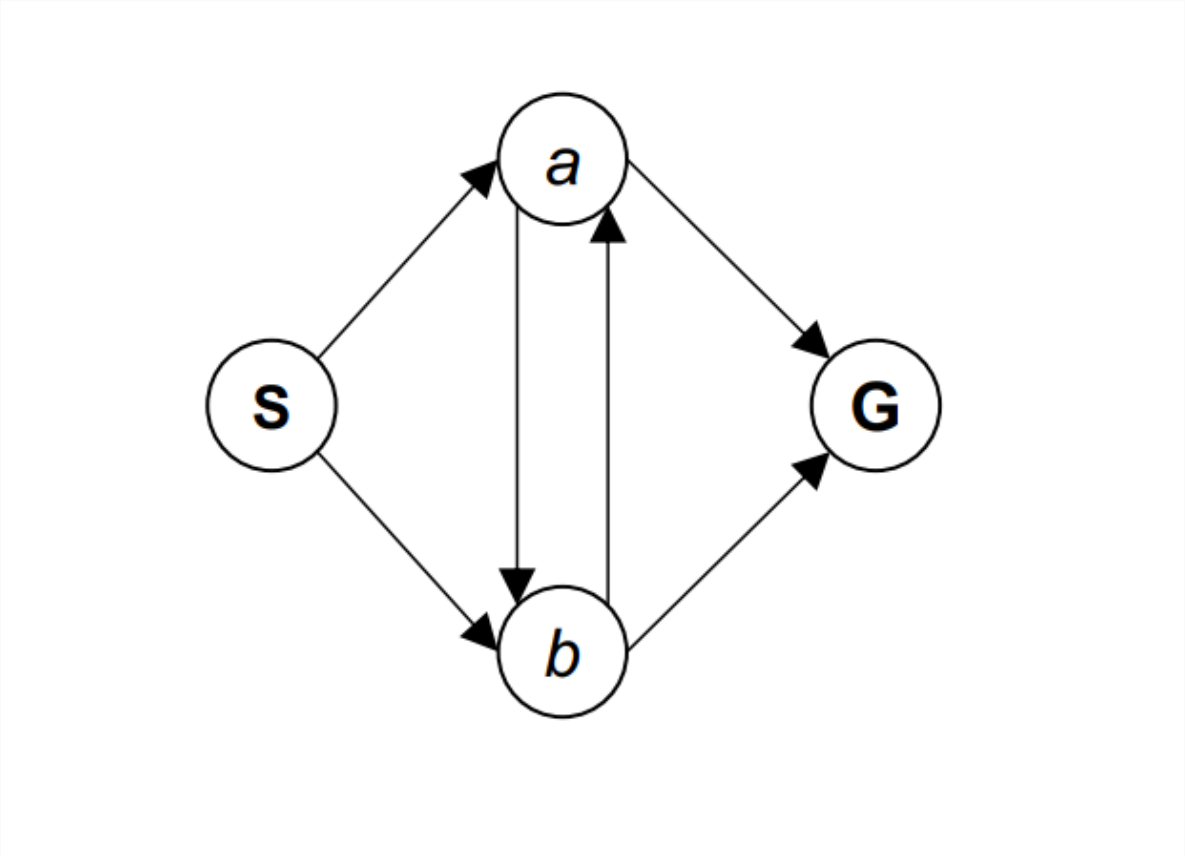


Ejercicio: Árboles de búsqueda



Qué tan grande es el árbol de búsqueda?

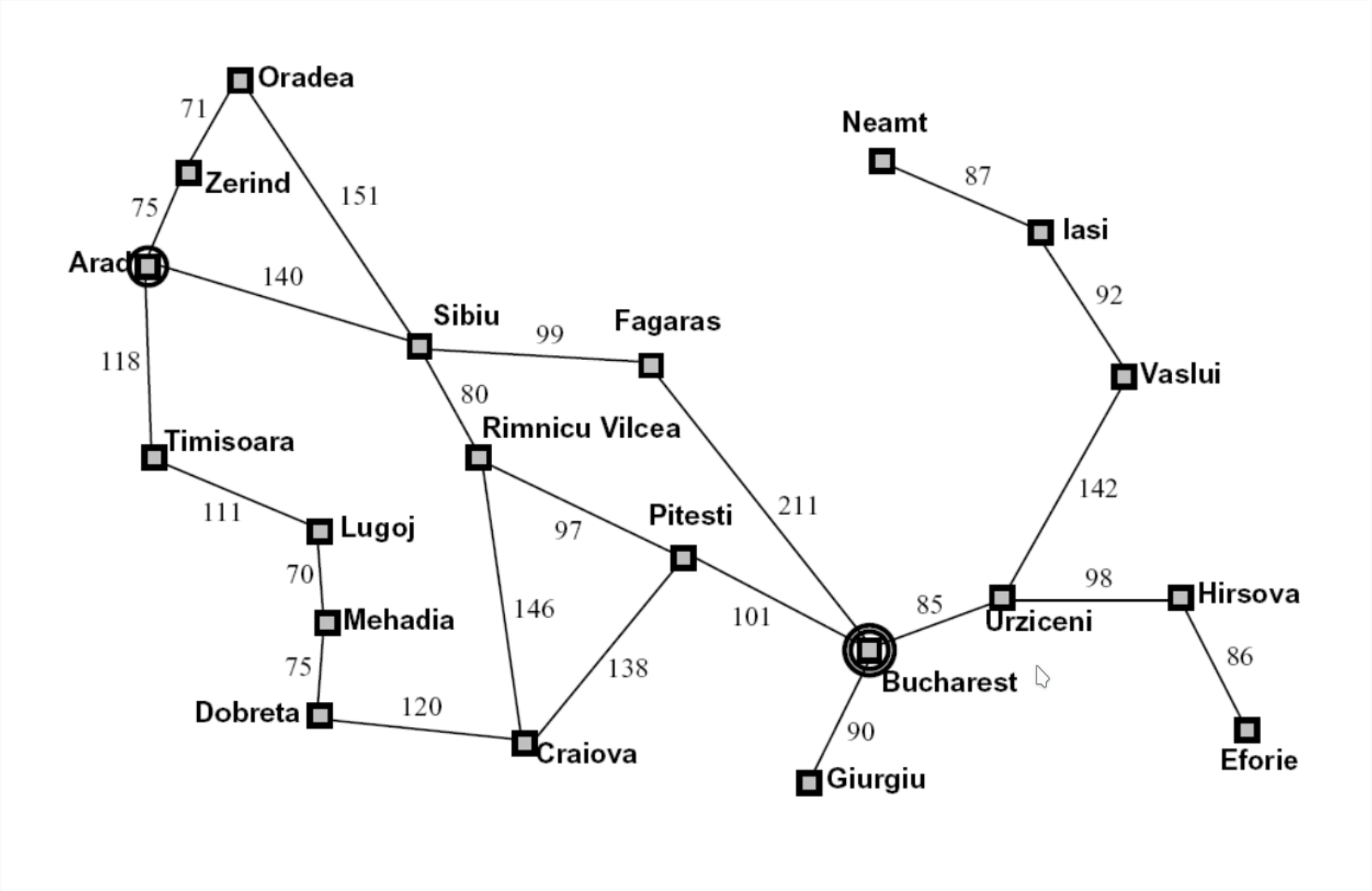
Ejercicio: Árboles de búsqueda



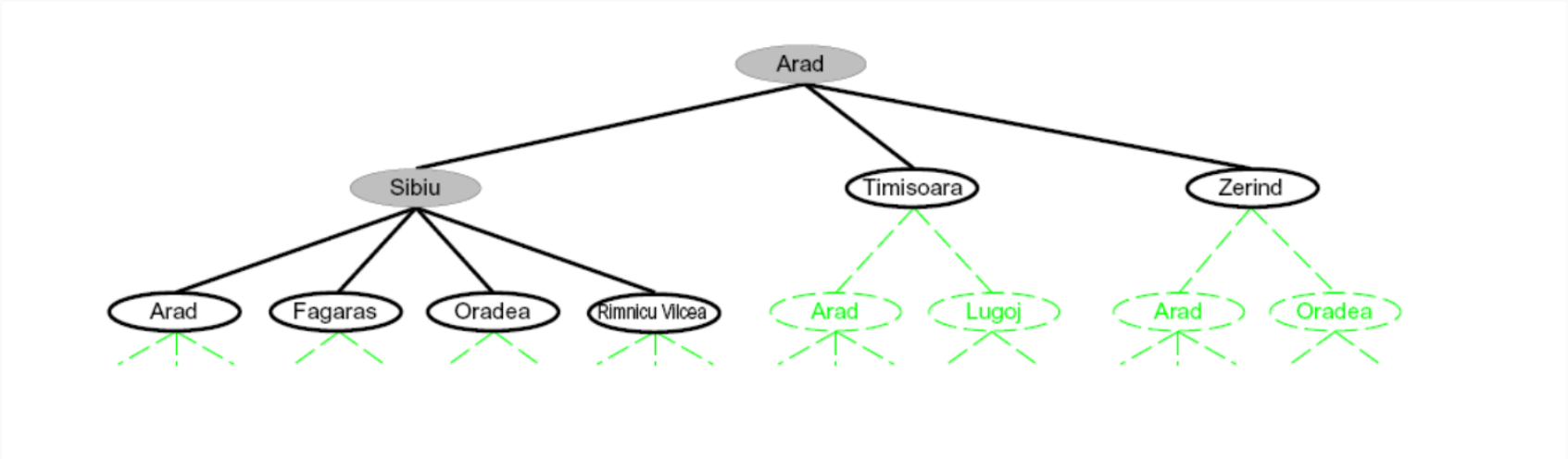
Qué tan grande es el árbol de búsqueda?

$$\infty$$

Ejemplo: Viajando por Rumania



Búsqueda de árboles



Búsqueda:

- Expandir planes potenciales.
- Mantener una frontera de planes parciales bajo consideración.
- Tratar de expandir la menor cantidad de nodos posible.

Búsqueda de árboles generalizada

```
1   def busqueda_arbol(problema, estrategia):
2       inicializa el arbol con el estado inicial.
3       while True:
4         if no hay candidatos para expandir:
5           return falla
6         seleccionar nodo para expandir
7         if nodo es objetivo:
8           return solucion
9         else:
10          expandir nodo y agregar nodos resultantes al arbol
                .
```

Ideas importantes:

- Frontera (fringe).
- Expansión de nodos.
- Extrategia de expansión.

Búsqueda de árboles generalizada

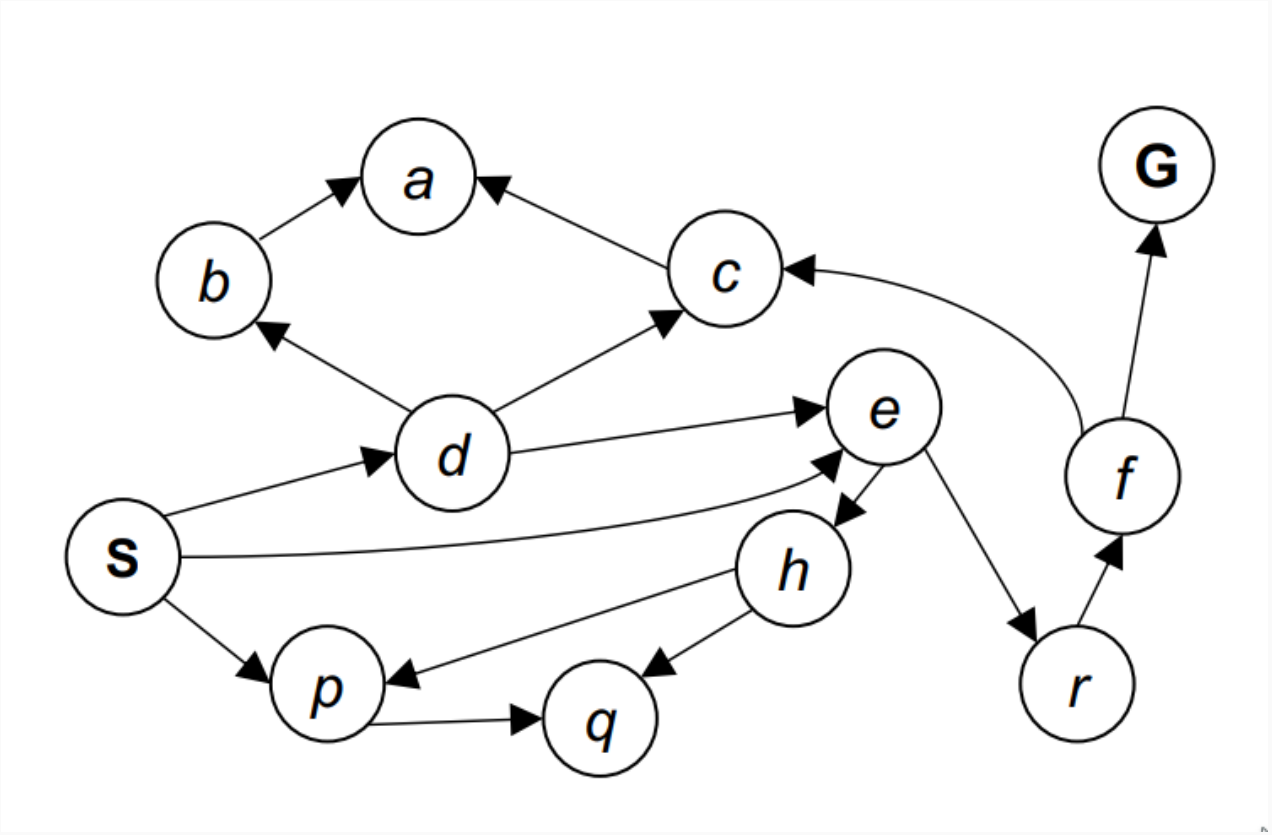
```
1   def busqueda_arbol(problema, estrategia):
2       inicializa el arbol con el estado inicial.
3       while True:
4         if no hay candidatos para expandir:
5           return falla
6         seleccionar nodo para expandir
7         if nodo es objetivo:
8           return solucion
9         else:
10          expandir nodo y agregar nodos resultantes al arbol
                .
```

Ideas importantes:

- Frontera (fringe).
- Expansión de nodos.
- Extrategia de expansión.

Pregunta principal: Cuáles nodos de la frontera expandir?

Ejemplo: Búsqueda de árboles



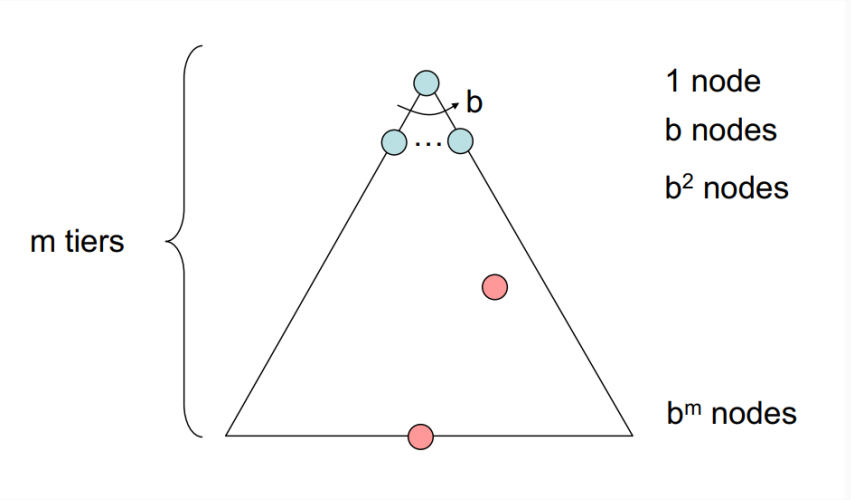
Búsqueda a Ciegas

Depth-First Search

Estrategia: Expandir el nodo con más profundidad primero.
Implementación: La frontera es una pila (LIFO).

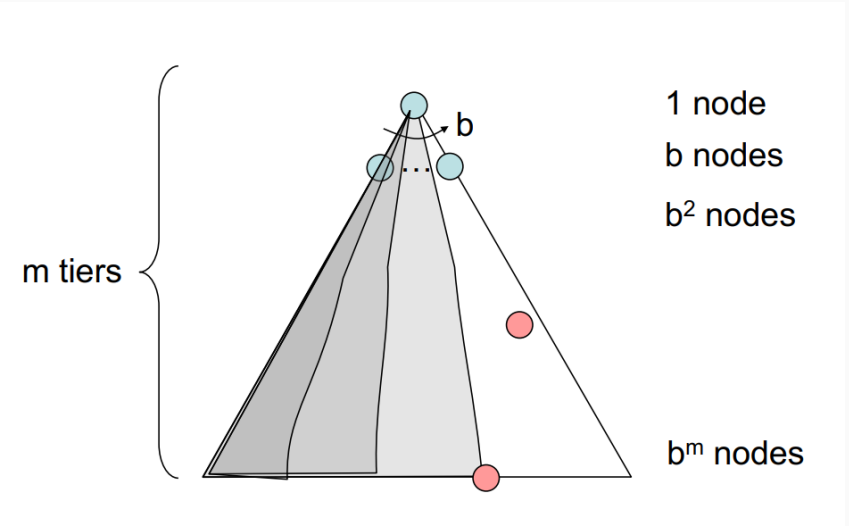
Propiedades de algoritmos de búsqueda

- Completo: Solución garantizada si existe?
- Óptimo: Garantizada la mejor solución?
- Complejidad en tiempo?
- Complejidad en espacio?
- Gráfico:
  - b es el factor de ramificación.
  - m es la profundidad máxima.
  - existen soluciones en distintas profundidades.
- Número de nodos en el árbol: $1 + b + b^2 + \cdots + b^m = O(b^m)$



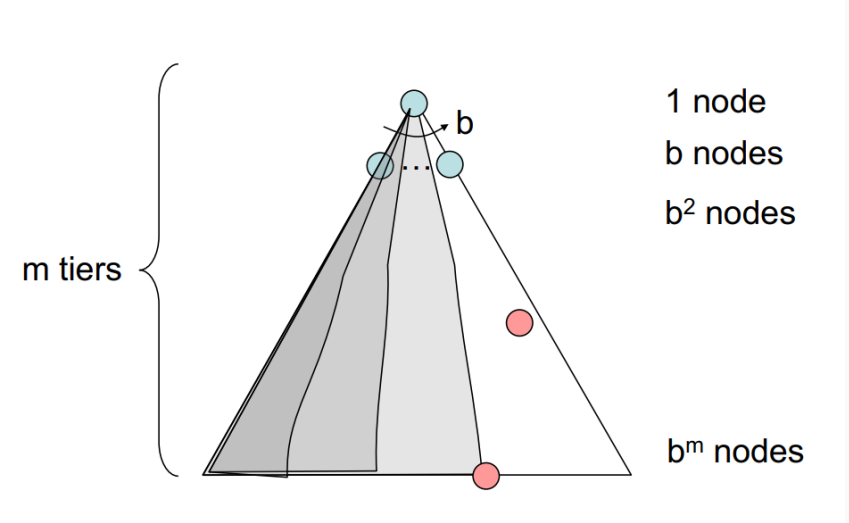
Depth-First Search: propiedades

- Cuáles nodos expande DFS?
  - Estrategia a la izquierda.
  - Puede procesar todo el árbol.
  - Si el arbol es finito, toma $O(b^m)$.
- Cuánto espacio ocupa la frontera?
  - .
- Es completo?
  - .
- Es óptimo?
  - .



Depth-First Search: propiedades

- Cuáles nodos expande DFS?
  - Estrategia a la izquierda.
  - Puede procesar todo el árbol.
  - Si el arbol es finito, toma $O(b^m)$.
- Cuánto espacio ocupa la frontera?
  - Solamente los hijos en el camino principal $O(bm)$.
- Es completo?
  - m puede ser infinito, solo si evitamos ciclos.
- Es óptimo?
  - No, encuentra la solución más cercana a la izquierda, sin importar profundidad ni costo.

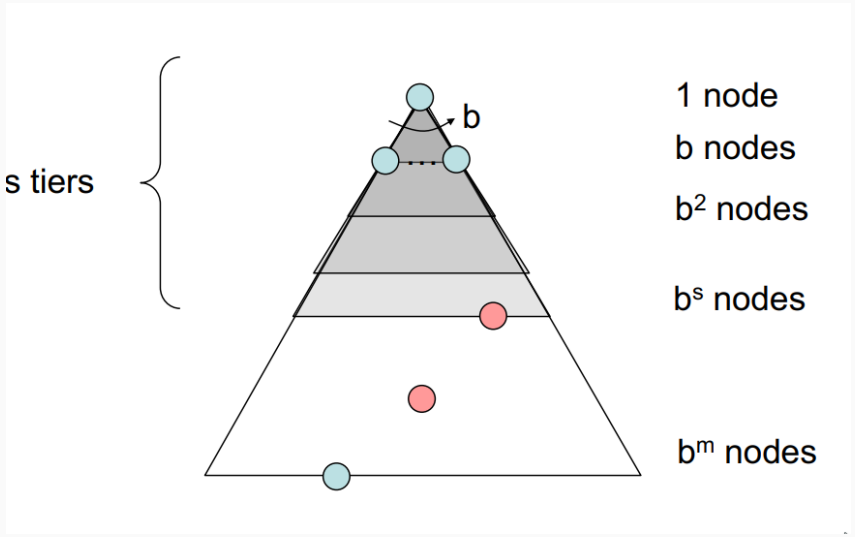


Breadth-First Search

Estrategia: Expandir el nodo con menos profundidad primero.
Implementación: La frontera es una cola (FIFO).

Breadth-First Search: propiedades

- Cuáles nodos expande BFS?
  - Los nodos más cercanos.
  - s es la profundidad de la solución más cercana.
  - La búsqueda toma $O(b^s)$ en tiempo.
- Cuánto espacio ocupa la frontera?
  - Aproximadamente el nivel final $O(b^s)$.
- Es completo?
  - s debe ser finito si existe una solución, entonces si.
- Es óptimo?
  - Solamente si todos los costos son 1.

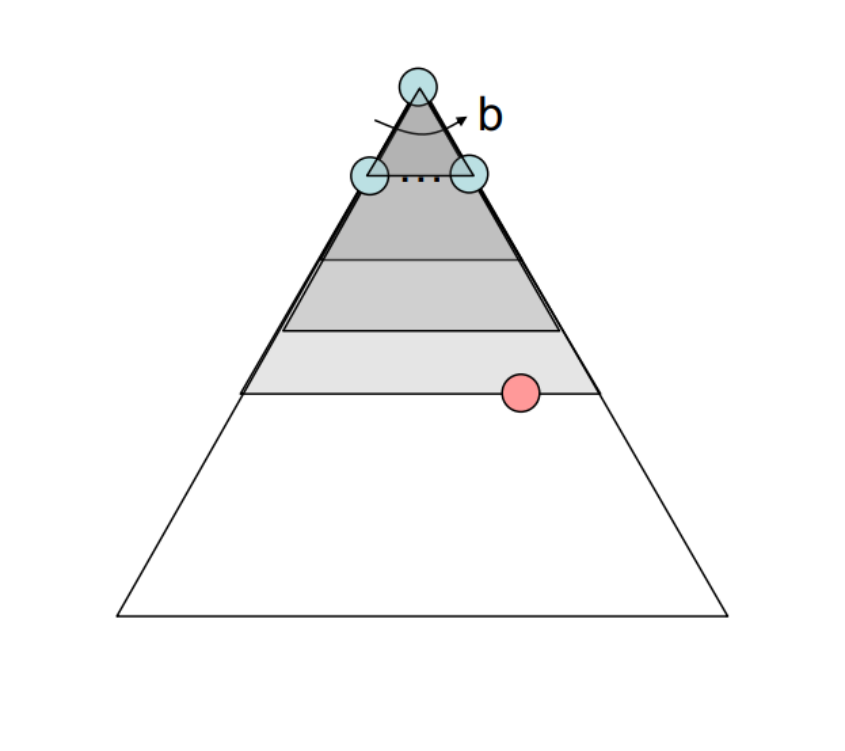


Quiz: DFS vs BFS

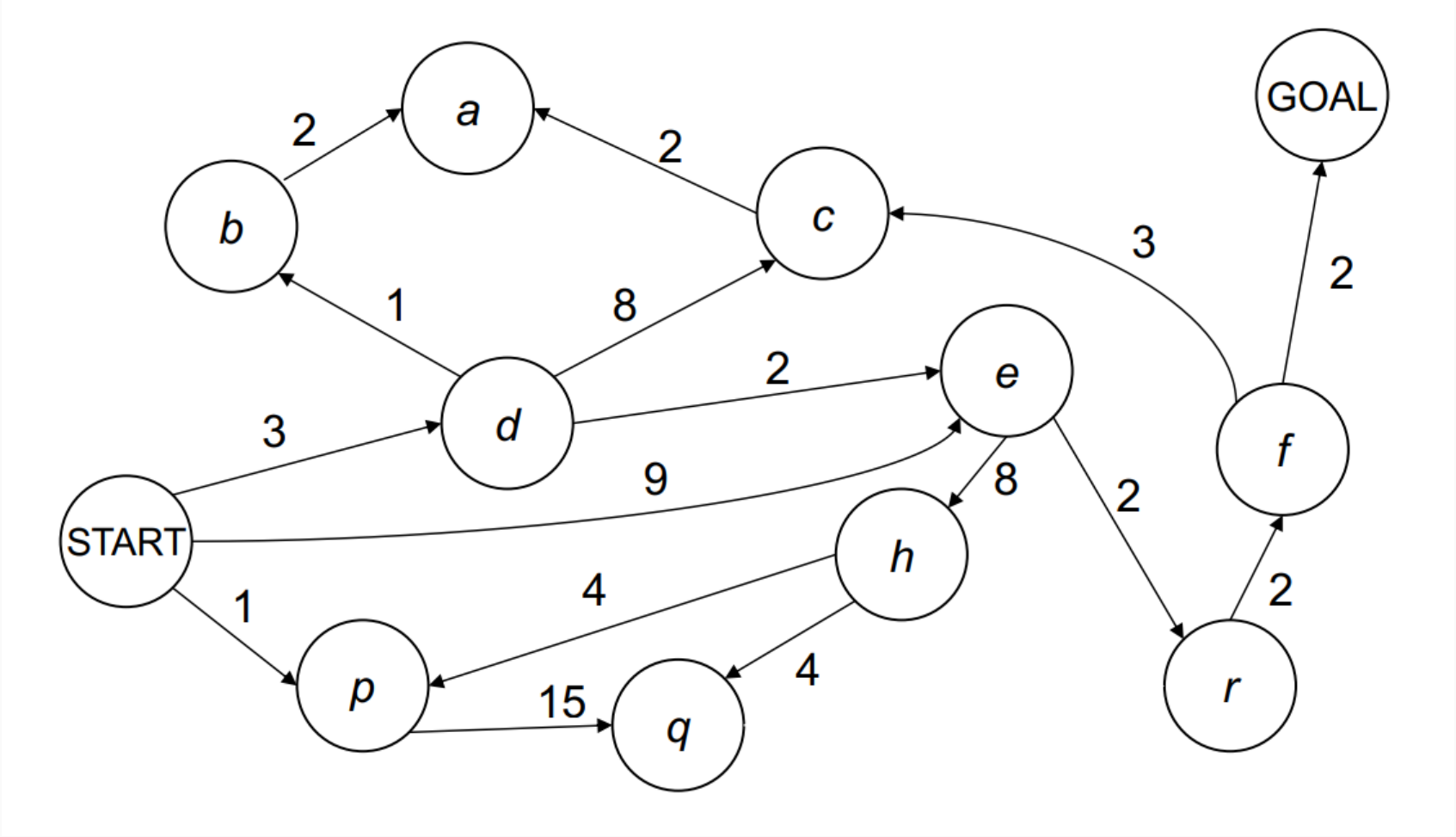
- Cuándo BFS es mejor que DFS?
  - Encuentra el camino más corto.
- Cuándo DFS es mejor que BFS?
  - Mejor aprovechamiento de la memoria.

Iterative Deepening

- Idea: aprovechar memoria usada por DFS con el tiempo y la solución corta de BFS.
  - Corre DFS con límite de profundidad 1, si no hay solución...
  - Corre DFS con límite de profundidad 2, si no hay solución...
  - Corre DFS con límite de profundidad 3...
- No es redundante?
  - Generalmente, la mayor parte del trabajo ocurre en el nivel más bajo.



Cost-sensitive search



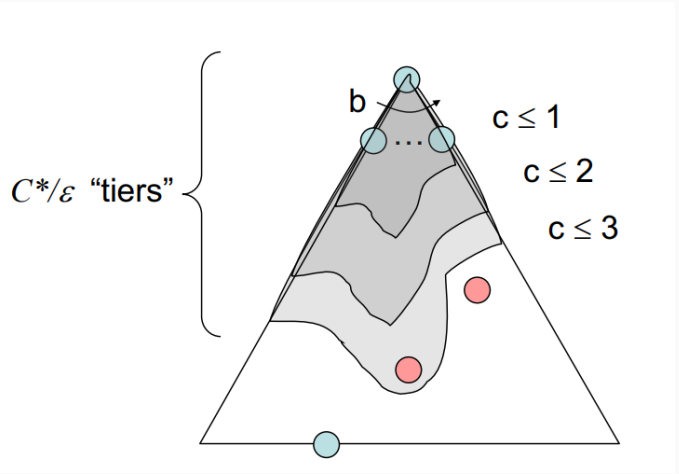
BFS encuentra el mejor camino en términos de números de acciones. No encuentra el camino con el menor costo. Ahora trataremos un algoritmo similar que encuentra el camino con el menor costo.

Uniform-Cost Search

Estrategia: Expandir el nodo con menos costo primero.
Implementación: La frontera es una cola de prioritaria (costo acumulado).

Uniform Cost Search: propiedades

- Cuáles nodos expande UCS?
  - Todos los nodos con costo $<$ solución más barata.
  - Si el costo de la solución es C^* y los arcos al menos ϵ, entonces, la "profundidad efectiva" es aprox. $\frac{C^*}{\epsilon}$.
  - La búsqueda toma $O(b^{\frac{C^*}{\epsilon}})$ en tiempo.
- Cuánto espacio ocupa la frontera?
  - Aproximadamente el nivel final $O(b^{\frac{C^*}{\epsilon}})$.
- Es completo?
  - Asumiendo que la mejor solución tiene costo finito.
- Es óptimo?
  - Si.



Uniform Cost: Detalles

- UCS explora contornos de costo creciente.
- UCS es completo y óptimo.
- Lo malo:
  - Explora opciones en “todas las direcciones”.
  - No existe información acerca de la ubicación del objetivo.

La única cola

- Todos estos algoritmos son iguales excepto por la estrategia de expansión de la frontera
  - Conceptualmente, todas las fronteras son colas prioritarias.
  - En la práctica, para DFS y BFS, se puede implementar usando pilas y colas tradicionales.
  - Se puede programar una implementación que tome como variable el objeto de cola.

Búsqueda y modelos

- Los algoritmos de búsqueda operan sobre modelos del entorno.
  - El agente no prueba todas las posibilidades en el mundo real.
  - La planificación se hace en *simulación*.
  - La búsqueda es tan buena como el modelo.

Preguntas?