Búsqueda no informada

O “Búsqueda ciega”

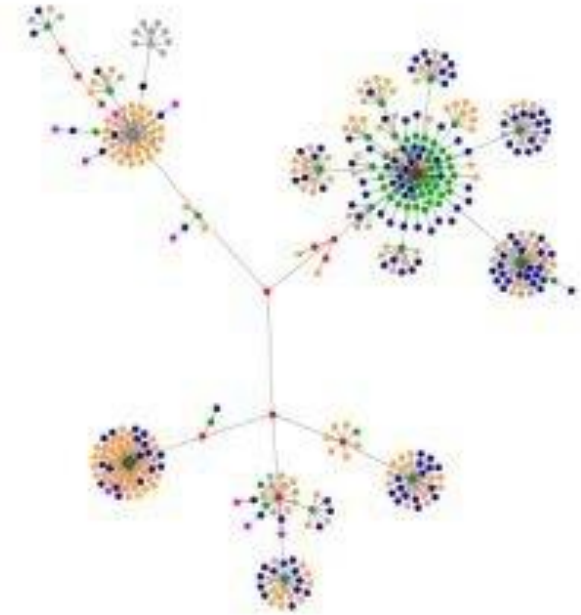
Búsqueda no informada

- Estos algoritmos comienzan en un estado inicial y aplican las acciones del agente para explorar el espacio de estados.

- El objetivo es llegar a un estado final o meta, el cual se logra al satisfacer una función de paro.

- Estos algoritmos no utilizan información adicional para guiar la búsqueda hacia el estado meta u objetivo, **únicamente información de los estados ya visitados o expandidos**.

Problema



- Espacio de estados S
  - $S = \{e_1, e_2, e_3, \dots\}$
- Estado inicial $s_0 \in S$
- Acciones del agente (Función sucesor)
  - *expandir* estado
- Función meta (o de paro):
- Parar: $S \rightarrow$ {verdadero, falso}
- $parar(s) = \begin{cases} V, & if\ s = meta \\ F, & otro\ caso \end{cases}$

Resolver el problema, consiste en encontrar la secuencia de acciones que logre esta meta.

Encontrar la ruta de un estado a otro

- Evitar caer en ciclos infinitos
- Poder recordar estados anteriores
- Usar estructuras de datos que logren esto
  - Pilas
  - Colas
  - Colas de prioridad
  - Tablas de dispersión

Búsqueda primero en profundidad (DFS)

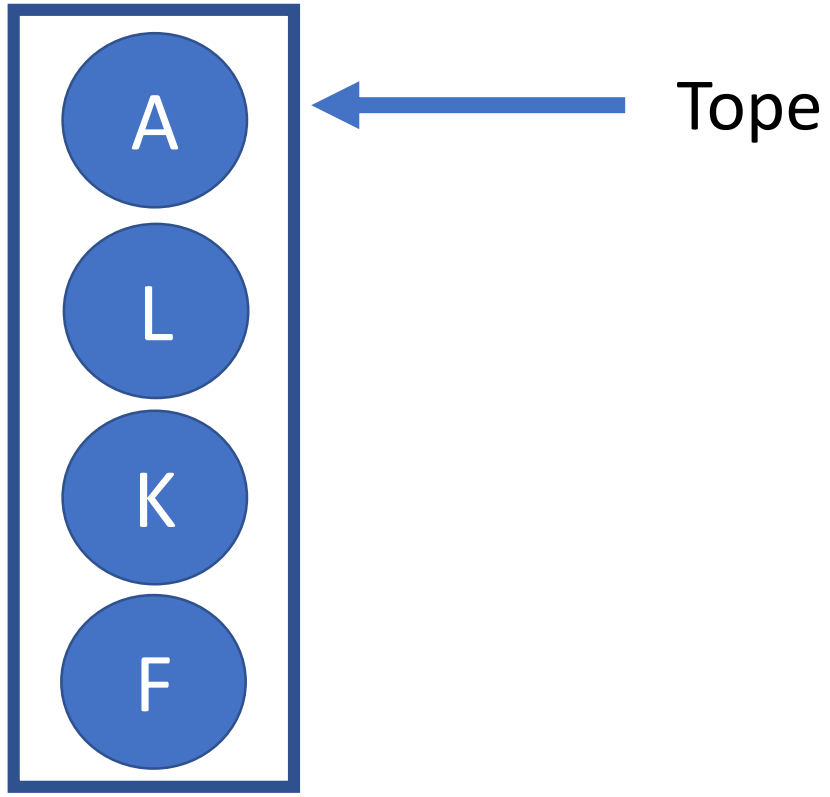
- *Depth First Search*
- Utiliza dos estructuras de datos para almacenar los estados descubiertos y para la toma de decisiones (dirección de la búsqueda):

    1. "Agenda": estados que se vayan descubriendo durante la búsqueda (frontera de búsqueda). Es una pila.
    2. Conjunto de estados expandidos o estados cerrados (un estado no puede expandirse más de una vez).

- NO se garantiza una solución óptima (número de pasos)

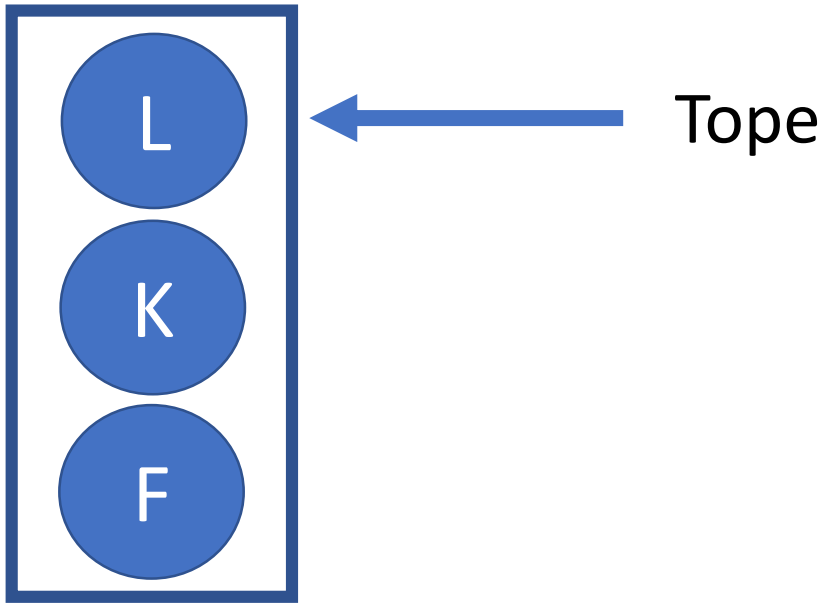
Agenda

- Pila (*stack*)
  - LIFO (*last in first out*)
  - Agregar (*push*)
  - Sacar (*pop*)



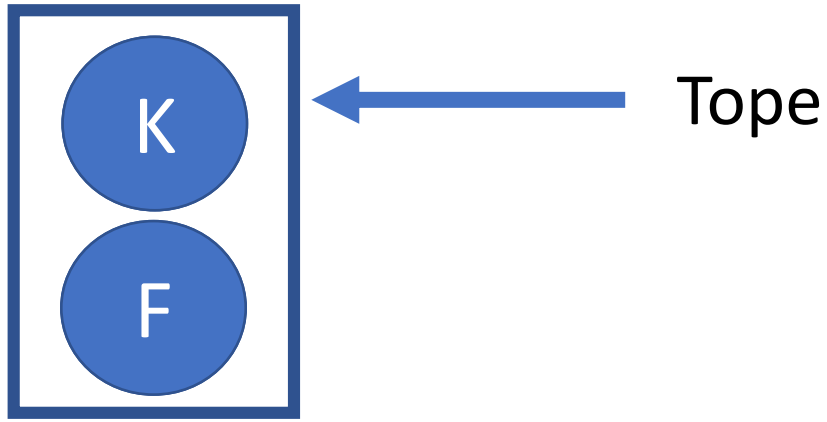
Agenda

- Pila (*stack*)
  - Agregar (*push*)
  - Sacar (*pop*)



Agenda

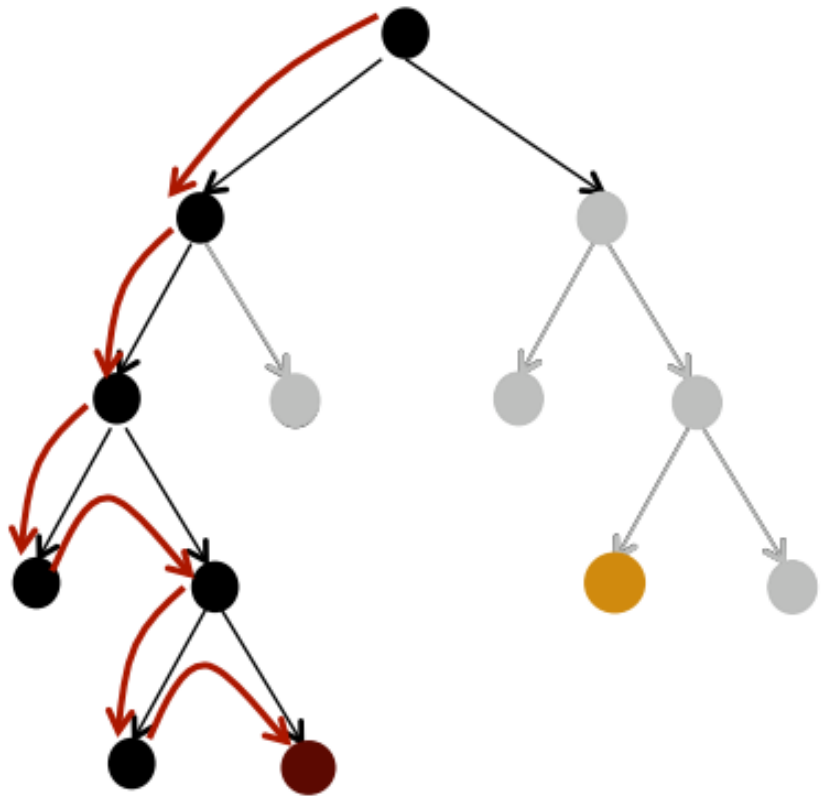
- Pila (*stack*)
  - Agregar (*push*)
  - Sacar (*pop*)



Conjunto de estados expandidos

- Conjunto basado en tabla de dispersión (hash set)
  - Eficiencia computacional
  - Verificar en tiempo constante si un estado pertenece a este conjunto.
  - La tabla de dispersión va a tardar el mismo tiempo así se tenga 1 estado o 100000000.

Búsqueda en Profundidad



Búsqueda primero en anchura (BFS)

- *Breadth First Search*
- Utiliza dos estructuras de datos para almacenar los estados descubiertos y para la toma de decisiones (dirección de la búsqueda):

    1. "Agenda": estados que se vayan descubriendo durante la búsqueda (frontera de búsqueda). Es una cola.
    2. Conjunto de estados expandidos o estados cerrados (un estado no puede expandirse más de una vez).

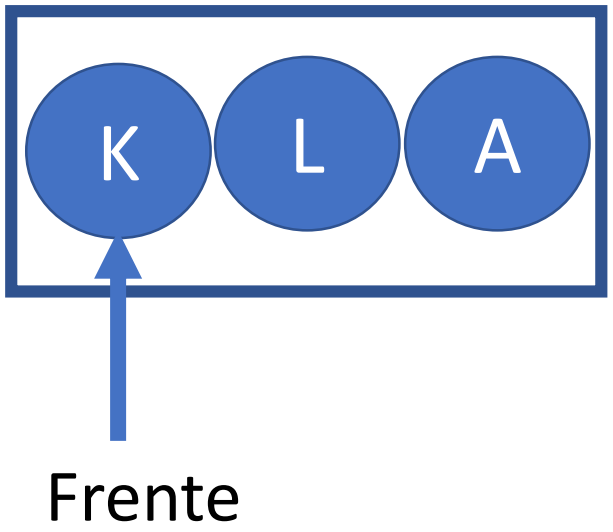
- Se garantiza una solución óptima (número de pasos)

Agenda

- Cola (*queue*)
  - FIFO (*first in first out*)
  - Agregar (*push*)
  - Sacar (*popleft*)

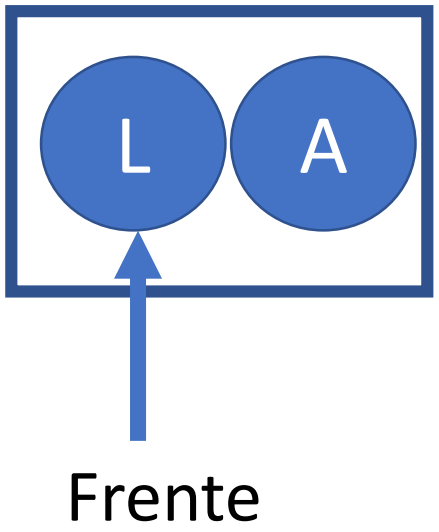
Agenda

- Cola (*queue*)
  - FIFO (*first in first out*)
  - Agregar (*push*)
  - Sacar (*popleft*)

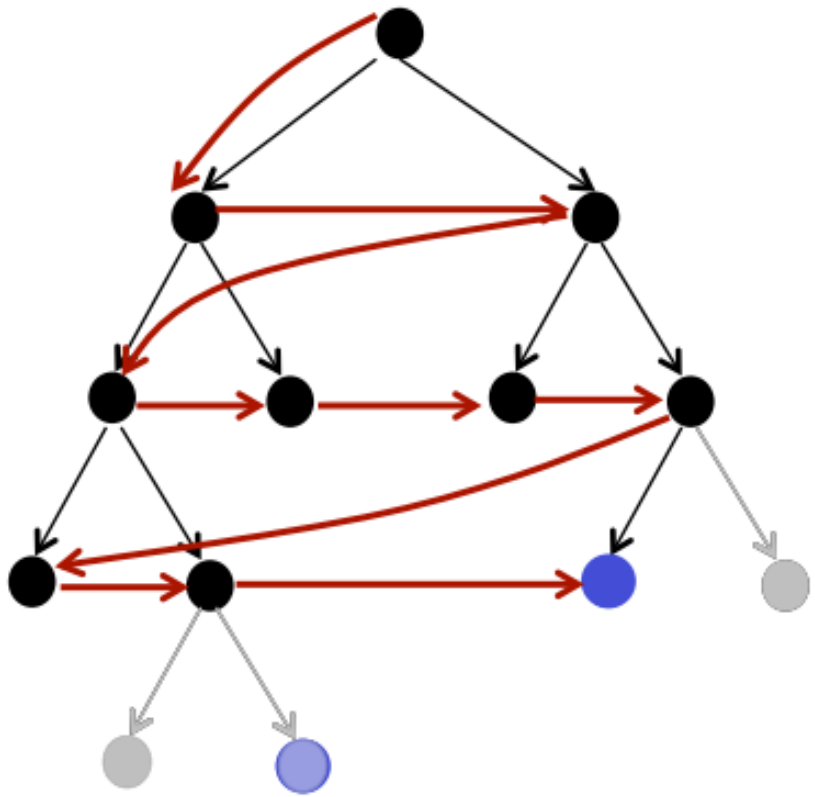


Agenda

- Cola (*queue*)
  - FIFO (*first in first out*)
  - Agregar (*push*)
  - Sacar (*popleft*)



Búsqueda en Amplitud o en Anchura



Búsqueda en profundidad limitada (DLS)

- *Depth Limited Search*
- Se introduce un límite o cota de profundidad (c). Todo estado que esté más allá de esta profundidad, va a ser ignorado por el algoritmo.
- No utiliza una lista o conjunto de expandidos.
- Se puede expandir un nodo más de una vez

Búsqueda en profundidad limitada (DLS)

- Ventajas:
  - mejora significativa de la complejidad en espacio.
  - completo para límites de profundidad adecuados (**completitud = existe solución**)
- Desventajas:
  - No es óptima: el nodo meta que se encuentra puede no ser de profundidad mínima
  - Es común que unos límites “buenos” de profundidad sólo pueden establecerse cuando el problema ya haya sido resuelto
  - Es óptima únicamente si la cota es igual a la profundidad del nodo meta (d), c = d
  - No es completo cuando c < d

Profundidad iterada (ID)

- *Iterative Deepening*
- Evita el problema de elegir una cota (c), al probar múltiples límites de profundidad.

- Estrategia:
  - Enumerar todos los límites de profundidad posibles.
  - Realizar búsqueda en profundidad limitada en cada nivel.

Profundidad iterada (ID)

Entradas:

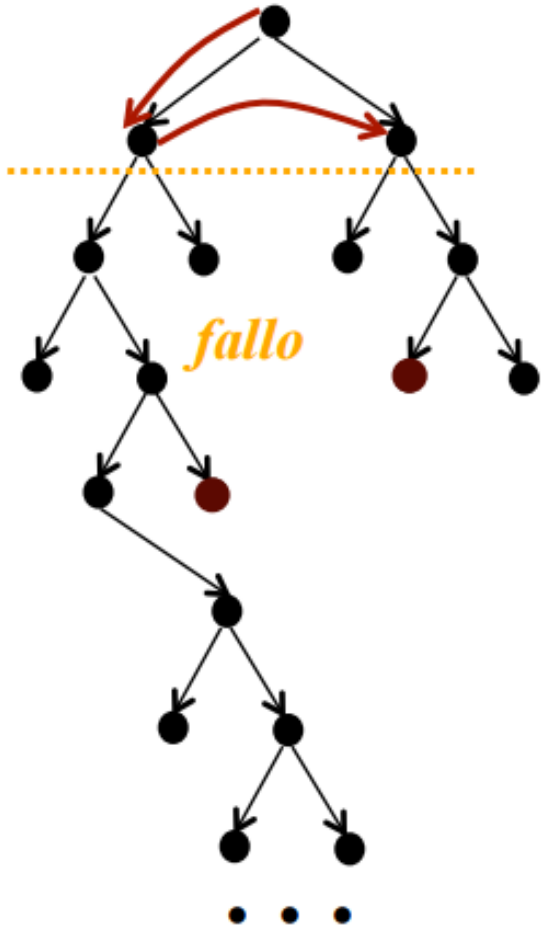
s_0: Estado Inicial

parar: Función de paro

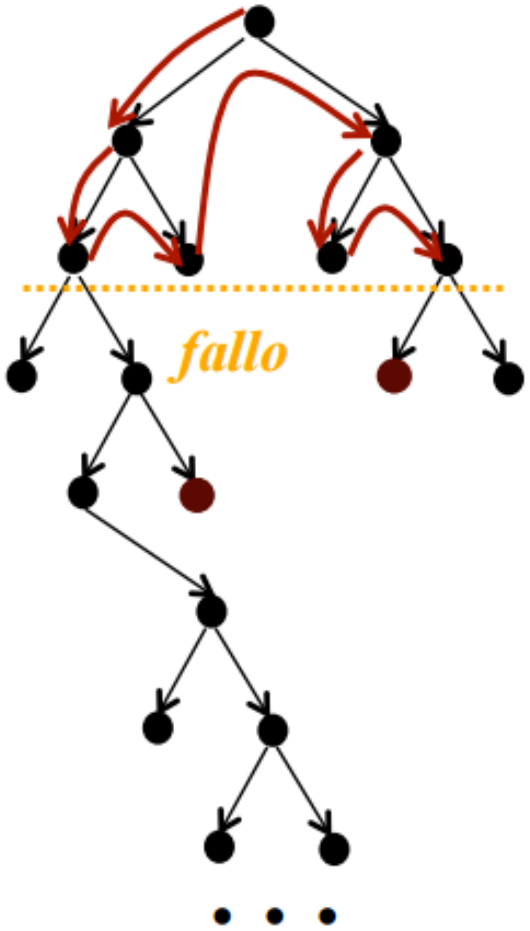
```
i ← 1
Solución ← null
    While solución = null:
        solución ← DLS(s_0 , parar, c = i)
        i ← i + 1
    return solución
```

Profundidad iterada (ID)

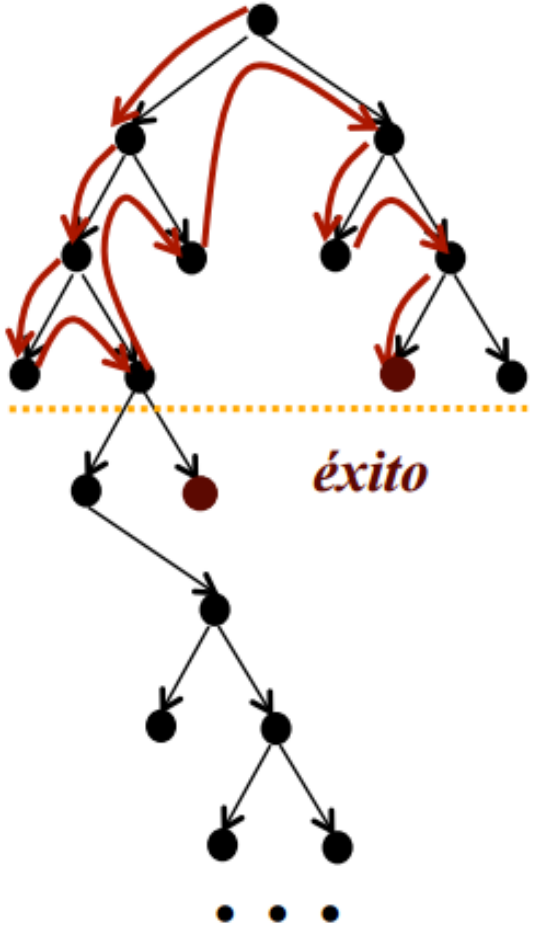
límite c = 1



límite c = 2

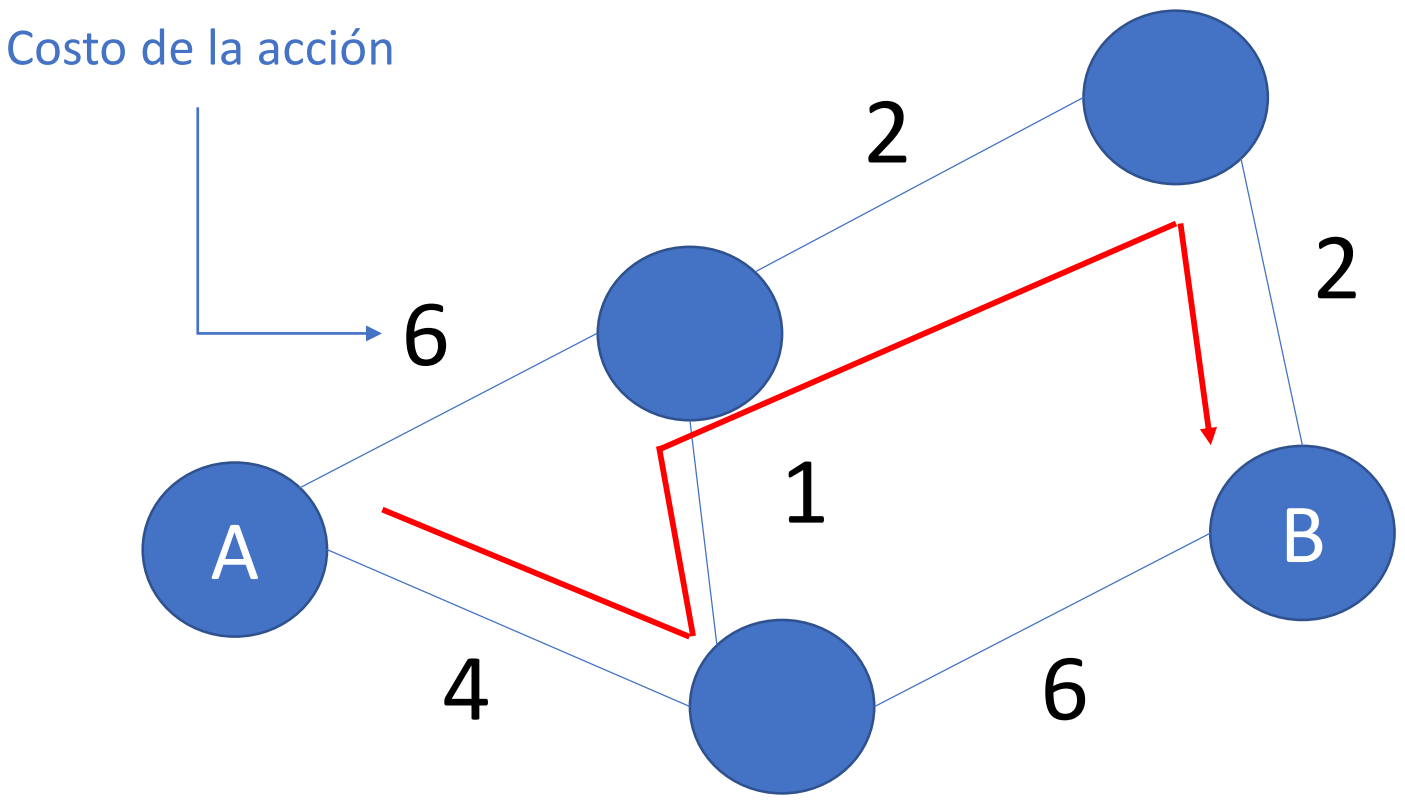


límite c = 3



Se ha considerado "optimalidad“, medida como el número de pasos que le toma al agente llegar del estado inicial al estado meta.

Se pretende generalizar este tratamiento para considerar que las acciones tienen costos, y estos costos pueden ser diferentes para cada una de ellas.



La trayectoria óptima no necesariamente es la que tiene menos pasos.

Búsqueda de costo uniforme (UCS)

- *Uniform Cost Search*
- Modificación del algoritmo BFS:
  1. La agenda utiliza una cola de prioridad. El elemento de menor costo se encuentra al frente.
  2. NO se detiene cuando se encuentra el estado meta, sino hasta que esté frente de la cola de prioridad.
- Requiere mayor memoria y toma más tiempo.
- Regresa una solución óptima.
- Es completo.

Búsqueda de costo uniforme (UCS)

- Los algoritmos de **búsqueda no informada** pueden consumir muchos recursos computacionales.
  - Especialmente para soluciones óptimas.

- Aplicables a problemas con grafos pequeños.

- Para problemas más grandes, introducir conocimiento del dominio del problema.
  - Algoritmos de búsqueda informada.