Razonamiento y Planificación Automática

César Augusto Guzmán Álvarez

Doctor en Inteligencia Artificial

Tema 5 : Búsqueda offline

Sesión 2 / 2

unir LA UNIVERSIDAD EN INTERNET

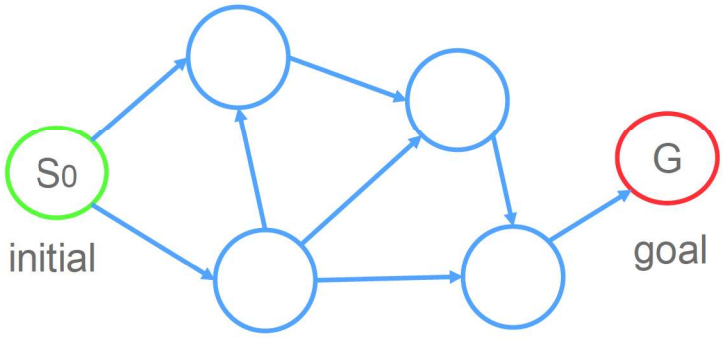
Universidad Internacional de La Rioja

Resumen – Tema anterior

Tema 5 : Búsqueda offline

Sesión 1 :

- Agentes basados en búsqueda
- Búsqueda offline
- Búsqueda en amplitud



Índice

Sesión 1 :

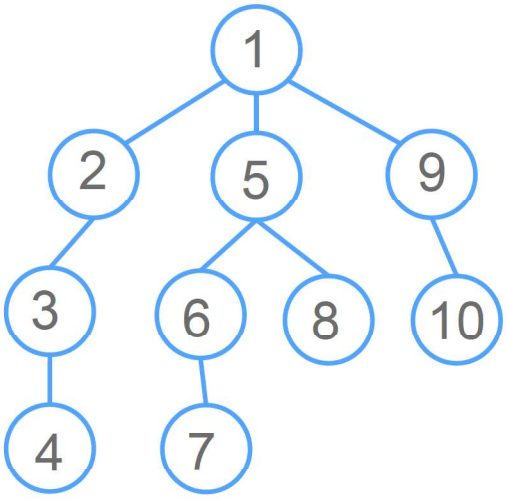
- Agentes basados en búsqueda
- Búsqueda offline
- Búsqueda en amplitud

Sesión 2 :

- Búsqueda en profundidad
- Búsqueda de coste uniforme
- Practica – DFS and BFS

Búsqueda en profundidad

depth-first search (DFS)



- Algoritmo de búsqueda sin información
- Completo y no es óptimo.
- Complejidad computacional y espacial:

  Peor caso computacional:

  $O(|V|+|E|) = O(b^d)$

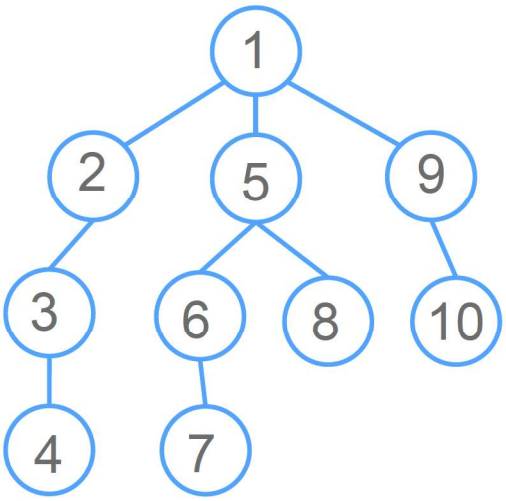
  Peor caso espacial:

  $O(V) = O(bd)$

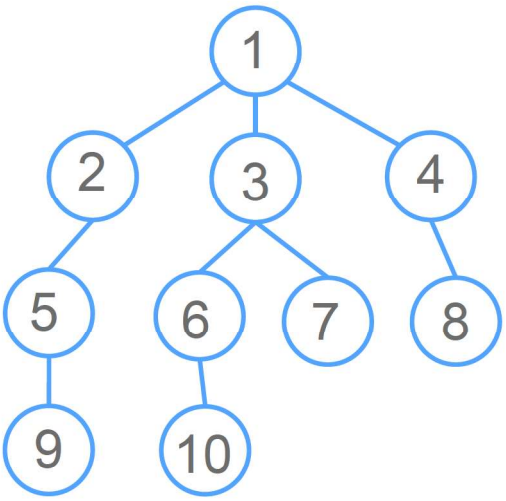
Búsqueda en profundidad

depth-first search (DFS) vs breadth-first search (BFS)

DFS

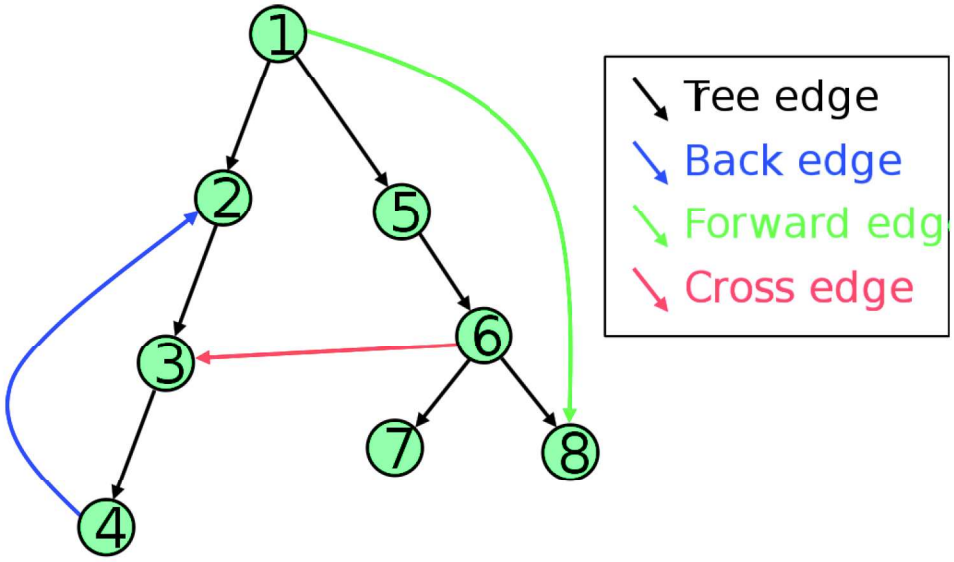


BFS



Búsqueda en profundidad

depth-first search – type of edges



Fuente: https://en.wikipedia.org/wiki/Depth-first_search#/media/File:Tree_edges.svg

Búsqueda en profundidad

depth-first search (DFS) - algoritmo

```
Input : Grafo, estado inicial S0, estado final G
1:  definir pilaAbierta como pila
2:  pilaAbierta.push(S0)
3:  mientras pilaAbierta ≠ ∅
4:      nodo = pilaAbierta.pop()
5:      si G ⊆ nodo entonces
6:              retornar nodo
7:      fin si
8:      si nodo no está marcado como visitado:
9:              marcar nodo como visitado
10:             sucesores <- Grafo.hijos(nodo)
11:             para cada sucesor ∈ sucesores hacer
12:                     sucesor.padre = nodo
13:                     pilaAbierta.push(sucesor)
14:             fin para
15:     fin si
16: fin mientras
```

Características principales:

- Similar a la implementación en anchura
  - No recursiva.
- **Difiere en :**
  - Utiliza una pila
  - Verifica que un nodo ha sido visitado hasta que lo saca de la pila

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp. 540–549.

Goodrich, Michael T.; Tamassia, Roberto (2001), Algorithm Design: Foundations, Analysis, and Internet Examples, Wiley, ISBN 0-471-38365-1

Búsqueda en profundidad

depth first search (DFS) – algoritmo recursivo

```
1: funcion DFS(G, nodo):
2:     marcar nodo como visitado
3:     sucesores <- Grafo.hijos(nodo)
4:     para cada sucesor en sucesores hacer
5:         si sucesor no está marcado como visitado entonces
6:             sucesor.padre = nodo
7:             recursivamente llamar DFS(G, sucesor)
```

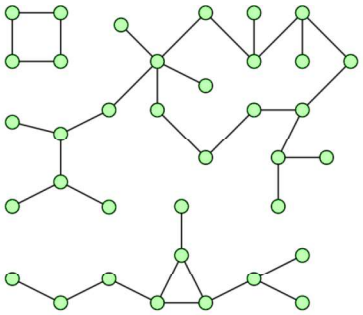
Retorna todos los vértices marcados como visitados desde nodo.

Fuente: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp. 540–549.

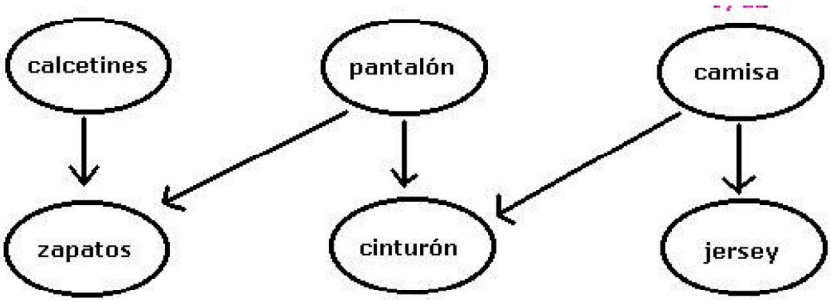
Búsqueda en profundidad

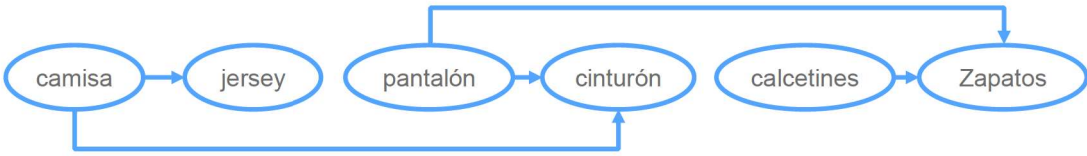
depth first search (DFS) - aplicación

Encontrar componentes conectados



Ordenamiento topológico





Índice

Sesión 1 :

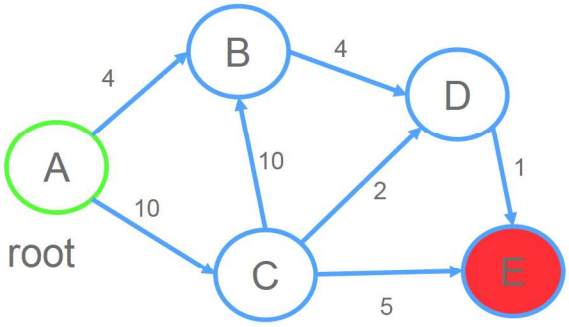
- Agentes basados en búsqueda
- Búsqueda offline
- Búsqueda en amplitud

Sesión 2 :

- Búsqueda en profundidad
- **Búsqueda de coste uniforme**

Búsqueda de coste uniforme

Uniform cost search (UCS)



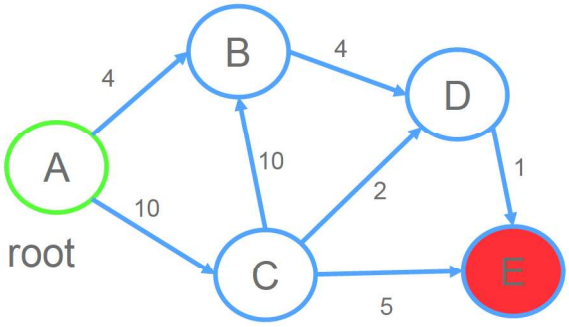
- algoritmo de búsqueda sin información
- Camino de costo mínimo.
- Completo y optimo.

Número de pasos 2, Costo = 15

Número de pasos 3, Costo = 9

Búsqueda de coste uniforme

Uniform cost search (UCS)



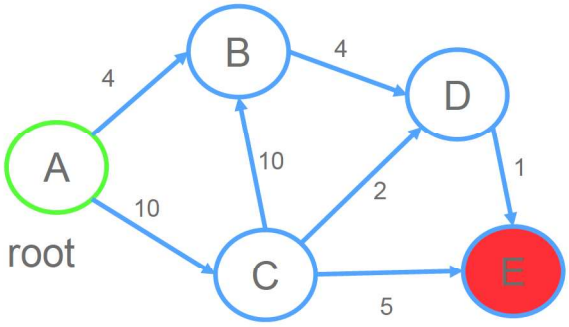
1. Cola de prioridades
2. Se detiene cuando el estado objetivo está al frente de la cola de prioridades.

Búsqueda de coste uniforme

Uniform cost search (UCS)

1. Cola de prioridades
2. Se detiene cuando el estado objetivo está al frente de la cola de prioridades.

f(n) = g(n)



priority queue

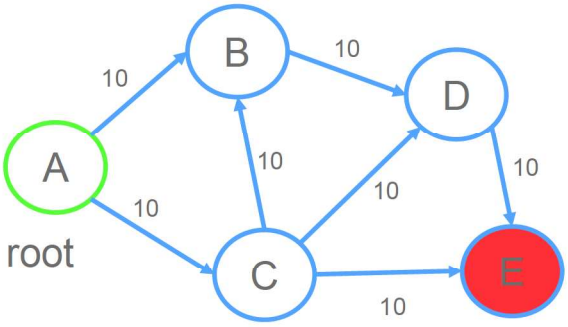
A,0	C,10 B,4	C,10 D,8	C,10 E,9

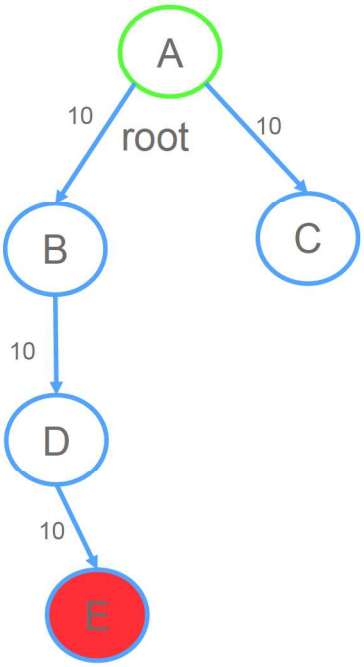
explored

	A	A,B	A,B,D

Búsqueda de coste uniforme

Uniform cost search (UCS)





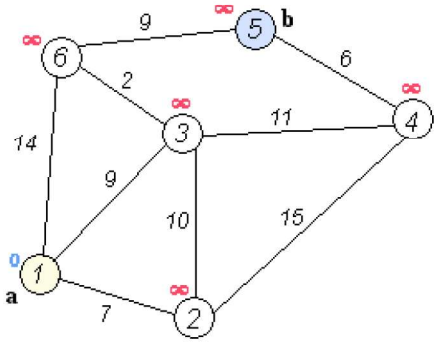
Búsqueda de coste uniforme

Algoritmo

```
Input : Grafo, estado inicial S0, estado final G
definir colaAbierta como cola prioridad
colaAbierta.add(S0)
mientras colaAbierta ≠ ∅
    nodo = colaAbierta.pop()
    si G ⊆ nodo entonces
        retornar camino a nodo
    fin si
    marcar nodo como visitado
    sucesores <- Grafo.hijos(nodo)
    para cada sucesor ∈ sucesores hacer
        si sucesor no está marcado como visitado:
            si sucesor no está en colaAbierta:
                sucesor.padre = nodo
                colaAbierta.add(sucesor)
            sino :
                reemplazar sucesor si el coste es menor
            fin si
        fin si
    fin para
fin mientras
```

Características principales:

- No recursiva.
- Utiliza una cola de prioridades
- Primero verifica si nodo es no procesado.
- Marcar como visitado:
  a) guardar en otro conjunto
  b) utilizar un atributo del nodo.



Practica – DFS and BFS

Open Labyrinth

Gracias!