Search Problems

* Inicial state
* Actions
* transition model
* goal test
* Path cost funtion

} Optimal solution

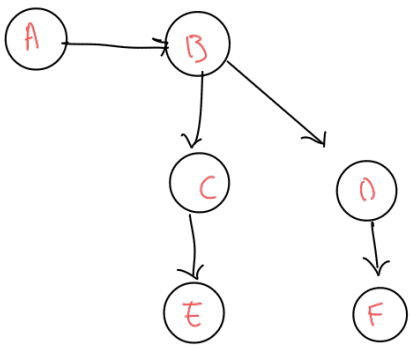
Nodo

a data structure that keeps track of

* a state
* a parent (node that generated this node)
* an action (action applied to parent to get node)
* a path cost (from initial state to node)

Depth- first Search

empezando en A, meta es F



Frontera
A

Iniciar con una frontera que contiene el estado inicial

* Repetir
- Si la frontera esta vacia no hay solucion
- Remove a node from the frontier
- if node contains goal state, return the solution
- Expande node, add resulting nodes to the frontier

Frontera
No vacio

Frontera

F
No goal

F
B

Repetimos el algoritmo, ahora B es el initial state

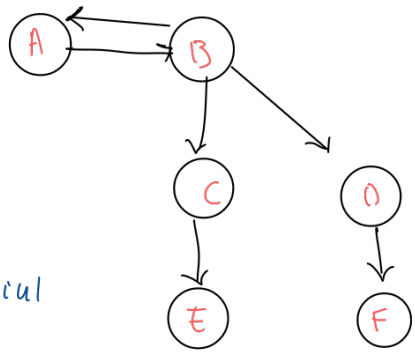
F
No vacia

F

F
No goal

F
C, D

Ahora partimos del hecho que el nodo puede regresar



debemos considerar la opcion de caer en un bucle entre A y B, o con algun otro nodo

* Iniciar con una frontera que tenga el estado inicial
* Iniciar con un explorador vacio

Repetir:

* Si la frontera esta vacia, no hay solucion
* Remover el nodo de la frontera
* Si el nodo contiene la meta, regresar la solucion

Frontier	Explored Set
A	
	A
B	A
C D	A B

Front	explore set
C	A B D
C F	A B D
C	A B D F
	A B D F C

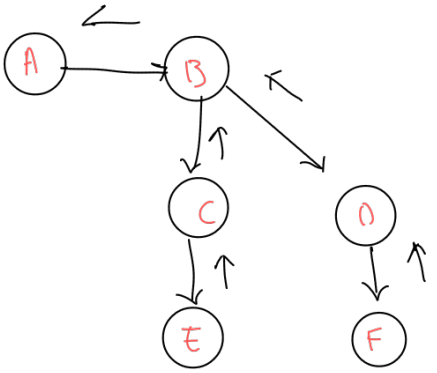
Last input first output → Lifo

Breadth - First Search

Search algorithm that always expand the shallowest node in the frontier

Queue → first-in- first out data type

empezando en A



Frontier: A
explored set:

→

Frontier : empty
explored set: A

↓

Frontier: B
explored set: A

Se explora B (comparar si es la meta)

↓

Frontier: C D
explored set: A B

↓

Frontier: D
explored set: A B C

←

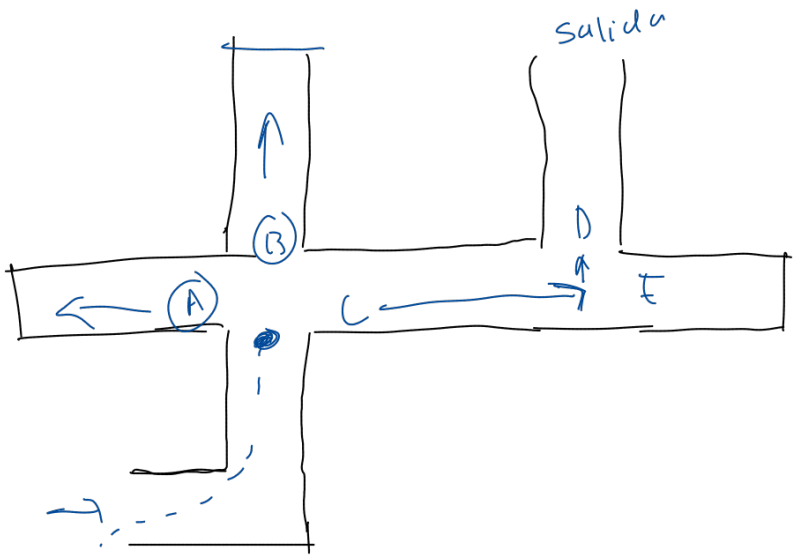
Frontier: D E
explored set: A B C

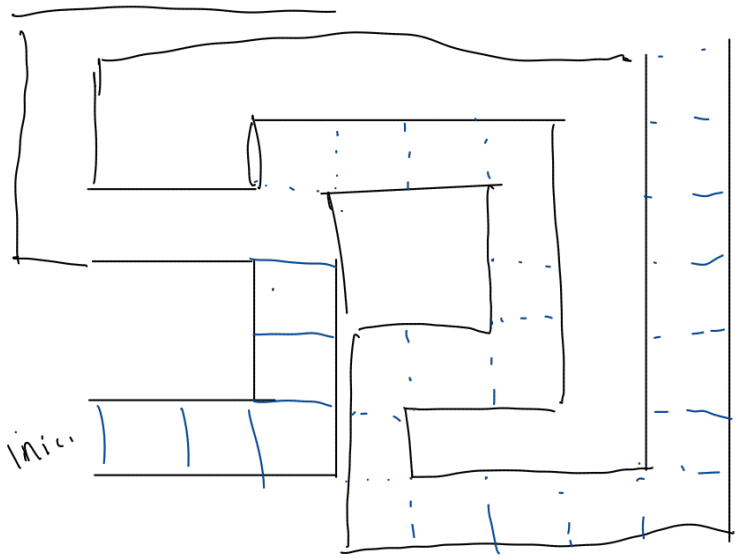
↓

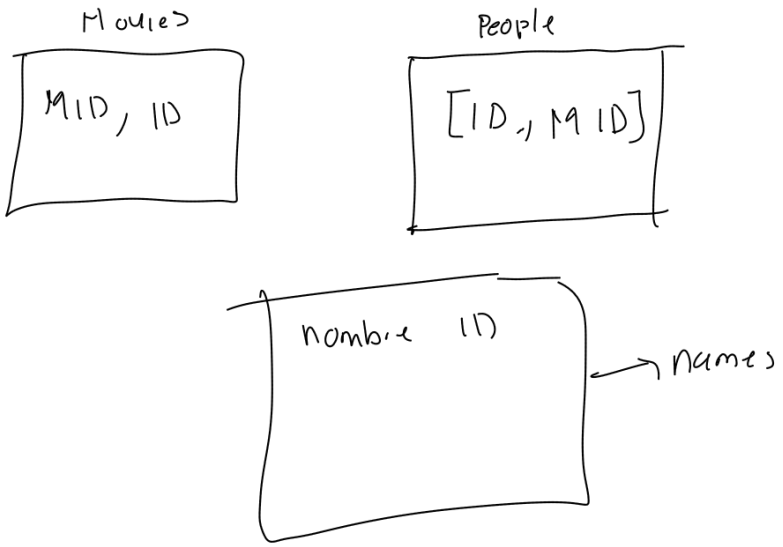
Frontier: E F
explored set: A B C D

→

Frontier : F
explored set: A B C D E






Movies
MID, ID
People
[ID, MID]
nombre ID
names