UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

Paradigmas de Programación

Unidad 3

Problema Paradigma Lógico

¡A trabajar!

Contenido

Problema 1: Star Wars

Problema 2: Quicksort

Problema 3: Grafos

Problema 1: Star Wars

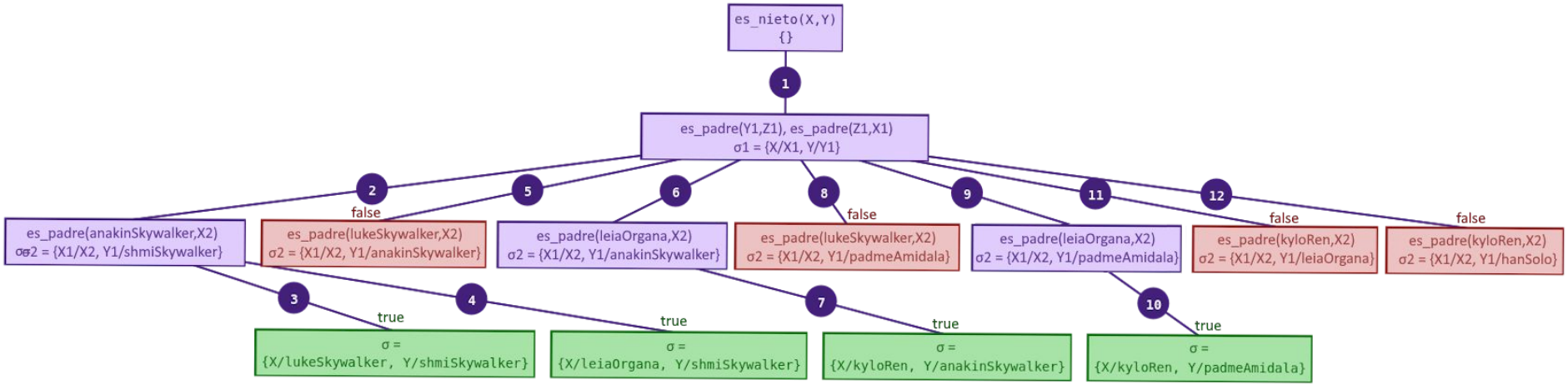
Problema 1: Star Wars

Supongamos que nos proponemos identificar los parentescos entre todos los personajes de la saga Star Wars.


SPOILER
ALERT

```
es_padre(shmiSkywalker, anakinSkywalker).
es_padre(anakinSkywalker, lukeSkywalker).
es_padre(anakinSkywalker, leiaOrgana).
es_padre(padmeAmidala, lukeSkywalker).
es_padre(padmeAmidala, leiaOrgana).
es_padre(leiaOrgana, kyloRen).
es_padre(hanSolo, kyloRen).

es_hermano(X,Y):-es_padre(Z,X),es_padre(Z,Y),X\==Y.
es_primo(X,Y):- es_padre(Z, Y), es_padre(W, X), es_hermano(W, Z).
es_nieto(X,Y):- es_padre(Y,Z),es_padre(Z,X).

es_descendiente(X,Y):- es_padre(Y,X).
es_descendiente(X,Y):- es_padre(Y,Z), es_descendiente(X,Z).
```

es_nieto(X, Y).


es_nieto(X,Y)
{}
1
es_padre(Y1,Z1), es_padre(Z1,X1)
σ1 = {X/X1, Y/Y1}
2
es_padre(anakinSkywalker,X2)
σσ2 = {X1/X2, Y1/shmiSkywalker}
false
5
es_padre(lukeSkywalker,X2)
σ2 = {X1/X2, Y1/anakinSkywalker}
6
es_padre(leiaOrgana,X2)
σ2 = {X1/X2, Y1/anakinSkywalker}
8
false
es_padre(lukeSkywalker,X2)
σ2 = {X1/X2, Y1/padmeAmidala}
9
es_padre(leiaOrgana,X2)
σ2 = {X1/X2, Y1/padmeAmidala}
11
false
es_padre(kyloRen,X2)
σ2 = {X1/X2, Y1/leiaOrgana}
12
false
es_padre(kyloRen,X2)
σ2 = {X1/X2, Y1/hanSolo}
3
true
σ =
{X/lukeSkywalker, Y/shmiSkywalker}
4
true
σ =
{X/leiaOrgana, Y/shmiSkywalker}
7
true
σ =
{X/kyloRen, Y/anakinSkywalker}
10
true
σ =
{X/kyloRen, Y/padmeAmidala}

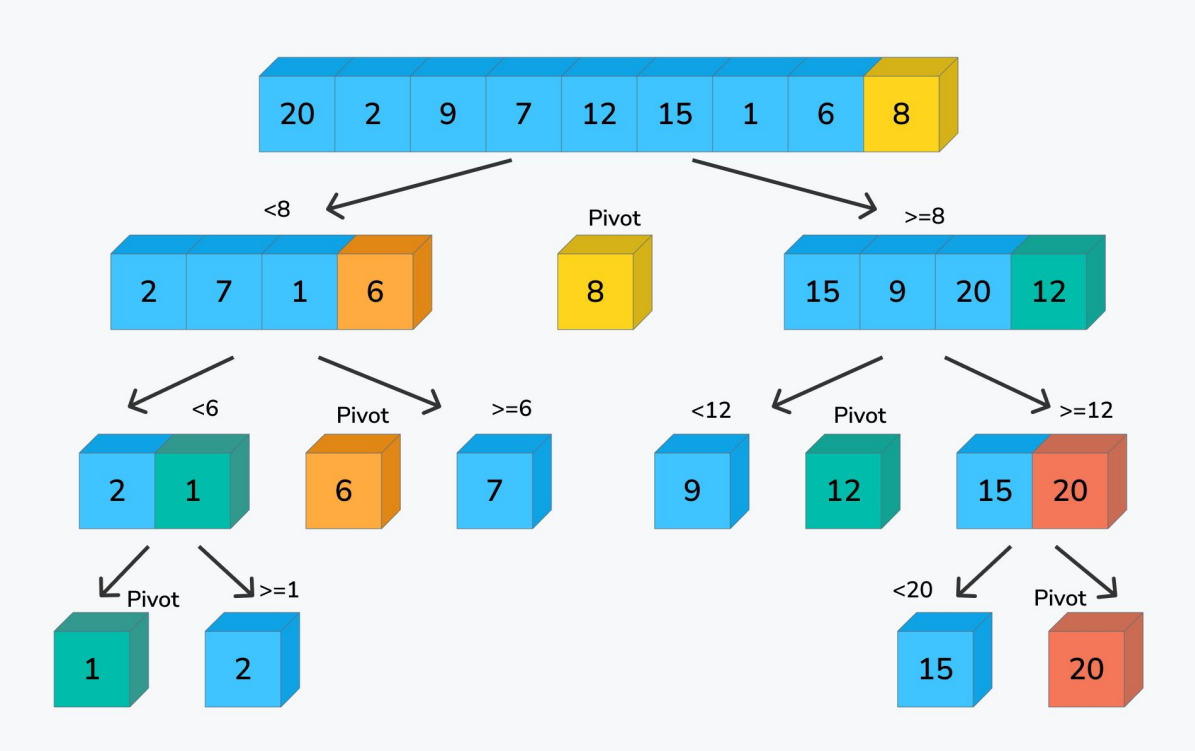
¡Usen Trace!

trace: comando para debuggear en Prolog. Documentación: https://www.swi-prolog.org/pldoc/man?section=trace-example

```
2 ?- trace.
true.

[trace] 2 ?- es_nieto(X,Y).
   Call: (10) es_nieto(_6936, _6938) ? creep
   Call: (11) es_padre(_6938, _8262) ? creep
   Exit: (11) es_padre(shmiSkywalker, anakinSkywalker) ? creep
   Call: (11) es_padre(anakinSkywalker, _6936) ? creep
   Exit: (11) es_padre(anakinSkywalker, lukeSkywalker) ? creep
   Exit: (10) es_nieto(lukeSkywalker, shmiSkywalker) ? creep
X = lukeSkywalker,
Y = shmiSkywalker .
```

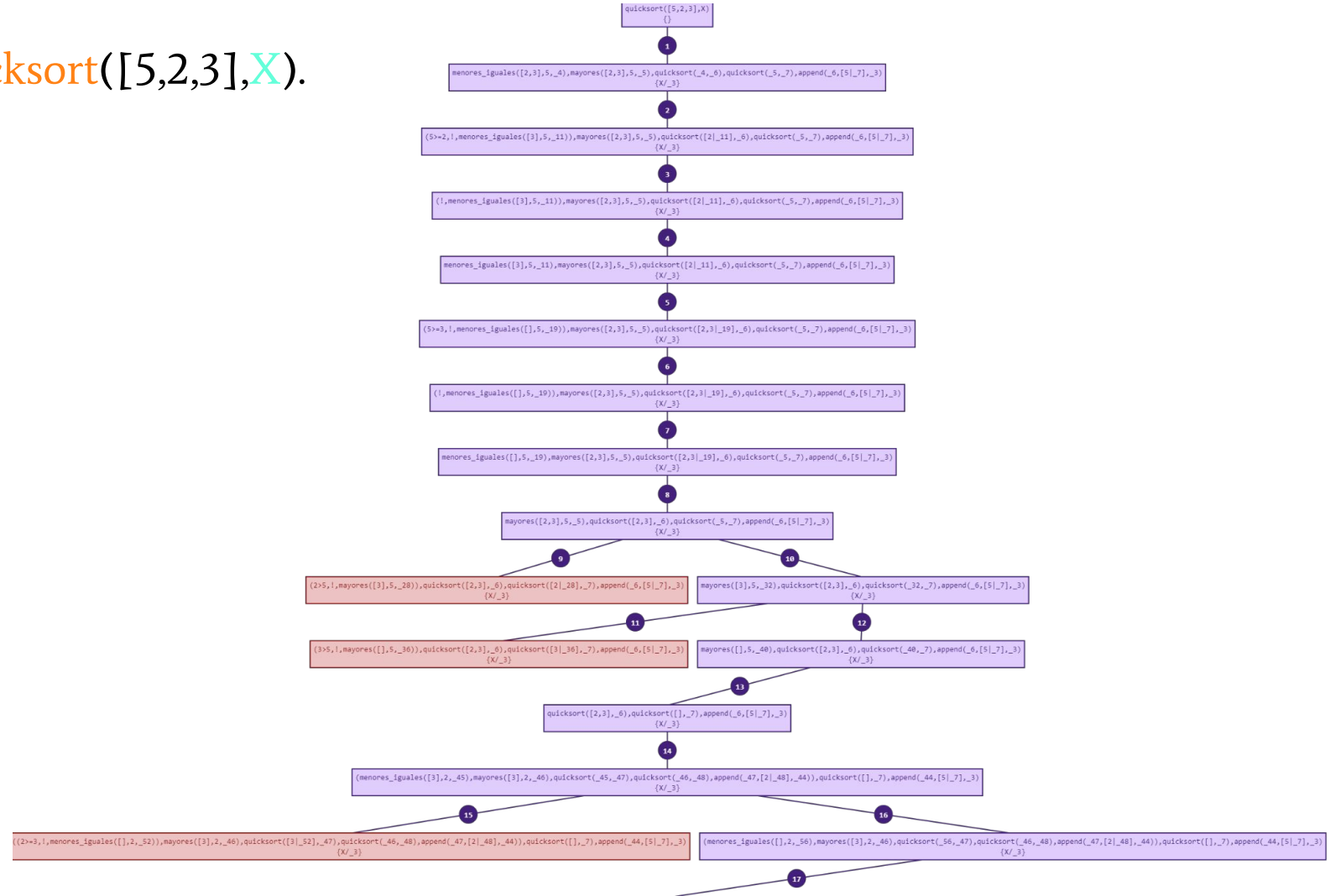
Problema 2: Quicksort

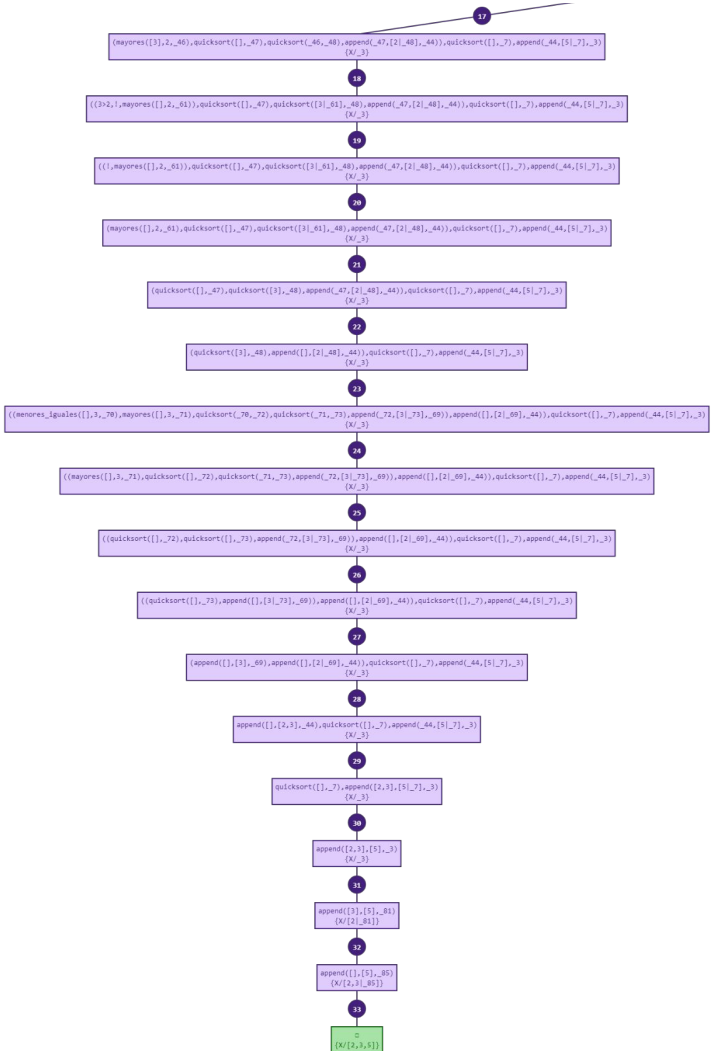
Problema 2: Quicksort



```
menores_iguales([],_,[]).
menores_iguales([H|T],X,[H|L]):- X >= H, !, menores_iguales(T,X,L).
menores_iguales([_|T],X,L):- menores_iguales(T,X,L).

mayores([],_,[]).
mayores([H|T],X,[H|L]):- H > X, !, mayores(T,X,L).
mayores([_|T],X,L):- mayores(T,X,L).

quicksort([],[]).
quicksort([H|T],L):- menores_iguales(T,H,L1), mayores(T,H,L2), quicksort(L1,Q1),
    quicksort(L2,Q2), append(Q1, [H|Q2], L).
```

quicksort([5,2,3],X).

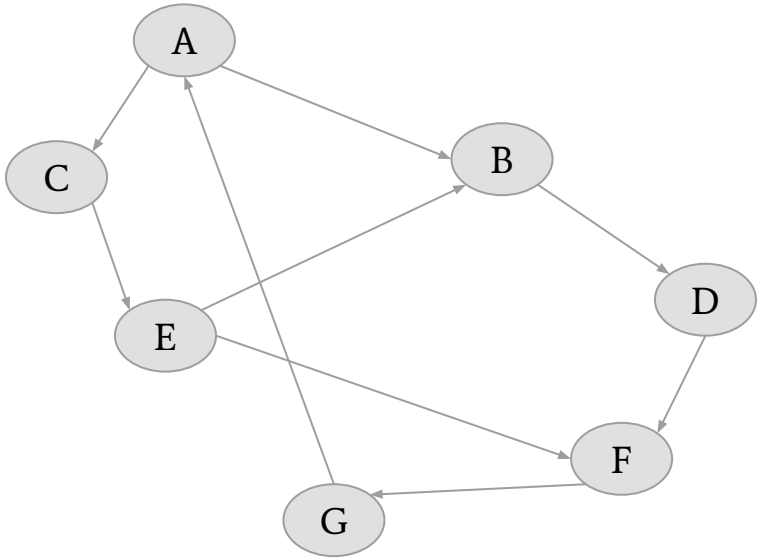



17
(mayores([3],2,_46),quicksort([],_47),quicksort(_46,_48),append(_47,[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
18
((3>2,!,mayores([],2,_61)),quicksort([],_47),quicksort([3|_61],_48),append(_47,[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
19
((!,mayores([],2,_61)),quicksort([],_47),quicksort([3|_61],_48),append(_47,[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
20
(mayores([],2,_61),quicksort([],_47),quicksort([3|_61],_48),append(_47,[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
21
(quicksort([],_47),quicksort([3],_48),append(_47,[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
22
(quicksort([3],_48),append([],[2|_48],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
23
((menores_iguales([],3,_70),mayores([],3,_71),quicksort(_70,_72),quicksort(_71,_73),append(_72,[3|_73],_69)),append([],[2|_69],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
24
((mayores([],3,_71),quicksort([],_72),quicksort(_71,_73),append(_72,[3|_73],_69)),append([],[2|_69],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
25
((quicksort([],_72),quicksort([],_73),append(_72,[3|_73],_69)),append([],[2|_69],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
26
((quicksort([],_73),append([],[3|_73],_69)),append([],[2|_69],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
27
(append([],[3],_69),append([],[2|_69],_44)),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
28
append([],[2,3],_44),quicksort([],_7),append(_44,[5|_7],_3)
{X/_3}
29
quicksort([],_7),append([2,3],[5|_7],_3)
{X/_3}
30
append([2,3],[5],_3)
{X/_3}
31
append([3],[5],_81)
{X/[2|_81]}
32
append([],[5],_85)
{X/[2,3|_85]}
33
c
{X/[2,3,5]}

Problema 3: Grafos

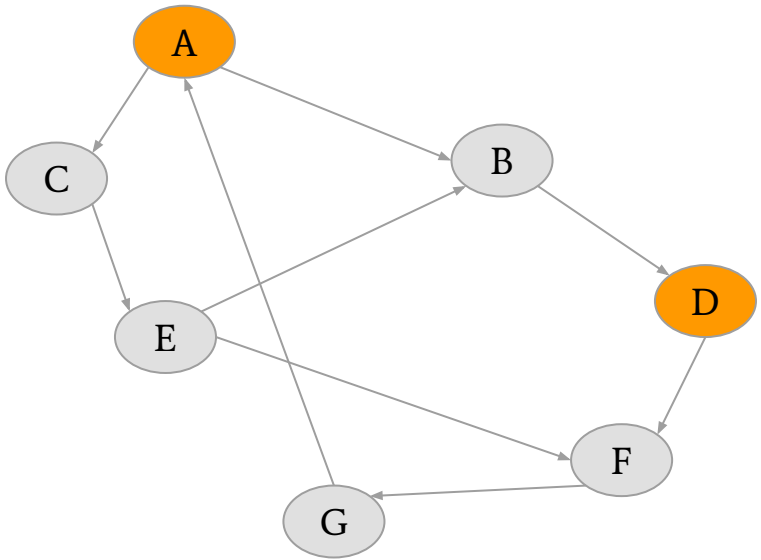
Problema 3: Grafos

Elaborar un programa que dado un grafo, sea capaz de encontrar un camino existente entre 2 puntos del mismo.

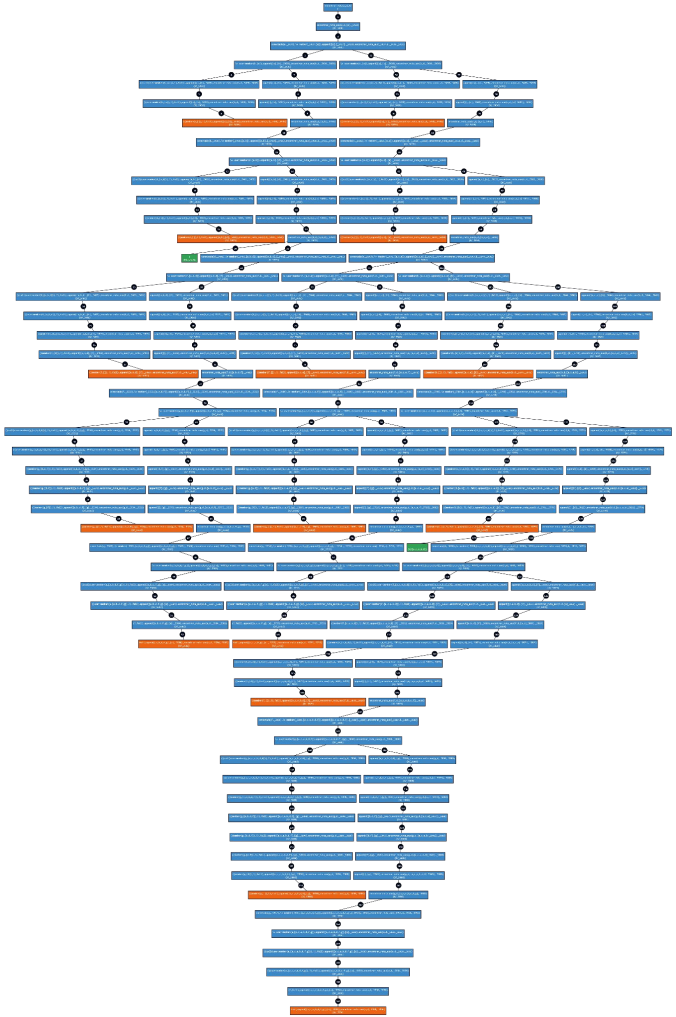


```
conectada(a, b).
conectada(a, c).
conectada(b, d).
conectada(c, e).
conectada(d, f).
conectada(e, f).
conectada(e, b).
conectada(f, g).
conectada(g, a).

% Predicado para encontrar una ruta desde Origen hasta Destino
encontrar_ruta(Origen, Destino, Camino) :-
    encontrar_ruta_aux(Origen, Destino, [Origen], Camino).

% Caso base: llegamos al destino
encontrar_ruta_aux(Destino, Destino, Camino, Camino).

% Caso recursivo: encontramos una conexión intermedia y continuamos la búsqueda.
encontrar_ruta_aux(Origen, Destino, Camino, CaminoFinal) :-
    conectada(Origen, Intermedia),
    \+ member(Intermedia, Camino),  % Evita ciclos.
    append(Camino, [Intermedia], CaminoExtendido), % Añade la conexión intermedia al camino.
    encontrar_ruta_aux(Intermedia, Destino, CaminoExtendido , CaminoFinal),
    !. % Usamos corte porque solo queremos la primera solucion.
```

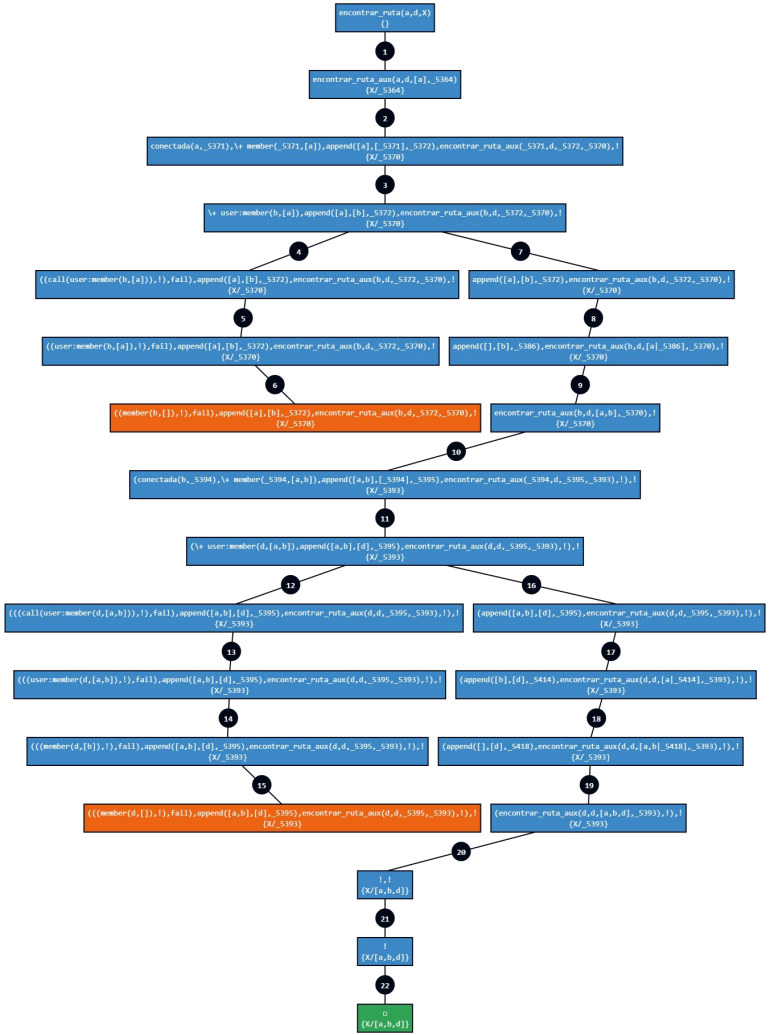
¿Cómo se realizaría el árbol para la consulta: encontrar_ruta(a,d,X). ?



Sin utilizar Corte (!)



Utilizando Corte (!)



Repositorio Github

Link de repositorio de github con código ejemplo:

https://github.com/bautifrigole/Paradigmas


Volvemos en una semana