



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

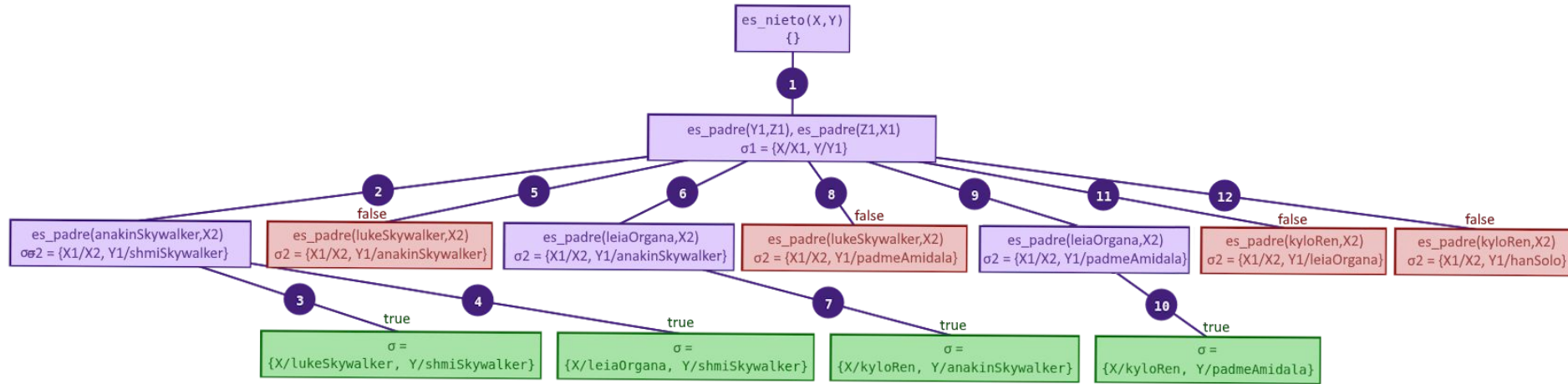
Supongamos que nos proponemos identificar los parentescos entre todos los personajes de la saga Star Wars.



**SPOILER
ALERT**

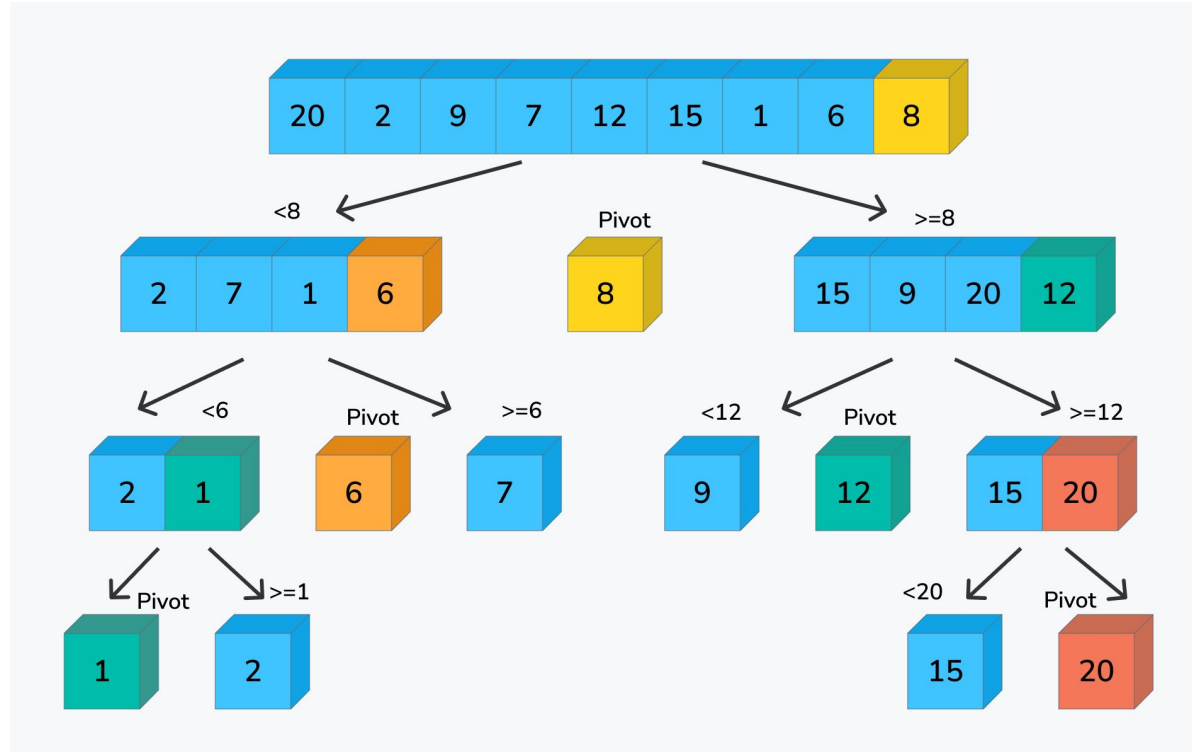
```
1 es_padre(shmiSkywalker, anakinSkywalker).
2 es_padre(anakinSkywalker, lukeSkywalker).
3 es_padre(anakinSkywalker, leiaOrgana).
4 es_padre(padmeAmidala, lukeSkywalker).
5 es_padre(padmeAmidala, leiaOrgana).
6 es_padre(leiaOrgana, kyloRen).
7 es_padre(hanSolo, kyloRen).
8
9 es_hermano(X,Y):-es_padre(Z,X),es_padre(Z,Y),X\==Y.
10 es_primo(X,Y):- es_padre(Z, Y), es_padre(W, X), es_hermano(W, Z).
11 es_nieto(X,Y):- es_padre(Y,Z),es_padre(Z,X).
12
13 es_descendiente(X,Y):- es_padre(Y,X).
14 es_descendiente(X,Y):- es_padre(Y,Z), es_descendiente(X,Z).
```

es_nieto(X, Y).



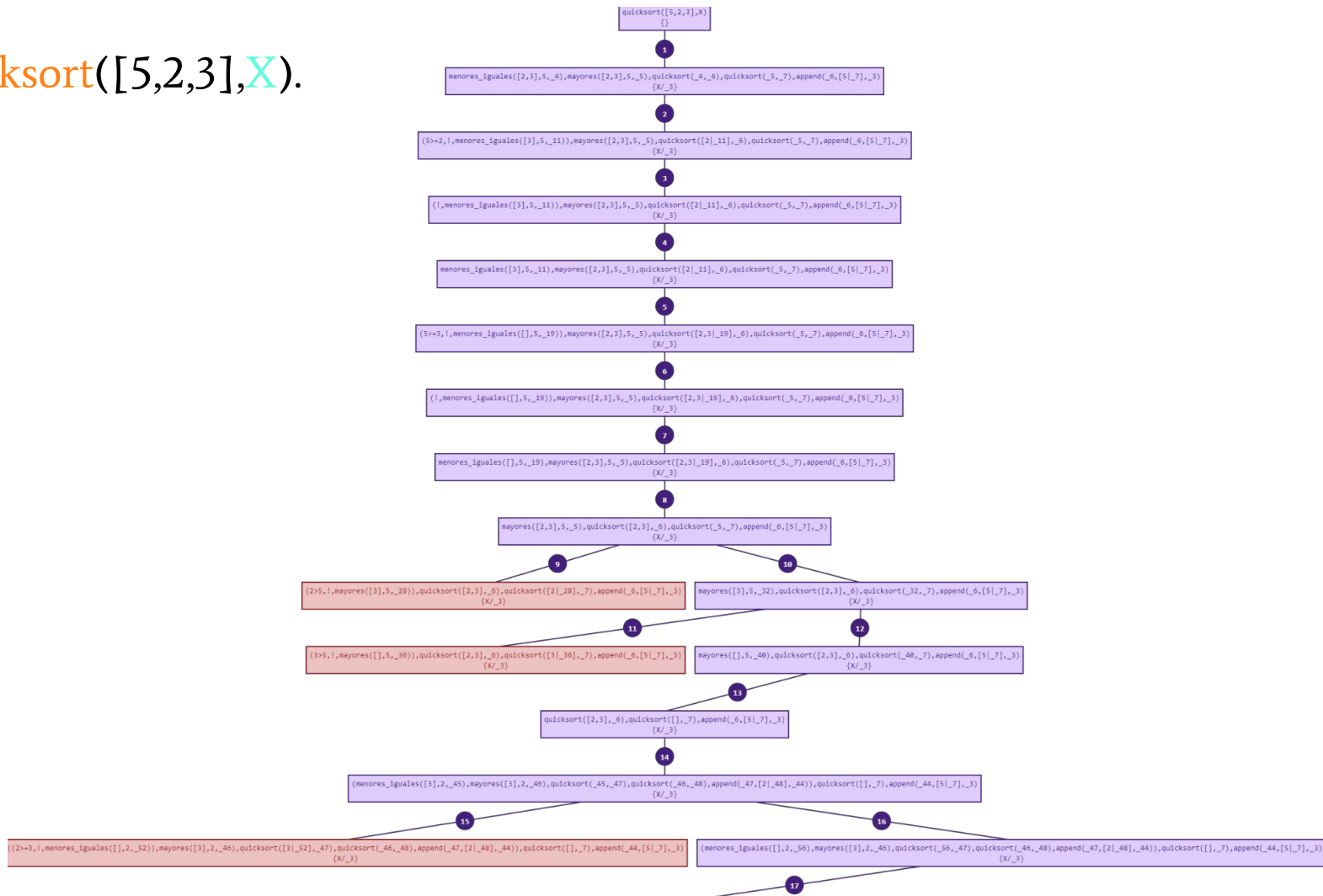
Problema 2: Quicksort

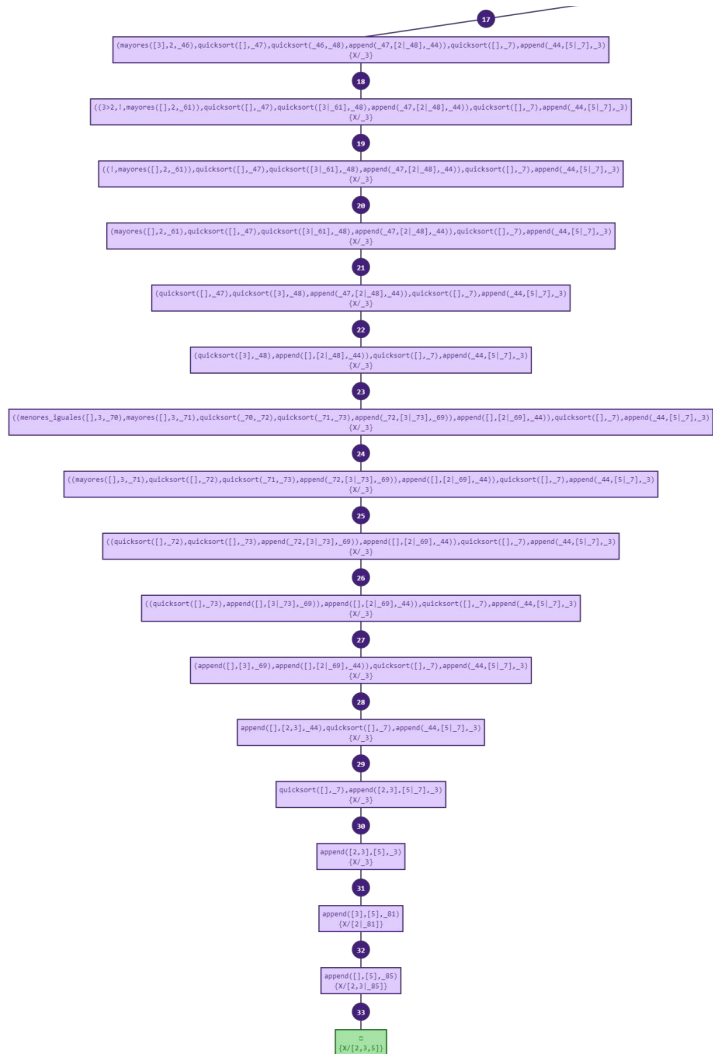
Problema 2: Quicksort



```
1 menores_iguales([],_,[]).
2 menores_iguales([H|T],X,[H|L]):- X >= H, !, menores_iguales(T,X,L).
3 menores_iguales([_|T],X,L):- menores_iguales(T,X,L).
4
5 mayores([],_,[]).
6 mayores([H|T],X,[H|L]):- H > X, !, mayores(T,X,L).
7 mayores([_|T],X,L):- mayores(T,X,L).
8
9 quicksort([],[]).
10 quicksort([H|T],L):- menores_iguales(T,H,L1), mayores(T,H,L2), quicksort(L1,Q1),
11    quicksort(L2,Q2), append(Q1, [H|Q2], L).
```

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

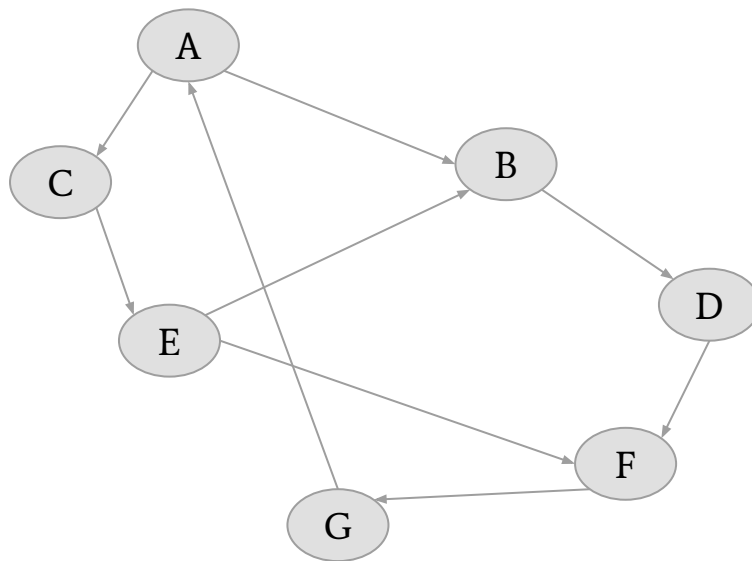




Problema 3: Grafos

Problema 3: Grafos

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

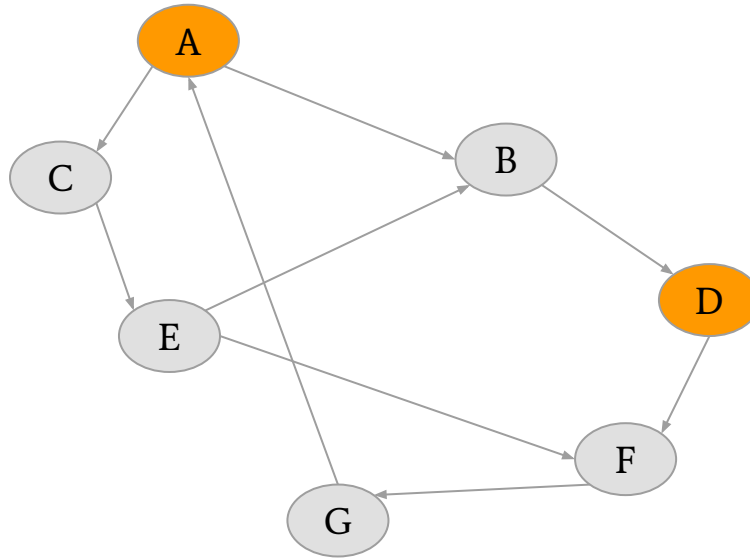


```

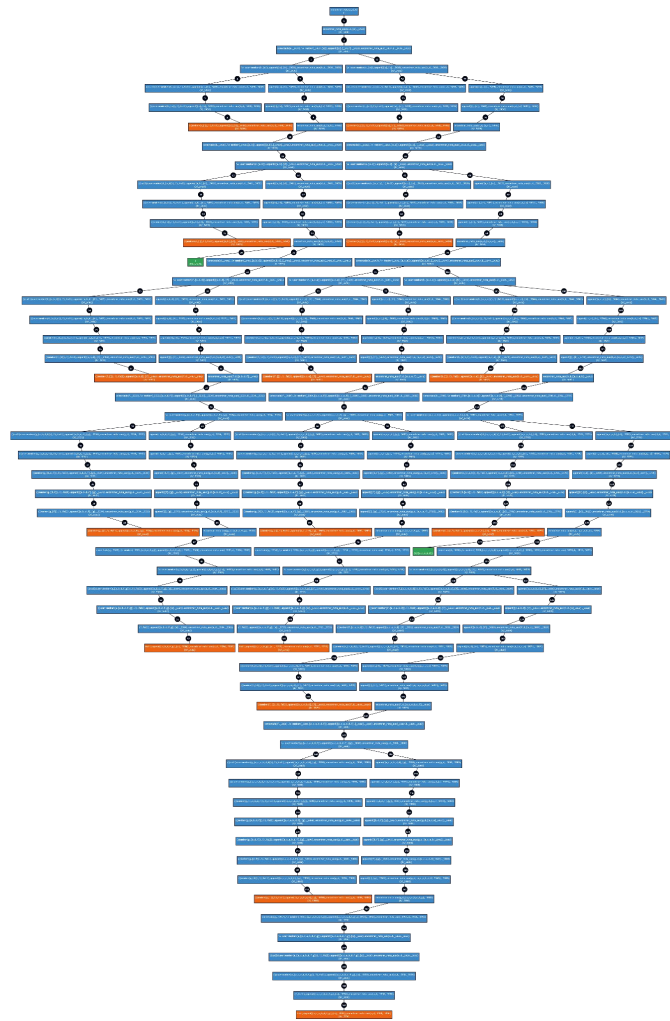
1  conectada(a, b).
2  conectada(a, c).
3  conectada(b, d).
4  conectada(c, e).
5  conectada(d, f).
6  conectada(e, f).
7  conectada(e, b).
8  conectada(f, g).
9  conectada(g, a).
10
11 % Predicado para encontrar una ruta desde Origen hasta Destino
12 encontrar_ruta(Origen, Destino, Camino) :-
13     encontrar_ruta_aux(Origen, Destino, [Origen], Camino).
14
15 % Caso base: Llegamos al destino
16 encontrar_ruta_aux(Destino, Destino, Camino, Camino).
17
18 % Caso recursivo: encontramos una conexión intermedia y continuamos la búsqueda.
19 encontrar_ruta_aux(Origen, Destino, Camino, CaminoFinal) :-
20     conectada(Origen, Intermedia),
21     \+ member(Intermedia, Camino), % Evita ciclos.
22     append(Camino, [Intermedia], CaminoExtendido), % Añade la conexión intermedia al camino.
23     encontrar_ruta_aux(Intermedia, Destino, CaminoExtendido, CaminoFinal),
24     !. % Usamos corte porque solo queremos la primera solución.

```

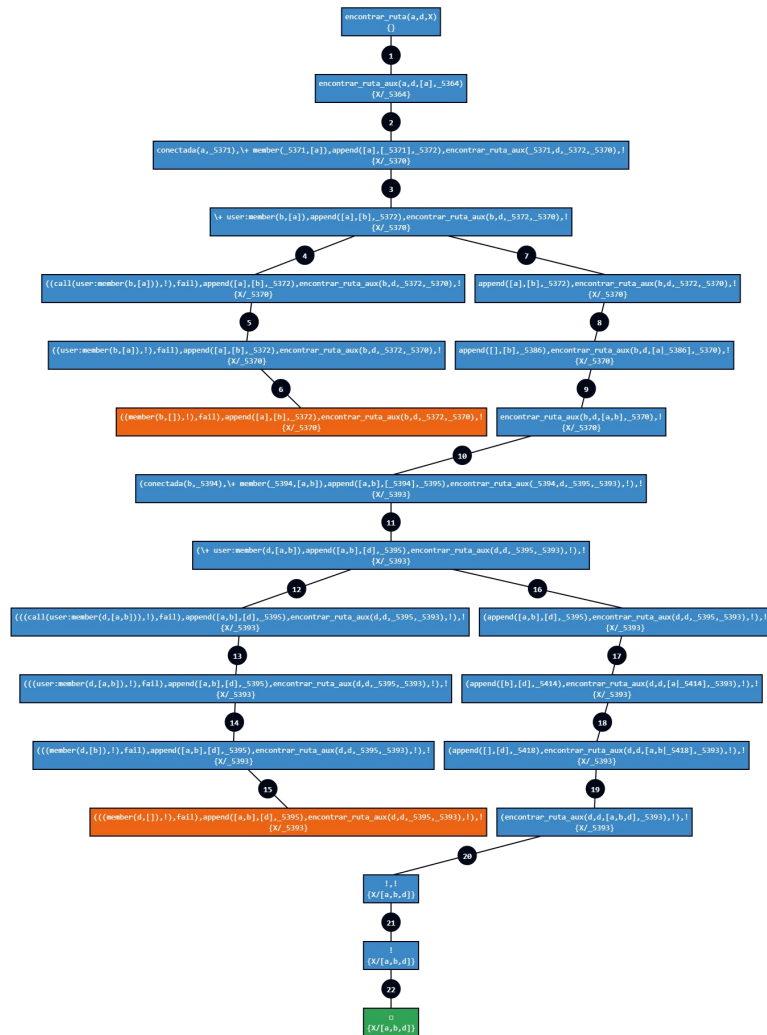

¿Cómo se realizaría el árbol para la consulta: `encontrar_ruta(a,d,X).` ?



Sin utilizar Corte (!)



Utilizando Corte (!)



Repositorio Github

Link de repositorio de github con código ejemplo:

<https://github.com/bautifrigole/Paradigmas>



Volvemos en una semana