Programación Lógica

2025

Instituto de Computación - Facultad de Ingeniería

Generación y Chequeo

Programación imperativa: la forma de obtener la solución está codificada en el algoritmo

Alternativa: describir el problema a través de un generador que enumera candidatos, y chequear en cada caso si es la solución

```
solucion(X):-
    generar(X), % generar candidato
    evaluar(X). % evaluar si es solución al problema
```

Ejemplo: suman_N

```
suman_N(+L,+N,?E1,?E2)
```

A partir de los elementos de L, obtener todos los pares de elementos que suman exactamente N

Versión 2: Sin sumar un elemento consigo mismo

select/3

```
% select(?X,?L,?R) ← X es un elemento de L, R es el resto

select(X,[X|L],L).

select(X,[Y|L],R) :- select(X,L,R).
```

Dependiendo de la instanciación, puede utilizarse por lo menos de tres formas:

- Obtener un elemento cualquiera y el resto
- Insertar un elemento en un lugar cualquiera
- Borrar una instancia de un elemento

Siempre de manera no determinista

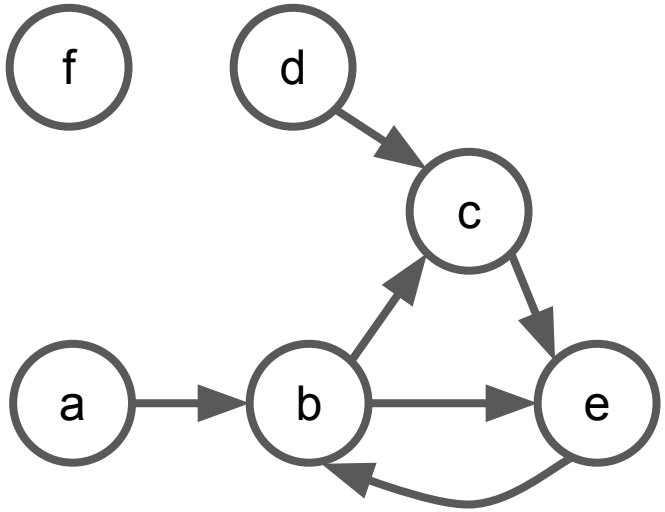
Permutación

```
permutacion(+L,?P)
```

P es una permutación de los elementos de L

¿Cuál permutación?

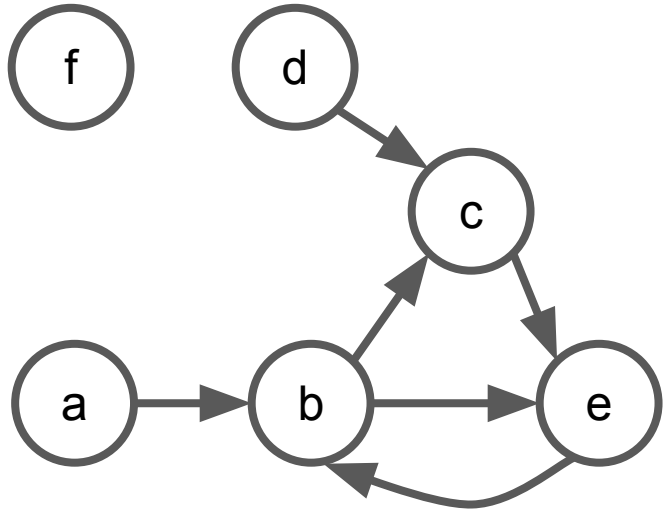
Todas!




```
vertice(a).
vertice(b).
vertice(c).
vertice(d).
vertice(e).
vertice(f).
```

```
arista(a,b).
arista(b,c).
arista(b,e).
arista(c,e).
arista(d,c).
arista(e,b).
```

```
hay_camino(X,Y) :- arista(X,Y).
hay_camino(X,Y) :- arista(X,Z), hay_camino(Z,Y).
```

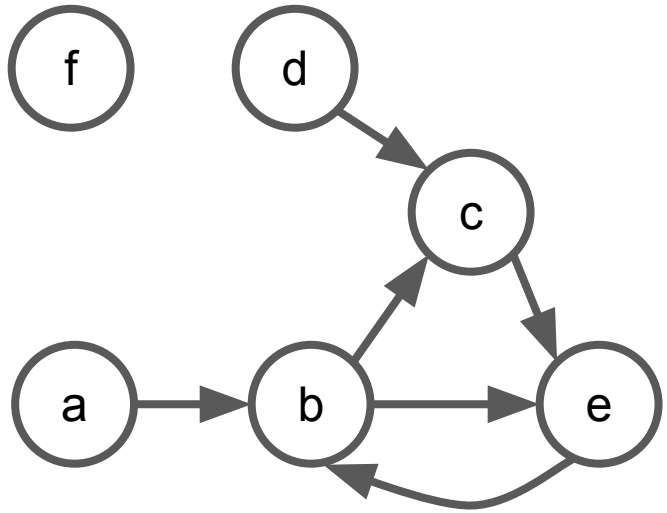



```
vertice(a).
vertice(b).
vertice(c).
vertice(d).
vertice(e).
vertice(f).
```

```
arista(a,b).
arista(b,c).
arista(b,e).
arista(c,e).
arista(d,c).
arista(e,b).
```

```
hay_camino(X,Y) :- hay_camino(X,Y,[X]).

% hay_camino(X,Y,V) ← V es la lista de visitados
hay_camino(X,X,_).
hay_camino(X,Y,V) :-
      arista(X,Z),
      not_member(Z,V)
      hay_camino(Z,Y,[Z|V]).
```




```
vertice(a).
vertice(b).
vertice(c).
vertice(d).
vertice(e).
vertice(f).
```

```
arista(a,b).
arista(b,c).
arista(b,e).
arista(c,e).
arista(d,c).
arista(e,b).
```

```
% camino(?X,?Y,?P) ← P es el camino
camino(X,Y,P) :- camino(X,Y,[X],P).

camino(X,X,_,[X]).
camino(X,Y,V,[X|P]) :-
      arista(X,Z),
      not_member(Z,V)
      camino(Z,Y,[Z|V],P).
```

Problemas de cruce de ríos

Un pastor tiene que atravesar a la otra orilla de un río con un lobo, una cabra y una lechuga. Dispone de una barca en la que solo caben él y una de las otras tres cosas. Si el lobo se queda solo con la cabra se la come, si la cabra se queda sola con la lechuga se la come. ¿Cómo debe hacerlo?

Problemas de cruce de ríos

Idea: representamos el problema como un grafo. Cada vértice representa un estado y las aristas indican cambios de estado:

```
% Defino primero las posiciones válidas

% Asumo que las listas están siempre ordenadas

grupo_valido([]).

grupo_valido([cabra]).

grupo_valido([lechuga]).

grupo_valido([lobo]).

grupo_valido([lechuga,lobo]).
```

Problemas de cruce de ríos

Idea: representamos el problema como un grafo. Cada vértice representa un estado y las aristas indican cambios de estado:

```
viaje(estado(Izq,Der,bote_izq),estado(RestoIzq,NewDer,bote_der)):-
    select(X,Izq,RestoIzq),
    grupo_valido(RestoIzq),
    sort([X|Der],NewDer).

viaje(estado(Izq,Der,bote_der),estado(NewIzq,RestoDer,bote_izq)):-
    select(X,Der,RestoDer),
    grupo_valido(RestoDer),
    sort([X|Izq],NewIzq).

% Viajes del pastor solo, es simplemente cambiar de lado el bote
viaje(estado(Izq,Der,bote_izq),estado(Izq,Der,bote_der)):-
    grupo_valido(Izq).
viaje(estado(Izq,Der,bote_der),estado(Izq,Der,bote_izq)):-
    grupo_valido(Der).
```

Problemas de cruce de ríos

Idea: representamos el problema como un grafo. Cada vértice representa un estado y las aristas indican cambios de estado:

```
% Ahora armamos los caminos, exactamente igual que antes
camino_pastor(Estado,Estado,_,[Estado]).
camino_pastor(Estado1,Estado2,V,[Estado1|Camino1]):-
     not_member(Estado1,V),
     viaje(Estado1,Estado3),
     camino_pastor(Estado3,Estado2,[Estado1|V],Camino1).


% La solución es buscar un camino entre el estado inicial y el final
problema(Solucion):-
    camino_pastor(estado([cabra,lechuga,lobo],[],bote_izq),estado([]
    ,[cabra,lechuga,lobo],bote_der),[],Solucion).
```