Paradigmas de Programación

Programación Lógica

Guía de Ejercicios Nº 3

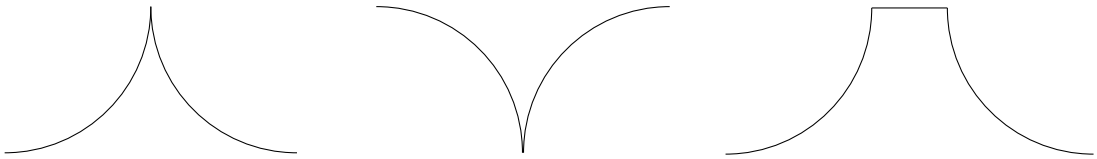
Índice de ejercicios

Listas

Ejercicio 1

Reconocimiento de patrones

Un dispositivo envía secuencias de datos bajo forma de listas enteros, los cuales se desean clasificar de acuerdo al patrón que los mismos representan. Se ha decidido que cada secuencia sea catalogada de la siguiente forma:



máximo mínimo meseta

y si la misma no sigue ninguno de estos esquemas se idetificará como sin orden.

Se le solicita que defina el predicado `identifica(Datos, Secuencia)`, el primer argumento es la secuencia de datos (una lista) y el segundo corresponde al símbolo que caracteriza la secuencia.

Ejemplos:
```
?- identifica([1, 3, 4, 10, 99, 98, 23, 2], Z).
Z = maximo
?- identifica([34, 3, 6, 99, 100], Z).
Z = minimo
?- identifica([34, 35, 67, 67, 67, 30, 21], meseta).
true
?- identifica([23, 45, 46, 2, 1, 4], sinorden).
true
```

Ejercicio 2

Palíndromo

Definir en Prolog el predicado `palindromo(Lista)` que evalúa verdadero si el argumento es un palíndromo.

Ejemplos:
```
?- palindromo([1, 2, 0, 0, 2]).
false
?- palindromo([1, 1, 2, 2, 1, 1]).
true
?- palindromo(['hola']).
true
```

Ejercicio 3

Sustitución de patrones

Definir un predicado `sust(Lista1, Patron1, Patron2, Lista2)` con todos sus argumentos listas. La evaluación será verdadera si `Lista2` es la resultante de sustituir todas las ocurrencias de `Patron1` por `Patron2` en `Lista1`.

Ejemplos:
```
?- sust([p, r, o, l, g, g, o, g, g, g ], [g, g], [], L).
L = [p, r, o, l, o, g]
?- sust([1, 0, 0, 1, 1], [2], [1, 1], [1, 0, 0, 1, 1]).
true
?- sust([a, b, c, d, 1, a, b, c, 1, a], [1, a], [c], T).
T = [a, b, c, d, c, b, c, c]
?- sust([1, 2, 3, 4, z], [1, 2, 4], [a, b, c, d, e, f], L).
L = [1, 2, 3, 4, z]
```

Ejercicio 4

Subsecuencia creciente más larga

El problema de la subsecuencia creciente más larga ("longest increasing subsequence problem") consiste en encontrar, a partir de una secuencia dada, la subsecuencia de valores en orden estrictamente creciente de mayor longitud posible. Por ejemplo, dada la secuencia S = (20 19 1 5 2 4 6), la subsecuencia (1 2 4 6) es solución al problema planteado.

Se solicita definir `subCrecienteMayor(Secuencia, Sub)` que se satisface si `Sub` es una subsecuencia estrictamente creciente de mayor longitud de Secuencia. Debe usar `subSecuencia/2` (Guía de Ejercicios Nº 2).

Ejemplos:
```
?- subCrecienteMayor([20, 19, 1, 5, 2, 4, 6], S).
S = [1, 2, 4, 6]
?- subCrecienteMayor([2, 4, 8, 3, 4], S).
S = [2, 3, 4] ;
S = [2, 4, 8]
```

Ejercicio 5

Intercala tramos

Se desea desarrollar un programa Prolog que permita intercalar los elementos de dos listas considerando tramos de intercalación de distintas longitudes:

```
intercala(Lista1, Lista2, LongTramoL1, LongTramoL2, Resultado)
```

Es decir, para cada lista se indicará la cantidad de elementos de la misma que se intercalarán sucesivamente para formar la lista resultante. Debe tenerse en cuenta que cada lista incluya la cantidad de elementos necesaria para ser intercalada correctamente y que cualquiera de ellas puede intercalar el tramo inicial o el tramo final.

Ejemplos:
```
?- intercala([a1, a2, a3, a4, a5, a6], [b1, b2, b3, b4], 3, 2,
        [a1, a2, a3, b1, b2, a4, a5, a6, b3, b4]).
true
?- intercala([a1, a2, a3, a4], [b1, b2], 4, 1, X).
X = [b1, a1, a2, a3, a4, b2]
```

Base de datos

Ejercicio 6

Libros

Dada la siguiente base de datos de libros en Prolog:

```
libro(titulo("The Art of Prolog"), editorial("The MIT Press"),
     autor([sterling, shapiro])).
libro(titulo("The Unified Software Development Process"), editorial("Addison-Wesley"),
     autor([jacobson, booch, rumbaugh])).
libro(titulo("Logic for Problem Solving"), editorial("Prentice Hall"),
     autor([kowalski])).

edicion(titulo("Logic for Problem Solving"), numero(1),
     isbn(978-0444003683), idioma("inglés")).
edicion(titulo("The Art of Prolog"), numero(2),
     isbn(978-0262193382), idioma("inglés")).
```

```
edicion(titulo("Logic for Problem Solving"), numero(2),
     isbn(978-3837036299), idioma("inglés")).
edicion(titulo("The Unified Software Development Process"), numero(1),
     isbn(978-8478290369), idioma("español")).
edicion(titulo("The Art of Prolog"), numero(1),
     isbn(978-2225819483), idioma("francés")).
edicion(titulo("The Unified Software Development Process"), numero(1),
     isbn(978-0201571691), idioma("inglés")).

impreso(isbn(978-0262193382), pag(560), precio(65), anio(1994)).
impreso(isbn(978-0201571691), pag(472), precio(56), anio(2000)).
impreso(isbn(978-3837036299), pag(346), precio(64), anio(2014)).
impreso(isbn(978-2225819483), pag(505), precio(30), anio(1997)).
impreso(isbn(978-8478290369), pag(472), precio(25), anio(2000)).
```

Definir los predicados solicitados:

a) `edicionImpresa(Titulo, Idioma, Anio)`: evalúa verdadero el libro titulado `Titulo` tiene una edición impresa en el año `Anio` y en el idioma dado por el segundo argumento.
Ejemplo:
```
?- idioma(Tit, "español", A).
Tit = "The Unified Software Development Process", A = 2000
```

b) `cantidadAutores(Titulo, Cantidad)`: se satisface si `Cantidad` corresponde a la cantidad de autores del libro con `Titulo`.
Ejemplos:
```
?- cantidadAutores("The Art of Prolog", C).
C = 2
?- cantidadAutores(T, C).
T = "The Art of Prolog", C = 2 ;
T = "The Unified Software Development Process", C = 3 ;
T = "Logic for Problem Solving", C = 1
```

c) `listaEditoriales`[1]`(Editoriales)`: evalúa verdadero si el argumento es una lista compuesta por todas las editoriales.
Ejemplo:
```
?- listaEditoriales(L).
L = ["The MIT Press", "Addison-Wesley", "Prentice Hall"]
```

d) `precioMaximo`[2]`(Titulo)`: se satisface si el libro con `Titulo` tiene la edición impresa de mayor precio.
Ejemplo:
```
?- precioMaximo(T).
T = "The Art of Prolog"
```

Ejercicio 7

Campeonato de selecciones

Se cuenta con información sobre partidos jugados en un campeonato de fútbol de selecciones en los siguientes predicados Prolog:

[1] Para la resolución debe armar una lista de resultados (apunte Estrategias Prolog, punto A).
[2] Para la resolución puede considerar el uso de la doble negación (apunte Estrategias Prolog, punto B).

`seleccionado(Jugador, Seleccion)`	Representa que Jugador fue seleccionado y pertenece a Seleccion. Del jugador se tiene el apellido y número.
`partido(NroPartido, Eq1, Eq2, GolesEq1, GolesEq2)`	Representa la relación entre los equipos que jugaron un partido y los goles realizados por cada uno de ellos.
`presencia(NroPartido, Jugador)`	Representa que Jugador jugó el partido indicado por NroPartido.

En base a este conjunto de hechos definir los siguientes predicados:

a) `jugadoresVictoriosos(Lista)`, que evalúe verdadero si el argumento es una lista de los apellidos de los jugadores que hayan ganado todos los partidos que jugaron.

b) `partidosJugados(Lista)`, que evalúe verdadero si el argumento es una lista donde cada elemento tiene la estructura: `[NroPartido, Eq1, Eq2]`. Además, la lista debe estar ordenada ascendentemente por `NroPartido`.

Considere la siguiente base de datos:

```
seleccionado(jugador("Messi", 10), argentina).
seleccionado(jugador("Karacic", 5), australia).
seleccionado(jugador("Estrada", 11), ecuador).
seleccionado(jugador("Mendy", 6), senegal).
partido(35, ecuador, senegal, 1, 2).
partido(21, tunez, australia, 0, 1).
partido(61, argentina, croacia, 3, 0).
partido(50, argentina, australia, 2, 1).
partido(18, qatar, senegal, 1, 3).
partido(8, argentina, arabia, 1, 2).
partido(1, qatar, ecuador, 0, 2).
partido(39, polonia, argentina, 0, 2).
presencia(21, jugador("Karacic", 5)).
presencia(35, jugador("Mendy", 6)).
presencia(8, jugador("Messi", 10)).
presencia(61, jugador("Messi", 10)).
presencia(1, jugador("Estrada", 11)).
presencia(39, jugador("Messi", 10)).
presencia(18, jugador("Mendy", 6)).
presencia(50, jugador("Messi", 10)).
```

Ejemplos:

```
?- jugadoresVictoriosos(Js).
Js = ["Karacic", "Estrada", "Mendy"]
?- partidosJugados(Ps).
Ps = [[1, qatar, ecuador], [8, argentina, arabia], [18, qatar, senegal], ... ]
```

Ejercicio 8

Agencia de viajes

Una agencia de viajes desea ofrecer tours a varios destinos turísticos. Para ello cuenta con el predicado `ciudad/2` que representa a qué país pertenece una ciudad que puede formar parte del tour.

Además, la política de la agencia establece que luego de salir de un país no se puede volver a ingresar al mismo. Por ejemplo, si las ciudades elegidas son Venecia, Atenas y Roma, los posibles tours deberán: 1) comenzar en Italia, visitar Roma y Venecia (en cualquier orden), para luego continuar a Grecia o 2) comenzar en Atenas para luego visitar las dos ciudades italianas.

Se solicita definir en Prolog el predicado `tour(Lista, Tour)` que evalúa verdadero si `Tour` es una lista con las mismas ciudades de `Lista` pero en una secuencia (que corresponde al orden de recorrido del tour) que cumple con la política de la agencia antes mencionada.

```
ciudad(dresde, alemania).
ciudad(basilea, suiza).
ciudad(roma, italia).
ciudad(hamburgo, alemania).
ciudad(atenas, grecia).
ciudad(venecia, italia).
ciudad(colonia, alemania).
ciudad(viena, austria).
ciudad(florencia, italia).
ciudad(munich, alemania).
ciudad(berna, suiza).
```

Ejemplos:

```
?- tour([roma, atenas, venecia], T).
T = [roma, venecia, atenas] ;
T = [atenas, roma, venecia] ;
T = [atenas, venecia, roma] ;
T = [venecia, roma, atenas]
?- tour([dresde, viena, colonia, hamburgo], T).
T = [dresde, colonia, hamburgo, viena] ;
T = [dresde, hamburgo, colonia, viena] ;
T = [viena, dresde, colonia, hamburgo] ...
```

Ejercicio 9

Tabla de resultados

Se está realizando un torneo deportivo y es necesario mantener actualizada la tabla de posiciones de acuerdo a los resultados. La tabla de posiciones contiene la siguiente información:

Equipo	Puntaje	Partidos Ganados	Partidos Empatados	Partidos Perdidos
EquipoA	11	3	2	0
EquipoB	9	2	3	1
EquipoD	7	1	4	1
...	...	...	...	...

Se solicita que esta tabla sea representada con una lista ordenada de acuerdo al puntaje de los equipos. Cada elemento de la misma poseerá los siguientes datos:

```
[equipo, puntaje, partidos ganados, partidos empatados, partidos perdidos]
```

Definir el predicado Prolog `actualizar(Tabla, Resultados, TablaActualizada)` que evalúa verdadero si `TablaActualizada` es la resultante de actualizar `Tabla` con la incorporación de los resultados dados en `Resultados`. `Resultados` es una lista en la cual se consignan los resultados de partidos:

```
[Equipo1, Equipo2, R]
```

Si R = 1, ganó el Equipo1 y se le adicionan 3 puntos.
Si R = 2, ganó el Equipo2 y se le adicionan 3 puntos.
Si R = 3, fue un empate y se adiciona 1 punto a cada equipo.

Al incorporar los resultados se deberá actualizar la información de los equipos involucrados en el mismo y asegurarse que la tabla de posiciones quede ordenada.

Ejemplo:

```
?- Tabla=[[equipoB, 0, 0, 0, 0], [equipoA, 0, 0, 0, 0], [equipoC, 0, 0, 0, 0]],
   Resultados=[[equipoC, equipoB, 3], [equipoA, equipoC, 2]],
   actualizar(Tabla, Resultados, TablaAct).
TablaAct = [[equipoC, 4, 1, 1, 0], [equipoB, 1, 0, 1, 0], [equipoA, 0, 0, 0, 1]]
```

Árboles

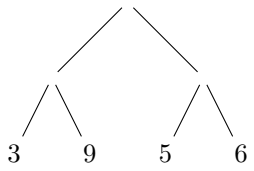
Ejercicio 10

Árbol binario ponderado

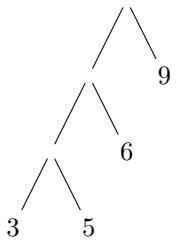
Sea $p_1, p_2, \ldots p_n$ un conjunto de números positivos los cuales reciben el nombre de pesos y A un árbol binario con n hojas (n > 1). Si asignamos a cada hoja del árbol un peso, obtenemos lo que se denomina un árbol binario para los pesos $p_1, p_2, \ldots p_n$. La siguiente fórmula

$$\sum_{i=1}^{n} p_i * h(p_i) \tag{1}$$

da el peso total del árbol, donde $h(p_i)$ es el número de nivel (cantidad de aristas para llegar a una hoja) asignado al peso p_i. A continuación se dan dos ejemplos:



`Arbol1`



`Arbol2`

Definir en Prolog los siguientes predicados:

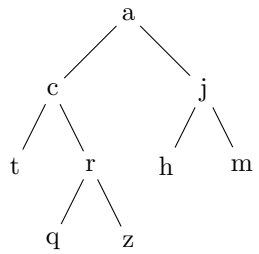
a) `pesoArbol(Arbol, Peso)`: evalúa verdadero si `Peso` corresponde al peso total de `Arbol` de acuerdo a (1).
Ejemplos:
```
?- pesoArbol(Arbol1, N).
N = 46
?- pesoArbol(Arbol2, 45).
N = true
```

b) `mapeoArboles(LArbol, Arbol)`: evalúa verdadero si `LArbol` es una lista de árboles binarios y `Arbol` es el árbol binario de `LArbol` con menor peso. Aquí deberá usar el predicado definido en el punto anterior.

Ejercicio 11

Recorrido horizontal

Una estrategia de recorrido de árboles es la HORIZONTAL, la cual consiste en visitar los nodos que están en un mismo nivel (de izquierda a derecha), luego proceder con los nodos del próximo nivel y así sucesivamente hasta visitar todos los nodos del árbol. A continuación se presenta un ejemplo de recorrido horizontal.



(a c j t r h m q z)

La forma de tratar este recorrido usualmente es trabajar con dos listas, ldesarrollado y lnodos. Se inicia lnodos con la raíz del árbol. Luego se retira el primer elemento de lnodos, se lo agrega en último lugar en ldesarrollado, se hallan los nodos sucesores del mismo en el árbol y se incorporan al final de lnodos. Se reitera el paso anterior, y el algoritmo termina cuando lnodos esta vacía y el recorrido será ldesarrollado. Por ejemplo:

Se le solicita que defina un predicado `recorridoHorizontal(Arbol, Lista)` que al ser evaluado permita obtener la lista con los nodos del árbol ordenados de acuerdo al recorrido horizontal antes descripto. Indique claramente como representará el árbol.

Ejercicio 12

Orden lexicográfico

Un árbol binario se considera ordenado con respecto a un criterio dado, cuando cada nodo raíz R del árbol o de cualquier sub-árbol contenido en él:

a) es precedido (de acuerdo con el criterio de orden establecido) por todo nodo raíz del sub-arbol izquierdo del sub árbol cuya raíz es R.

b) precede a todo nodo raíz del sub-árbol derecho del sub-arbol cuya raíz es R.

c) considere que todo nodo hoja es un sub-árbol ordenado.

Se le pide que defina en Prolog el predicado:

`ordenLexicografico(ArbolBinario)`

que posee el argumento ArbolBinario, el cual representa un árbol binario cuyos nodos son símbolos. La evaluación de ordenLexicográfico dará como resultado True si el árbol que es argumento se encuentra ordenado alfabéticamente.

Además, considere que puede utilizar el siguiente predicado (o sea, que ya está definido):

`list_text(ListaValoresASCII, Simbolo)`

el cual acepta como argumento un Símbolo y unifica el argumento `ListaValoresASCII` con la lista de valores ASCII correspondiente a los caracteres que componen el símbolo `Simbolo`.

Ejemplos:

```
?- list_text(L, paradigma).
L = [112, 97, 114, 97, 100, 105, 103, 109, 97]
```

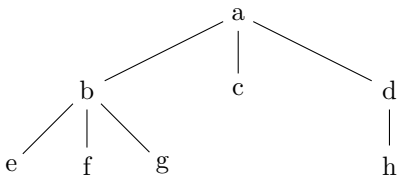
Los valores ASCII son enteros asociados a los caracteres que guardan el mismo orden que los caracteres que representan.

Ejercicio 13

Árboles n-arios

A diferencia de un árbol binario, un árbol n-ario tiene la característica de admitir, para cada nodo, un número $n \geq 0$ de descendientes. En Prolog, una representación posible para un árbol n-ario será

utilizando la función `nodo(R,Hs)`, donde R es el valor almacenado en el nodo y Hs es una lista de árboles n-arios que corresponde con los descendientes (o hijos) del mismo. Ejemplo de árbol n-ario con su representación en la variable `Arbol`:




```
Arbol = nodo(a, [
   nodo(b, [nodo(e, []), nodo(f, []),
              nodo(g, [])]),
   nodo(c, []),
   nodo(d, [nodo(h, [])])
                ]
             )
```

Se solicita definir los siguientes predicados:

a) `esHoja(ANario)`: que sea exitoso cuando el árbol n-ario pasado como argumento sea un nodo hoja (un nodo sin descendientes).
Ejemplos:
```
?- esHoja(Arbol).
false
?- esHoja(nodo(c, [])).
true
```

b) `cantidad(ANario, N)`: que dado un árbol n-ario permita obtener en `N` la cantidad de nodos del mismo.
Ejemplos:
```
?- cantidad(Arbol, N).
N = 8
?- cantidad(nodo(c, []), 1).
true
```

c) `listaHojas(ANario, Lista)`: que dado un árbol n-ario permite obtener la lista de todos los valores almacenados en los nodos hoja del mismo.
Ejemplos:
```
?- listaHojas(Arbol, L).
L = [e, f, g, c, h]
?- cantidad(nodo(c, []), L).
L = [c]
```

d) `quitaHojas(ANario1, ANario2)`: que dado el árbol n-ario `ANario1` permita obtener otro árbol n-ario `ANario2` que resulta luego de eliminar todos los nodos hoja de `ANario1`. El predicado debe fallar si `ANario1` es un nodo hoja.
Ejemplo:
```
?- quitaHojas(Arbol, A2).
A2 = nodo(a, [nodo(b, []), nodo(d, [])])
```

Matrices

Ejercicio 14

Operaciones con matrices

Una matriz puede ser representada mediante una lista de listas, siendo cada lista una fila de la matriz. Ejemplos de matrices y sus representaciones con listas en las variables `A`, `B` y `AT`:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \\ 4 & 4 & 4 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 4 \\ 7 & 2 & 10 \end{pmatrix} \qquad A^T = \begin{pmatrix} 1 & 1 & 4 \\ 2 & 3 & 4 \\ 3 & 5 & 4 \end{pmatrix}$$

```
A = [[1, 2, 3], [1, 3, 5],
     [4, 4, 4]]
```

```
B = [[0, 0, 4], [7, 2, 10]]
```

```
AT = [[1, 1, 4], [2, 3, 4],
      [3, 5, 4]]
```

Se solicita definir los siguientes predicados:

a) `elemento(Matriz, posicion(Fil, Col), Elemento)`: se satisface si `Elemento` se encuentra en la `Matriz` en la posición dada por la función `posicion(Fil, Col)`.
Ejemplo:
```
?- elemento(A, posicion(2, 1), E).
E = 1
```

b) `actualiza(Matriz, posicion(Fil, Col), Elemento, MatrizR)`: evalúa verdadero si `MatrizR` es la matriz resultante de colocar `Elemento` en `Matriz` en la posición dada por la función.
Ejemplo:
```
?- actualiza(A, posicion(2, 1), 9, M).
M = [[1, 2, 3], [9, 3, 5], [4, 4, 4]]
```

c) `transpuesta(Matriz, MatrizT)`: evalúa verdadero si `MatrizT` es la matriz transpuesta de `Matriz`.
Ejemplos:
```
?- transpuesta(AT, A).
true
?- transpuesta(B, BT).
BT = [[0, 7], [0, 2], [4, 10]]
```

d) `suma(Matriz1, Matriz2, MSuma)`: evalúa verdadero si `MSuma` es la matriz resultante de sumar `Matriz1` y `Matriz2`.
Ejemplo:
```
?- suma(A, A, MSuma).
MSuma = [[2, 4, 6], [2, 6, 10], [8, 8, 8]]
```

e) `sumaDiagonales(Matriz ,SD)`: evalúa verdadero si `SD` es una lista compuesta por los elementos que se obtienen de sumar la diagonal principal y diagonal secundaria de `Matriz`.
Ejemplo:
```
?- sumaDiagonal(A, L).
L = [4, 6, 8]
```

Ejercicio 15

Matriz rala

Existen muchas aplicaciones en las cuales se utilizan matrices de dos dimensiones de gran tamaño. Si la mayoría de los elementos de la matriz son ceros, la matriz se denomina Matriz Rala. Obviamente una Matriz Rala puede ser más eficientemente representada si se almacenan en memoria sólo las celdas con valores distintos de cero.

Con tal motivo se le solicita que defina un predicado convierte, con dos argumentos: la matriz completa y la representación compacta de la misma. El primer argumento es una matriz representada por una lista de listas, en donde cada una de las sublistas representa las respectivas filas de la matriz. Por ejemplo:

```
[[10, 0, 0, 10, 0], [0, 20, 0, 0, 0], [30, 0, 0, 0, 30],
                [40, 0, 0, 0, 0], [50, 0, 0, 0, 50]](*)
```

Es una matriz de 5×5, en donde A[1, 1] = 10, A[1, 2] = 0, A[2, 3] = 0, A[5, 5] = 50, etc.

Definir el predicado `convierte(Matriz,Rala)`, siendo el primer argumento una matriz con la representación antes mencionada (*) y `Rala` una lista de listas donde las sublistas corresponden a los elementos distintos de cero. Cada sublista tendrá tres elementos: `[fila, columna, valor]`. Para el ejemplo anterior:

```
Rala = [[1, 1, 10], [1, 4, 10], [2, 2, 20], [3, 1, 30], [3, 5, 30],
                  [4, 1, 40], [5, 1, 50], [5, 5, 50]]
```

Grafos

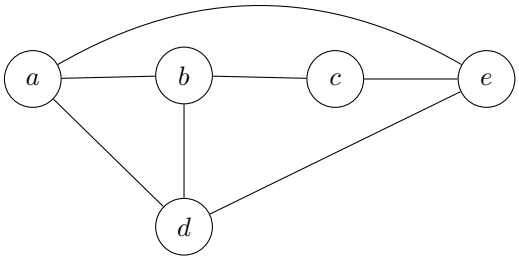
Ejercicio 16

Grafo no dirigido

Un grafo es un par $G = (V, A)$, donde V es un conjunto de elementos denominados vértices y A es un conjunto de pares de vértices denoninados aristas. Una posible representación en Prolog es con la función `grafo(Vs, As)`, donde `Vs` es una lista de vértices y `As` es una lista de funciones `a(v1, v2)` indicando que los vértices `v1` y `v2` están conectados (existe una arista).

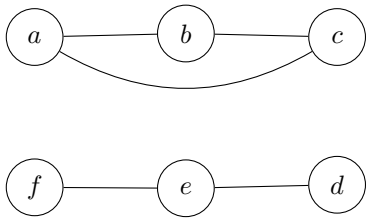
A continuación se presentan dos grafos con sus definiciones y representaciones en las variables `Grafo1` y `Grafo2`:

$G_1 = \big(\{a, b, c, d, e\}, \{(a, b), (a, d), (a, e), (b, c), (b, d), (c, e), (e, d)\}\big)$




```
Grafo1 = grafo([a,b,c,d,e],[a(a,b),
   a(a,d),a(a,e),a(b,c),a(b,d),
   a(c,e),a(e,d)])
```

$G_2 = \big(\{a, b, c, d, e, f\}, \{(a, b), (a, c), (b, c), (d, e), (f, e)\}\big)$




```
Grafo2 = grafo([a,b,c,d,e,f],[a(a,b),
   a(a,c),a(b,c),a(d,e),a(f,e)])
```

Se solicita definir los siguientes predicados Prolog:

a) `removerAristas(GrafoA, Vertice, GrafoB)`: evalúa verdadero si `GrafoB` es el grafo que se obtiene luego de haber eliminado del `GrafoA` todas las aristas que incluyen a `Vertice`.
Ejemplos:
```
?- removerAristas(Grafo1, e, G).
G = grafo([a, b, c, d, e], [a(a, b), a(a, d), a(b, c), a(b, d)]).
?- removerAristas(Grafo1, V, G).
V = a ; G = grafo([a, b, c, d, e], [a(b, c), a(b, d), a(c, e)]) ;
V = b ; G = grafo([a, b, c, d, e], [a(a, d), a(a, e), a(c, e)]) ...
```

b) `coberturaVertices(Grafo, Cobertura)`: se satisface si el segundo argumento es una cobertura de vértices de `Grafo`. Una cobertura de vértices, V', de un grafo $G = (V, A)$, es un subconjunto de V tal que cada arista de A tiene al menos uno de sus vértices en V'. Para la resolución debe usar el predicado del punto anterior.
Ejemplos:
```
?- coberturaVertices(Grafo2, [e, c, b]).
true
?- coberturaVertices(Grafo2 ,[a ,e]).
false
```

c) `clique(Grafo, Clique)`: evalúa verdadero si el segundo argumento es un clique de `Grafo`. Un clique, V', de un grafo $G = (V, A)$, es un subconjunto de V tal que cada par de vértices distintos en V' son adyacentes (tienen una arista que los conecta).
Ejemplos:
```
?- clique(Grafo1, [d, a, b]).
true
?- clique(Grafo2, [b, c]).
true
```

```
?- clique(Grafo2, [f, e, d]).
false
```

Ejercicio 17

Grafo conexo

Un grafo $G = (V, A)$ es conexo si para todo par de vértices $v_i, v_j \in V / v_i \neq v_j$ existe un camino (secuencia de aristas $\in A$) que los conecta. Por ejemplo, de los grafos del ejercicio anterior, G_1 es conexo mientras que G_2 es no conexo.

Se solicita definir los predicados `trail/3` para obtener los caminos sin repetición de aristas y `conexo/1` para determinar si un grafo dado es conexo:

a) `trail(Grafo, Camino, Vertices)`: evalúa verdadero si el segundo argumento es un camino de `Grafo` y `Vertices` es la lista de los vértices que corresponden al camino. Tener en cuenta que cada arista puede estar a lo sumo una vez en el camino.
Ejemplos:

```
?- trail(Grafo1[3], C, Vs).
C = [a(d, e)] ; Vs = [d, e] ;
...
C = [a(b, c), a(c, e), a(e, d), a(d, b)] ; Vs = [b, c, e, d, b] ;
...
?- trail(Grafo2[3], C, Vs), Vs=[a|_].
C = [a(a, c)] ; Vs = [a, c] ;
C = [a(a, c), a(c, b)] ; Vs = [a, c, b] ;
C = [a(a, b)] ; Vs = [a, b] ;
C = [a(a, b), a(b, c)] ; Vs = [a, b, c] ;
C = [a(a, b), a(b, c), a(c, a)] ; Vs = [a, b, c, a] ;
C = [a(a, c), a(c, b), a(b, a)] ; Vs = [a, c, b, a]
```

Observe que este predicado puede brindar distintos servicios.

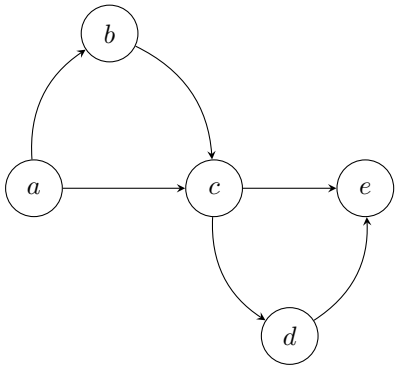
b) `conexo(Grafo)` el cual al ser evaluado tendrá éxito si el argumento corresponde a un grafo conexo, caso contrario fallará.

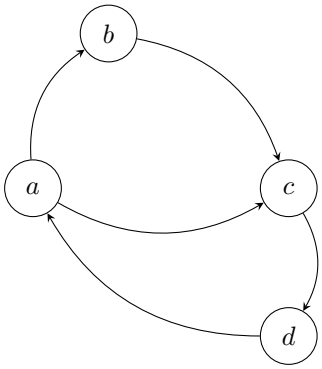
Ejercicio 18

Grafo dirigido

Una forma alternativa de representar[4] un grafo en Prolog es con una lista de aristas. Por cada arista hay una función de dos argumentos, los cuales corresponden a los vértices que conecta dicha arista. En este ejercicio se usarán grafos dirigidos, es decir, con direccionalidad en las aristas. Esta direccionalidad está dada por el orden de los argumentos de la función. A continuación hay dos ejemplos de grafos dirigidos con sus representaciones en las variables `Grafo3` y `Grafo4`:

[3] Corresponde al grafo del ejercicio 16.
[4] Las representaciones de grafos de esta guía pueden ser utilizadas indistintamente para grafos no dirigidos y dirigidos.






```
Grafo3 = [arista(a, b), arista(a, c),
   arista(b, c), arista(c, d),
   arista(c, e), arista(d, e)]
```

```
Grafo4 = [arista(a, b), arista(a, c),
   arista(b, c), arista(c, d),
   arista(d, a)]
```

Se solicita definir los siguientes predicados:

a) `gradoSalida(Grafo, Vertice, Grado)`: se satisface si `Grado` corresponde a la cantidad de aristas salientes de `Vertice` en `Grafo`.
Ejemplos:
```
?- gradoSalida(Grafo3 ,V, 2).
V = a ;
V = c
?- gradoSalida(Grafo4, V, N).
V = a, N = 2 ;
V = b, N = 1 ...
```

b) `ordenadosXGrado(Grafo, Vertices)`: evalúa verdadero si el segundo argumento es la lista de vértices de `Grafo` ordenados ascendentemente según el grado (salida). Para la resolución debe usar el predicado `gradoSalida/3`.
Ejemplos:
```
?- ordenadosXGrado(Grafo4, Vs).
Vs = [d, b, c, a]
?- ordenadosXGrado(Grafo3, Vs).
Vs = [e, b, d, a, c]
```

c) `walk(Grafo, Camino, Vertices, Long)`: se satisface si `Camino` es un camino de longitud `Long` (cantidad de aristas) de `Grafo`. `Vertices` es la lista de los vértices que corresponden al camino. A diferencia de `trail/3`, en este predicado no hay restricciones y el camino puede incluir aristas repetidas.
Ejemplos:
```
?- walk(Grafo3, C, Vs, 3).
C = [arista(a, b), arista(b, c), arista(c, d)], Vs = [a, b, c, d] ;
C = [arista(a, b), arista(b, c), arista(c, e)], Vs = [a, b, c, e] ;
C = [arista(a, c), arista(c, d), arista(d, e)], Vs = [a, c, d, e] ;
C = [arista(b, c), arista(c, d), arista(d, e)], Vs = [b, c, d, e] ;
?- walk(Grafo4, C, Vs, 5).
C = [arista(a, b), arista(b, c), arista(c, d), arista(d, a), arista(a, b)],
Vs = [a, b, c, d, a, b] ...
```

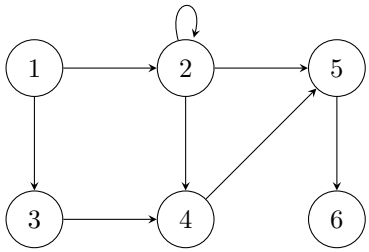
Ejercicio 19

Generación de grafo (longitud de camino)

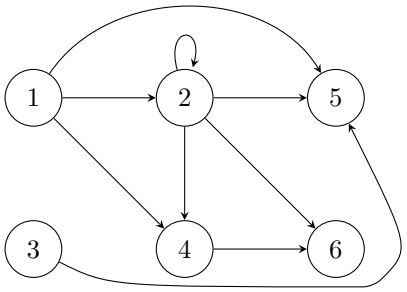
Se le solicita que defina el predicado `generaGrafo(Grafo1, N, Grafo2)` que genere a partir de `Grafo1` otro grafo, `Grafo2`, en el cual cada arista entre dos vértices representa la existencia de un

camino de longitud N en `Grafo1`.

En el siguiente ejemplo se muestra un grafo y el resultante para caminos de longitud 2. Por ejemplo, en el segundo grafo hay una arista del vértice 1 al 2, ya que existe un camino 1, 2, 2 en el grafo original.



Grafo original



Grafo que vincula aristas unidas por caminos de longitud 2 en el Grafo original

Indique claramente el tipo de representación utiliza para los grafos.

Varios

Ejercicio 20

Lista de posiciones de un elemento

Se le solicita que defina el predicado `ocurrencias(Elemento, Lista, Posiciones)`. El primer argumento (`Elemento`) puede aparecer una o más veces en `Lista`. El tercer argumento (`Posiciones`) es una lista de enteros que indican las posiciones en que ocurre `Elemento` en `Lista`.

La restricción que se impone es que no podrán utilizarse predicados con más de tres argumentos en la definición de ocurrencias, es decir, no utilizar predicados empleando argumentos contadores que siempre requieran ser evaluados con ese argumento con un valor, pudiéndose evaluar con ese argumento como variable.

Ejemplos:

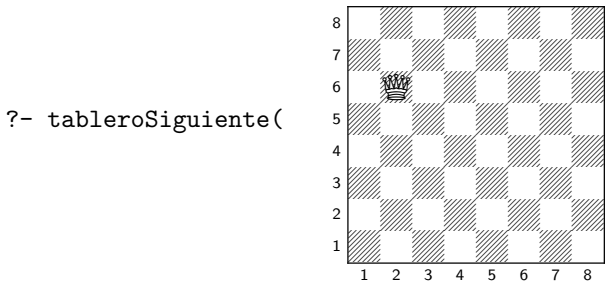
```
?- ocurrencias(a, [s, 1, 3, a, a, 2, d, a], T).
T = [4, 5, 8]
?- ocurrencias(2, [2, 1, 3, a, a, 2, d, a], [1, 6]).
true
?- ocurrencias(4, [s, 1, 3, a, a, 2, d, a], T).
T = []
```

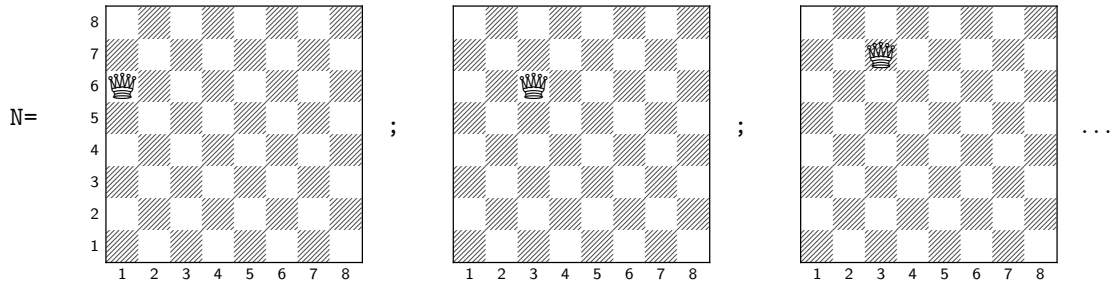
Ejercicio 21

Tablero con reina

Considere un tablero de ajedrez cuya única pieza sea una reina. Se le pide que desarrolle un programa Prolog que permita obtener los nuevos tableros posibles luego de realizar una movida con la mencionada pieza (la reina puede moverse las casillas que desee en dirección horizontal, vertical o diagonal). Ud. debe proponer y explicar la representación del tablero que utilice.

El predicado principal será `tableroSiguiente(TableroActual, TableroNuevo)`:





Ejercicio 22

Colorear mapa

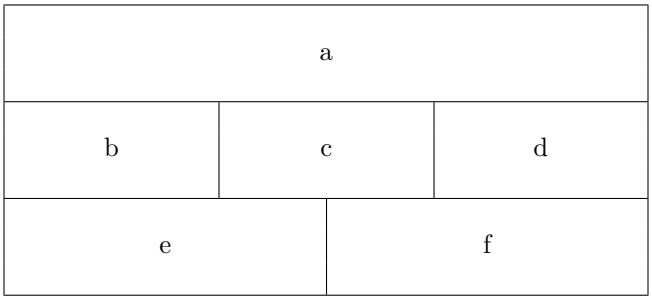
Se le solicita que proponga un programa Prolog para el problema de coloreado. El predicado principal es `colorear(Mapa, Colores)`. El argumento `Mapa` contiene la descripción de las regiones y los colores asignados a cada una de ellas, los cuales deben pertenecer al conjunto de colores disponibles en `Colores`.

La restricción del problema es que dos regiones vecinas no pueden tener asignado el mismo color.

a) Describa explícitamente que tipo de representación va a utilizar para Mapa y Colores.

b) Las reglas y predicados que componen el programa.

A continuación se presenta un mapa (simplificado) con sus regiones y colores asignados de forma que cumplen la restricción:



Ejercicio 23

Aprobación de materias

Se solicita definir el predicado `exigenciasCumplidas(Carrera, Nombre)` el cual será exitoso si el estudiante con `Nombre` ha aprobado todas las materias de la `Carrera`.

La información requerida se mantiene en los siguientes predicados (`exigenciasCumplidas/2` deberá trabajar con ellos):

```
estudiante(nombre("Romina Montalbán"), legajo(2490)).
estudiante(nombre("Daniel Hernández"), legajo(2310)).
...
materiaCarrera(carrera(ingenieria_civil), materia(analisis)).
materiaCarrera(carrera(ingenieria_civil), materia(algebra)).
materiaCarrera(carrera(ingenieria_civil), materia(materiales)).
...
materiaCarrera(carrera(ingenieria_quimica), materia(analisis)).
materiaCarrera(carrera(ingenieria_quimica), materia(quimica_general)).
materiaCarrera(carrera(ingenieria_quimica), materia(termodinamica)).
...
materiaAprobada(legajo(2490), carrera(ingenieria_quimica), materia(analisis)).
materiaAprobada(legajo(2490), carrera(ingenieria_quimica), materia(quimica_general)).
materiaAprobada(legajo(2490), carrera(ingenieria_quimica), materia(termodinamica)).
materiaAprobada(legajo(2310), carrera(ingenieria_civil), materia(analisis)).
...
```

Ejercicio 24

Secuencia faltante

Programar un predicado Prolog denominado `secuenciaFaltante(Lista1, Lista2)`, tal que para una lista dada (`Lista1`) que contiene una secuencia de números enteros ordenados ascendentemente, se obtenga en `Lista2` una secuencia ascendente de los números faltantes en `Lista1`. Los números faltantes que pertenecen a `Lista2` son aquellos que deberían encontrarse entre números consecutivos de `Lista1` pero no lo están.

Ejemplos:

```
?- secuenciaFaltante([1, 4, 6, 12], L2).
L2 = [2, 3, 5, 7, 8, 9, 10 ,11 ]
?- secuenciaFaltante([2, 4, 6], [1, 3, 5]).
false
?- secuenciaFaltante([-1, 0, 1, 2], []).
true
```

Ejercicio 25

Aerolínea

Empleando los siguientes predicados Prolog, una aerolínea cuenta con información de los vuelos que comercializa:

- `vuelo(IdVuelo, Desde, Hasta, Precio)` representa el hecho que el vuelo identificado por `IdVuelo` conecta los aeropuertos `Desde-Hasta` a un costo igual a `Precio`.
  Ejemplo:
  ```
  vuelo('IB750', 'Bruselas', 'BsAs', 151000).
  ```
- `vueloTramo(IdVuelo, Ntramo, Desde, Hasta, Km)` representa los tramos de un vuelo en particular, indicándose el número de kilómetros del tramo.
  Ejemplos sobre el vuelo definido en el punto anterior:
  ```
  vueloTramo('IB750', 1, 'Bruselas', 'Madrid', 1272).
  vueloTramo('IB750', 2, 'Madrid', 'SanPablo', 9725).
  ```
- `vueloDiario(IdVuelo, Día, NAvión, AsientosTotales)` representa el hecho que un vuelo va a ser realizado un día en particular con un avión determinado, el cual tiene un número total de asientos.

Ejemplo:
```
vuelo('IB750', '10/1/2023', 99765, 350).
```

- `segmentoVueloDiario(IdVuelo, Ntramo, Día, AsientosDisponible)` representa el hecho que el tramo de un vuelo realizado un día en particular tiene un número de asientosdisponibles (AsientosTotales - AsientosOcupados) igual a `AsientosDisponibles`.
  Ejemplos:
```
segmentoVueloDiario('IB750', 1, '10/1/2023', 312).
segmentoVueloDiario('IB750', 2, '10/1/2023', 350).
```

Definir los siguientes predicados:

a) `vueloCompleto(T, D)` representa que el vuelo `T` del día `D` no tiene asientos disponibles en al menos uno de sus tramos.

b) `asientosDisponibles(T, D, N)` representa que el vuelo `T` del día `D` tiene `N` asientos disponibles, siendo N es el mínimo de los asientos disponibles de todos los tramos.

c) `tramoMasPopular(Desde, Hasta)` representa el hecho que todos los vuelos entre 1/1/2023 y 28/2/2023, que han tenido como origen y destino `Desde` y `Hasta` han estado completos. Para definir este predicado considere que será el más popular aquel tramo que no tenga asientos disponibles.

d) `cantidadKm(T, N)` donde `T` es un determinado vuelo y `N` es la cantidad total de kilómetros del vuelo, incluyendo todos los tramos.