PRIMERA PARTE: HIPERCUBOS

En geometría, un hipercubo es un cubo n-dimensional. En un espacio tetradimensional, el hipercubo llamado teseracto es un cubo de cuatro dimensiones espaciales, que se compone de 8 cubos, 24 cuadrados (caras), 32 segmentos (aristas) y 16 puntos (vértices).

Se trata de programar un predicado **hipercubo/2** que proporcione el número de elementos n-dimensionales (puntos, segmentos, cuadrados, cubos, teseractos, etc.) que tiene un hipercubo de una dimensión dada.

hipercubo(N,L): N es un número natural (que vendrá dado en la llamada) que indica la dimensión del hipercubo en cuestión y L es la lista con los elementos existentes de cada dimensión, desde los elementos de dimensión N hasta los de dimensión 0.

Mostrar la llamada necesaria para encontrar los elementos del hipercubo de dimensión 12 y el resultado obtenido.

Una forma de calcular el número de elementos n-dimensionales es elevando el polinomio x+2 a la dimensión del hipercubo, siendo los coeficientes de cada potencia x^n del polinomio resultante el número de elementos n-dimensionales que tiene el hipercubo.

Por ejemplo, para el hipercubo de dimensión 2 (un cuadrado) tendríamos el polinomio (x+2)*(x+2) = x^2+4*x+4, es decir, tiene:

- un elemento de dimensión 2 (el cuadrado, correspondiente al sumando x^2),
- cuatro de dimensión 1 (segmentos) (los lados, correspondientes al sumando 4*x), y
- cuatro de dimensión 0 (puntos) (los vértices, correspondientes al sumando 4).

Ejemplos de llamadas al predicado hipercubo/2 son:

```
?- hipercubo(2,[1,4,4]).

      yes

?- hipercubo(0,L).

      L = [1] ?

      yes

?- hipercubo(5,L).

      L = [1,10,40,80,80,32] ? ;

      no

?- hipercubo(-1,L).

      no
```

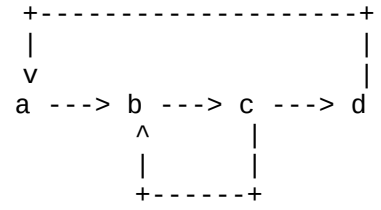
SEGUNDA PARTE: DETECTANDO CICLOS EN GRAFOS

Programar un predicado **hay_ciclo/2** para detectar la existencia de ciclos en grafos dirigidos.

hay_ciclo(Nombre, Recorrido): Nombre es el nombre asignado a una representación del grafo (que vendrá dado en la llamada) y Recorrido es una lista con nodos del grafo que forman un camino que contiene un ciclo (el camino que forma ciclo es una sublista de Recorrido). El predicado falla si no hay ninguna solución.

Los posibles grafos a los que se aplica el programa se almacenan mediante hechos del predicado grafo/2, de la forma grafo(Nombre, Grafo).

Ejemplo 1: grafo(seta, [l(a,b), l(b,c), l(c,d), l(c,b), l(d,a)]).



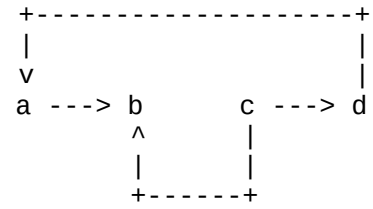
donde 'seta' es el nombre del grafo y cada arista se ha representado mediante una estructura l/2 cuyo primer argumento es el nodo origen de la arista, y cuyo segundo argumento es el nodo destino.

Una posible ejecución del programa sería como la siguiente (donde se han pedido dos soluciones):

```
?- hay_ciclo(seta, R).

     R = [b,c,b] ? ;

     R = [a,b,c,b] ?

     yes
```

Nota: La segunda solución contiene un nodo que no es parte del ciclo hallado, pero es válida porque sí contiene al ciclo.

Ejemplo 2: grafo(t, [l(a,b), l(c,d), l(c,b), l(d,a)]).



```
?- hay_ciclo(t, R).

     no
```

TERCERA PARTE: DECODIFICANDO LOS GENES

Se trata de resolver un problema relacionado con la búsqueda de la secuencia genética de una cadena de ARN: identificar aminoácidos dentro de una cadena. Las cadenas de ARN son secuencias de nucleótidos y cada tres nucleótidos codifican un aminoácido, componente básico de las proteínas. Se deben identificar las tripletas que codifican aminoácidos dentro de una secuencia de nucleótidos.

aminoacidos(ARN,Cadenas): la secuencia de nucleótidos ARN (que vendrá dada en la llamada) codifica las secuencias de aminoácidos Cadenas (en el mismo orden).

Existen cuatro nucleótidos diferentes: uracilo, citosina, adenina y guanina. Los abreviaremos por su primera letra: u, c, a, g. Existen 20 aminoácidos, que se abrevian a sus tres primeras letras en inglés: ala, arg, asn, asp, cys, gln, glu, gly, his, ile, leu, lys, met, phe, pro, ser, thr, trp, tyr, val. Las cadenas de ARN estarán representadas por “strings” de letras (de los aminoácidos); las cadenas de aminoácidos por secuencias de los nombres separados por guiones. Las cadenas identificadas estarán en una lista (que conservará el orden en que se encuentran en la cadena de ARN).

Nota: Un string es la lista de códigos ascii de las letras que lo forman, de manera que "acgu" es idénticamente igual a [97,99,103,117]. Compruébese:

```
?- X = "acgu".

      X = [97,99,103,117] ?

      yes
```

Dado que hay 4^3 = 64 posibles tripletas y solo 20 aminoácidos (más un código de parada), existe redundancia. Esto es, casi todos los aminoácidos se pueden codificar con varias tripletas diferentes. En muchos casos el último nucleótido de la tripleta es indistinto. A continuación utilizaremos números para abreviar varias distintas tripletas que codifican un mismo aminoácido. Los siguientes números representarán una cualquiera de las letras de su lista:

```
1 = [uc]    2 = [ag]    3 = [uca]   4 = [ucag]
```

La codificación de aminoácidos por nucleótidos es prácticamente universal, aunque actualmente se sabe que hay excepciones. La siguiente lista indica la codificación genética mas común (stop es el código de parada). Se utilizan los números anteriores para representar distintas secuencias de nucleótidos posibles (por lo que en realidad la lista contiene todas las 64 tripletas posibles).

```
ala = gc4   arg = ag2   arg = cg4   asn = aa1   asp = ga1   cys = ug1   gln = ca2
glu = ga2   gly = gg4   his = ca1   ile = au3   leu = cu4   leu = uu2   lys = aa2
met = aug   phe = uu1   pro = cc4   ser = ag1   ser = uc4   stop = ua2  stop = uga
thr = ac4   trp = ugg   tyr = ua1   val = gu4
```

Dada una cadena de ARN, la secuencia de tripletas que generan aminoácidos empieza siempre con la tripleta que se corresponde con el aminoácido metionina (met), a continuación siguen las tripletas que codifican los aminoácidos, que acaban con una tripleta que codifica el fin de la cadena (stop). Por ejemplo, la secuencia de nucleótidos augaaaaugggcuga codificaría la cadena de aminoácidos lys-met-gly. Los códigos de comienzo y parada no se incluyen en la cadena de aminoácidos.

Para complicar las cosas, no existen marcas en la cadena de ARN que indiquen donde empieza cada tripleta. Además, las secuencias de nucleótidos que generan aminoácidos pueden estar separadas por un número indeterminado de nucleótidos de composición aleatoria (aparentemente). Por ejemplo, la secuencia de nucleótidos ucaugacaaaauguggcuag codificaría (también) la cadena de aminoacidos lys-met-gly (uc(aug)ac(aaa)(aug)u(ggc)uag).

Pista: En tanto no se encuentre el código de comienzo (la tripleta que codifica met) se pueden ignorar todos los nucleótidos que se encuentren. En cambio, una vez encontrado, solo se pueden ignorar uno o dos nucleótidos, pero no tres, ya que

cualquier secuencia de tres codifica algo (un aminoácido o stop). Por ejemplo, la cadena del ejemplo anterior se puede leer también como uc(aug)a(caa)a(aug)u(ggc)uag (que codifica gln-met-gly) y como uc(aug)(aca)aa(aug)ug(gcu)ag (que codifica thr-met-gly) pero no como uc(aug)aca(aaa)(ugu)(ggc)uag (que codificaría lys-cys-gly, pero no es correcta porque ignora la tripleta "aca" que codifica thr).

Ejemplos de llamadas al predicado aminoacidos/2 son:

```
?- aminoacidos("augaaaaugggcuag",X).

      X = [lys-met-gly] ? ;

      X = [lys-trp] ? ;

      X = [lys-gly] ? ;

      X = [lys-trp] ? ;

      X = [lys-gly] ? ;

      X = [lys-gly] ? ;

      X = [asn-gly] ? ;

      X = [asn-gly] ? ;

      X = [gly] ? ;

      no

?- aminoacidos("augggcuagaugaaaaugggcuag",X).

      X = [gly,lys-met-gly] ? ;

      . . .

      X = [gly,gly] ? ;

      X = [gly] ? ;

      X = [gly-arg,gly] ? ;

      . . .

      % 85 soluciones en total

?- aminoacidos("augauguuacaccacaug",Cadenas).

      no
```