Estructuras Discretas

Inducción

Liliana Reyes

Universidad Nacional Autónoma de México
Facultad de Ciencias

21 de febrero de 2023

Recursión en Listas

Nuestro objetivo es comprender y manejar las listas desde un enfoque recursivo, para ello debemos de tener un **Caso Base** y un **Caso Recursivo**, los cuales definimos de la siguiente forma:

Definición

Una lista de elementos del mismo tipo, se define recursivamente como:

- **Caso Base:** La lista vacía, es una lista y la representamos por [ ].
- **Caso Recursivo** Si a es un elemento y xs una lista del mismo tipo de elementos, entonces la inclusión del elemento a a la lista xs, es una lista, la cual denotamos por $a : xs$.
  Donde : es la función constructora que añade el elemento a al inicio de la lista xs, de tal manera que a es la cabeza de la nueva lista y xs la cola.

Nota

Debemos hacer énfasis en algunos detalles

- El elemento a siempre debe ser del mismo tipo que los elementos de la lista xs que estemos considerando.
- La inclusión de un elemento en una lista, siempre coloca al elemento al inicio de la lista.

Ejemplo

Lista de frutas

Una lista de frutas se define recursivamente de la siguiente manera:

1. **Caso Base:** La lista vacía, la cual denotamos por [ ], es una *Lista de frutas*.
2. **Caso Recursivo:** Si f es una fruta y xs es una *lista de frutas* entonces $f : xs$ es una lista de frutas.

En concreto tenemos que, la lista de frutas [naranja, papaya, pera, manzana] utilizando la función constructora : se representa de la siguiente manera:

naranja:(papaya:(pera:(manzana:[])))

y se construye paso a paso como:

$L_0 =$ [] (por regla 1)

$L_1 =$ manzana : $L_0 =$ [manzana] (por regla 2)

$L_2 =$ pera : $L_1 =$ [pera, manzana] (por regla 2)

$L_3 =$ papaya : $L_2 =$ [papaya, pera, manzana] (por regla 2)

$L_4 =$ naranja : $L_3 =$ [naranja, papaya, pera, manzana] (por regla 2)

Ejemplo

Lista de números

Una lista de números se define de manera recursiva de la siguiente forma:

1. **Caso Base:** La lista vacía, la cual denotamos por [ ], es una *lista de números*.
2. **Caso Recursivo:** Si x es un número y xs es una *lista de números*; entonces $x : xs$; es una lista de números.

De manera similar a la lista de frutas, la lista [12,3,6,7,3] = 12:(3:(6:(7:(3:[]))), se construye de la siguiente forma:

$$L_0 = [] \text{ (por regla 1)}$$
$$L_1 = 3 : L_0 = 3 : [] \text{ (por regla 2)}$$
$$L_2 = 7 : L_1 = 7 : (3 : []) \text{ (por regla 2)}$$
$$L_3 = 6 : L_2 = 6 : (7 : (3 : [])) \text{ (por regla 2)}$$
$$L_4 = 3 : L_3 = 3 : (6 : (7 : (3 : [])) \text{ (por regla 2)}$$
$$L_5 = 12 : L_4 = 12 : (3 : (6 : (7 : (3 : [])))) \text{ (por regla 2)}$$

Funciones sobre listas

Consideremos algunas de las funciones **recursivas** básicas sobre listas y cómo definirlas.

Ejemplo

La función que obtiene el **último** elemento de una lista, devuelve el último elemento de una lista.
Caso Base: $\text{ultimo}\ [x] = x$
Caso Recursivo: $\text{ultimo}\ (x : xs) = \text{ultimo}\ xs$

Ejemplo

La función que obtiene los elementos **iniciales** de una lista, dada una lista regresa una lista con todos los elementos menos el último.
Caso Base: $\text{iniciales}\ [x] = [\ ]$
Caso Recursivo: $\text{iniciales}\ (x : xs) = x : (\text{iniciales}\ xs)$

Funciones sobre listas

NO Recursivas

Ahora consideramos dos de las funciones básicas sobre listas las cuales **NO** son recursivas, a pesar de ello veremos como definirlas ya que pueden ser útiles para realizar diversas funciones que sí lo sean.

Ejemplo

La función que obtiene la **cabeza** de una lista, esta función **no es recursiva**, y básicamente lo que hace es obtener el primer elemento de una lista.

$$\text{cabeza}\ (x : xs) = x$$

Ejemplo

La función que obtiene la **cola** de una lista, esta función **no es recursiva** y lo que hace es que dada una lista, regresa una lista que contiene todos los elementos de la lista menos la cabeza de la lista.

$$\text{cola}\ (x : xs) = xs$$

Concatenación de listas

La función **(concatLis** xs ys**)** une los elementos de ambas listas (por lo que ambas listas deben ser del mismo tipo), dando como resultado una lista que contiene todos los elementos de la primera lista xs, seguido de todos los elementos que contenga la segunda lista ys, $(xs \sqcup ys)$. Esta función puede ser definida mediante la siguiente definición recursiva:

Caso Base:

$$\text{concatLis } [] \; ys = ys$$

Caso Recursivo:

$$\text{concatLis } (x : xs) \quad ys = x : (\text{concatLis } xs \quad ys)$$

Ejemplo

Dadas dos listas del mismo tipo $L_1 = [1,2,3,4,5]$ y $L_2 = [6,7,8,9]$ concatenarlas.

$$\begin{aligned}
\text{concatLis } 1 : [2,3,4,5] \;\; [6,7,8,9] &= 1 : (\text{concatLis } 2 : [3,4,5] \;\; [6,7,8,9]) \\
&= 1 : (2 : (\text{concatLis } 3 : [4,5] \;\; [6,7,8,9])) \\
&= 1 : (2 : (3 : (\text{concatLis } 4 : [5] \;\; [6,7,8,9]))) \\
&= 1 : (2 : (3 : (4 : (\text{concatLis } 5 : [] \;\; [6,7,8,9])))) \\
&= 1 : (2 : (3 : (4 : (5 : (\text{concatLis } [] \;\; [6,7,8,9]))))) \\
&= 1 : (2 : (3 : (4 : (5 : [6,7,8,9])))) \\
&= [1,2,3,4,5,6,7,8,9]
\end{aligned}$$

Reversa de una lista

L

a función **(reversa** xs**)** obtiene la reversa de una lista. La reversa de una lista se considera como colocar los elementos en el orden inverso al que aparecían originalmente, la reversa de esta es la lista contiene los mismos elementos pero, el elemento que aparecía en la última posición ahora será el que esté en la cabeza de la lista, el elemento que se encontraba en la penúltima posición ahora se encontrará en la segunda y así sucesivamente hasta llegar al último elemento de la lista que será el que estaba en la primera posición.
Bajo estas consideraciones definimos la reversa de una lista de manera recursiva de la siguiente forma: **Caso Base**:

$$\text{reversa } [] = []$$

Caso Recursivo:

$$\text{reversa } (x : xs) = \text{concatLis } (\text{reversa } xs)[x]$$

Ejemplo

Tenemos la lista $[1,2,3,4]$, encontrar la reversa.

$$
\begin{aligned}
\text{reversa}\,1 : [2,3,4] &= \text{concatLis}(\text{reversa}\,2 : [3,4])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ (\text{reversa}\,3 : [4])[2])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ (\text{reversa}\,4 : [])[3])[2])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ (\text{reversa}[])[4])[3])[2])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ [][4])[3])[2])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ (\text{concatLis}\ [4][3])[2])[1] \\
&= \text{concatLis}\ (\text{concatLis}\ [4,3][2])[1] \\
&= \text{concatLis}\ [4,3,2][1] \\
&= [4,3,2,1]
\end{aligned}
$$

Pertenencia de un elemento a una lista

La función **(pertenece** e xs**)** verifica si el elemento e aparece en la lista xs, esta función regresa un valor booleano que nos indica si lo encontró o no. La función la podemos definir recursivamente de la siguiente forma: **Caso Base:**

$$\text{pertenece } e\ [] = \mathit{False}$$

Caso Recursivo:

$$\text{pertenece } e \quad (y : ys) = \text{if } (e = y)\ \mathit{True};\ \text{else pertenece } x \quad ys$$

Ejemplo

Teniendo un elemento d y una lista $L_1 = [a,b,c,d,e]$, queremos saber si el elemento pertenece a la lista.

$$\begin{aligned}
\text{pertenece } d \quad a:[b,c,d,e] &= \text{if}(d=a) \rightarrow \text{else pertenece } d \quad b:[c,d,e] \\
&= \text{if}(d=b) \rightarrow \text{else pertenece } d \quad c:[d,e] \\
&= \text{if}(d=c) \rightarrow \text{else pertenece } d \quad d:[e] \\
&= \text{if}(d=d) \rightarrow \mathit{True} \\
&= \mathit{True}
\end{aligned}$$

$$\begin{aligned}
\text{pertenece } z \quad a:[b,c,d,e] &= \text{if}(z=a) \rightarrow \text{else pertenece } z \quad b:[c,d,e] \\
&= \text{if}(z=b) \rightarrow \text{else pertenece } z \quad c:[d,e] \\
&= \text{if}(z=c) \rightarrow \text{else pertenece } z \quad d:[e] \\
&= \text{if}(z=d) \rightarrow \text{else pertenece } z \quad e:[] \\
&= \text{if}(z=e) \rightarrow \text{else pertenece } z \quad [] \\
&= \mathit{False}
\end{aligned}$$

Inducción en listas

Las listas es una de las estructuras de datos más comúnmente utilizada en computación, debido a que esta estructura la hemos definido de manera recursiva, es por ello que podemos utilizar el principio de inducción estructural.

Inducción en listas

Si tenemos una propiedad determinada P que cumplen las listas y se desea probar que dicha propiedad la cumplen todas las listas xs, basta proceder de la siguiente forma:

Base de la Inducción: Probar que $P([])$ cumple la propiedad.

Hipótesis de inducción: Suponer que $P(xs)$ cumple con la propiedad, donde xs es una lista con un número arbitrario de elementos.

Paso Inductivo: Probar que se cumple para $P(a : xs)$.

Si este es el caso, el principio de inducción para listas permite concluir que la propiedad P se cumple para cualquier lista xs.

Ejemplo

La función recursiva $\mathrm{concatLis}$ genera la concatenación de dos listas, es decir, para cualesquiera dos listas xs, ys, $\mathrm{concatLis}\ xs\ \ ys = xs \sqcup ys$.
Demostración: Inducción sobre la longitud de xs.
Base de la inducción: $xs = []$. Tenemos

$$[] \sqcup ys = ys = \mathrm{concatLis}\ []\ \ ys$$

Hipótesis de inducción: $\mathrm{concatLis}\ \ xs\ \ ys = xs \sqcup ys$
Paso inductivo: Demostrar que $\mathrm{concatLis}\ (a : xs)\ \ ys = (a : xs) \sqcup ys$

$$\begin{aligned}
(a : xs) \sqcup ys &= (a : (xs \sqcup ys)) && \text{(Razonamiento directo)}\\
&= (a : \mathrm{concatLis}\ xs\ \ ys) && \text{(H.I.)}\\
&= \mathrm{concatLis}\ (a : xs)\ \ ys && \text{(Definición recursiva de la función)}
\end{aligned}$$

En conclusión $\mathrm{concatLis}\ \ xs\ \ ys = xs \sqcup ys$ para cualquier lista xs, ys.

Ejemplo

La operacón de concatenación en listas es asociativa, es decir,

$$xs \sqcup (ys \sqcup zs) = (xs \sqcup ys) \sqcup zs$$

Demostración: Probaremos la asociatividad mediante inducción sobre la longitud de la lista xs.

Base de la inducción: $xs = []$, debemos mostrar que $[] \sqcup (ys \sqcup zs) = ([] \sqcup ys) \sqcup zs$.

$$\begin{aligned} [] \sqcup (ys \sqcup zs) &= ys \sqcup zs && \text{(Definición recursiva de } \sqcup) \\ &= ([] \sqcup ys) \sqcup zs && (ys = [] \sqcup ys) \end{aligned}$$

Hipótesis de Inducción: $xs \sqcup (ys \sqcup zs) = (xs \sqcup ys) \sqcup zs$

Paso Inductivo: Sea a un elemento, P.D. $(a : xs) \sqcup (ys \sqcup zs) = ((a : xs) \sqcup ys) \sqcup zs$

$$\begin{aligned} (a : xs) \sqcup (ys \sqcup zs) &= (a : (xs \sqcup (ys \sqcup zs))) && \text{(Definición recursiva de } \sqcup) \\ &= (a : ((xs \sqcup ys) \sqcup zs)) && \text{(H.I.)} \\ &= (a : (xs \sqcup ys)) \sqcup zs && \text{(Definición recursiva de } \sqcup) \\ &= ((a : xs) \sqcup ys) \sqcup zs && \text{(Definición recursiva de } \sqcup) \end{aligned}$$

Ejercicios

De manera similar demuestra las siguientes propiedades:

- $\text{longitud } xs \sqcup ys = \text{longitud } xs + \text{longitud } ys$
- $\text{longitud}(\text{reversa } xs) = \text{longitud } xs$
- $\text{reversa } xs \sqcup ys = \text{reversa } ys \sqcup \text{reversa } xs$