RECURSIVIDAD

- Conjuntos definidos recursivamente
  - Fórmulas en lógica de proposiciones
  - Los números naturales
  - Listas
- Correspondencias recursivas
  - Definición de correspondencia y de función recursiva
  - Árbol de dependencia de una correspondencia recursiva
  - Caracterización de funciones recursivas
  - Funciones recursivas definidas sobre números naturales y sobre listas

Bibliografía básica:

- Matemática Discreta. Libro de la asignatura. Cap. 4
- Matemática Discreta y sus aplicaciones. K.H. Rosen. Mc Graw Hill. Cap. 3
- Matemática discreta y lógica matemática. Hortalá; Leach; Rodríguez. Ed. Complutense. Cap. 1.3
- Matemática Discreta y Combinatoria. Grimaldi, R.P. Ed. Addison Wesley. Cap. 4, Cap. 10
- Matemática Discreta Norman L. Biggs. Cap 1.3 y Cap 12

Material de trabajo:

- Matemática Discreta. Problemas. Hoja 3.
- Actividad de Aprendizaje (Moodle)

Este material es un esquema de definiciones y resultados básicos. El objetivo del tema es el análisis (verificación y construcción) de funciones recursivas, por lo que el trabajo está centrado en la resolución de ejercicios y problemas.

MOTIVACIÓN:

Algunas veces se puede obtener la solución de un problema con un conjunto de datos de entrada a partir de la solución del mismo problema pero con un conjunto de datos de cardinal menor que el anterior, o bien del mismo problema con otros datos menores que los iniciales.

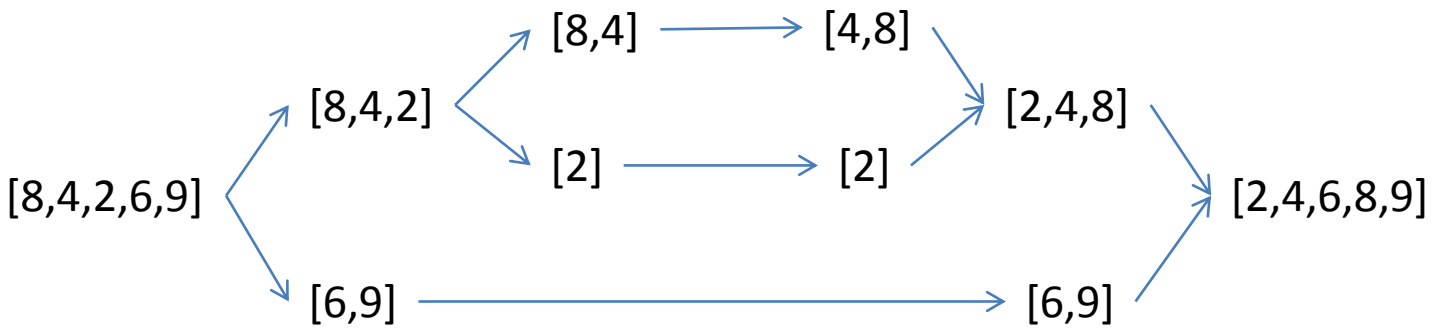
Ejemplo 1:

Si $a>b>0$, mcd (a, b) = mcd($b, a \bmod b$), esto permite reducir el problema de trabajar con valores a y b posiblemente muy grandes, a valores b y $a \bmod b$, que son menores que los iniciales. Si se sigue reduciendo con este criterio, se llega a mcd($c, 0) = c$, lo que detiene el proceso de reducción y proporciona el valor buscado.

mcd(33,21) = mcd(21, 12) = mcd(12,9) = mcd(9,3) = mcd(3, 0) = 3

Ejemplo 2:

Si se desea ordenar una lista de n números, se puede partir la lista en dos listas de tamaños lo más parecidos posibles, y, una vez ordenadas estas dos, mezclarlas adecuadamente. La estrategia de partir las listas se puede repetir hasta llegar a listas con uno o dos elementos, cuya ordenación es inmediata. El último paso sería ir mezclando listas adecuadamente.



Los ingredientes clave de esta estrategia son:

• reducir el problema a otro de la misma naturaleza pero de "tamaño" menor estableciendo la relación entre ambos
• llegar a casos iniciales cuya solución es fácil de obtener

CONJUNTOS DEFINIDOS RECURSIVAMENTE

Un conjunto está definido recursivamente si su definición consta de dos tipos de reglas:

- Reglas básicas: establecen que ciertos elementos están en el conjunto.
- Reglas recursivas: establecen que un elemento está en el conjunto si otros también lo están.

Ejemplos:

- El conjunto de fórmulas proposicionales

- El conjunto de los números naturales N
  a. Regla básica: $0 \in N$
  b. Regla recursiva: si $n \in N$ entonces $n+1 \in N$

- El conjunto de los números pares A
  a. Regla básica: $0 \in A$
  b. Regla recursiva: si $n \in N$ entonces $n+2 \in A$

CONJUNTOS DEFINIDOS RECURSIVAMENTE

Problemas asociados:

- Construir definiciones recursivas
- Determinar si un elemento pertenece a un conjunto

Ejemplos:

1- Dar una definición recursiva de los múltiplos naturales de 5.

2- Se considera el conjunto A de cadenas de bits definido por las reglas:

a. RB: La cadena 01 está en A
b. RR: Si α está en A, entonces $0\alpha1$ está en A

Generar algunos elementos de A y describir el tipo de cadenas que contiene A.

Analizar si 000111 y 00111 son cadenas de A

CONJUNTOS DEFINIDOS RECURSIVAMENTE

El conjunto de las listas

Aproximación intuitiva

[1, 2, 3, 4] [1, 2, 2, 2] [ ] [[2, 3], [2, 6], [3, 5], [4, 7]] [[3, [4,7]], [5,6], 2]

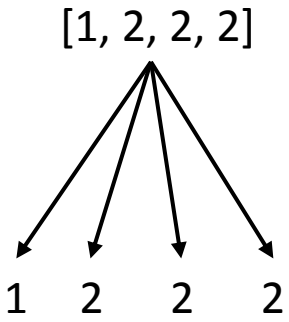
Sea Σ un conjunto no vacío. El conjunto de listas sobre Σ, LIST(Σ), se define mediante las siguientes reglas:

Reglas básicas: Si a0, a1, …., an $\in \Sigma$, entonces $[a_0, a_1, \ldots, a_n]$ es una lista. La secuencia [ ] es una lista, llamada lista vacía. Todas estas listas se denominan listas planas $LIST_P(\Sigma)$

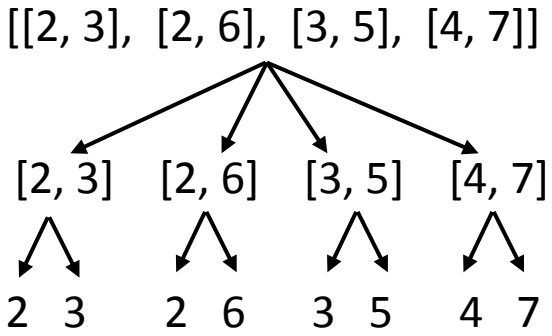
Reglas recursivas: Si $L_0, L_1, \ldots, L_n \in \Sigma \cup LIST(\Sigma)$, entonces $[L_0, L_1, \ldots L_n]$ es una lista.

CONJUNTOS DEFINIDOS RECURSIVAMENTE

Árbol estructural y profundidad de una lista

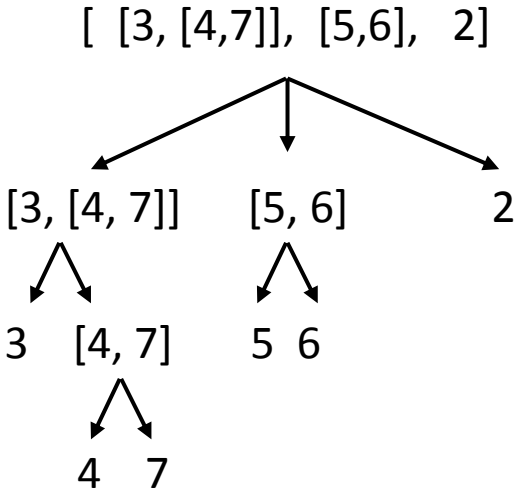


P(L) = 1



P(L) = 2

[ ] P(L) = 0



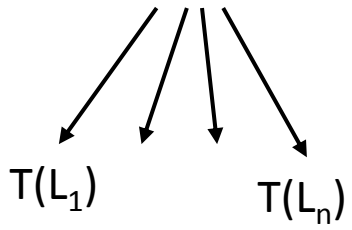
P(L) = 3

CONJUNTOS DEFINIDOS RECURSIVAMENTE

Se denomina árbol estructural de $L \in \Sigma \cup LIST(\Sigma)$ y se representa T(L) al árbol definido (recursivamente) por:

$T(L) = L$ Si $L = [\ ]$ o bien $L \in \Sigma$

$T(L) =$ L



Si $L = [L_1, \ldots, L_n]$

Se denomina profundidad de una lista a la longitud de la rama más larga de su árbol estructural.
La longitud de una rama de un árbol es el número de aristas de dicha rama.

FUNCIONES DEFINIDAS RECURSIVAMENTE

Una correspondencia recursiva es una correspondencia que aparece en su propia definición.

Ejemplos:

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \\ n * f(n-1) & si\ n \geq 1 \end{cases}$$

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \\ 1 + f((n-1)^2) & si\ n \geq 1 \end{cases}$$

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \\ 1 + f(n) & si\ n \geq 1 \end{cases}$$

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \\ 1 + f(n-2) & si\ n \geq 1 \end{cases}$$

FUNCIONES DEFINIDAS RECURSIVAMENTE

Una función recursiva es una correspondencia recursiva que es función y es evaluable a partir de las reglas que la definen en un número finito de pasos.
Dichas reglas se clasifican en:

a. Reglas básicas: definen explícitamente el valor de la función en algunos elementos. Dichos elementos forman el conjunto de partida o de salida.
b. Reglas recursivas: definen la función mediante referencias a sí misma.

Ejemplos:

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B. \\ n * f(n-1) & si\ n \geq 1 \quad R.R. \end{cases}$$

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B.1 \\ 4 & si\ n = 1 \quad R.B.2 \\ 1 + f(n-2) & si\ n \geq 2 \quad R.R. \end{cases}$$

FUNCIONES DEFINIDAS RECURSIVAMENTE

Dada una correspondencia recursiva f : A → B, para cada elemento a de A se denomina **árbol de dependencia de a Tf(a),** a un árbol en cuyos nodos aparecen los elementos en que debe evaluarse f, según la definición recursiva, para obtener el valor f(a).

Ejemplos: Los árboles de dependencia de n=4 para las siguientes correspondencias son:

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B. \\ n * f(n-1) & si\ n \geq 1 \quad R.R. \end{cases}$$

$f(4)_{RR} = 4*f(3)_{RR} = 4*3*f(2)_{RR} = 4*3*2*f(1)_{RR} = 4*3*2*1*f(0)_{RB} = 4*3*2*1*1$

Tf(4) =

FUNCIONES DEFINIDAS RECURSIVAMENTE

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B.1 \\ 4 & si\ n = 1 \quad R.B.2 \\ 1 + f(n-2) & si\ n \geq 2 \quad R.R. \end{cases}$$

f(4) = 1 + f(2) RR
f(2) = 1 + f(0) RR
f(0) = 1 RB
f(2) = 1+1
f(4) = 1+1+1 =3

Tf(4) =
4
2
0

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \\ 1 + f((n-1)^2) & si\ n \geq 1 \end{cases}$$

f(4) = 1 + f(9) RR
f(9) = 1 + f(64) RR
f(64) = 1 + f(4096) RR

Tf(4) =
4
64
4096

FUNCIONES DEFINIDAS RECURSIVAMENTE

Teorema: Una correspondencia recursiva f : A → B, es función recursiva si y solo si para todo a de A su árbol de dependencia es finito.

Ejemplo: Usar el método de inducción para probar que el árbol de dependencia Tf(n) es finito para todo n ∈ N.

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B. \\ n * f(n-1) & si\ n \geq 1 \quad R.R. \end{cases}$$

Paso base: f(0) =1 (RB) por lo tanto Tf(0) = 0 es un árbol de un solo nodo, es finito.

Paso de inducción: Sea n≥0, tal que Tf(n) es finito (H.I.). Hay que probar que Tf(n+1) también es finito. Por la RR se tiene que f(n+1) = (n+1)*f(n) y por lo tanto el árbol es:

Tf(n+1) = n+1
|
Tf(n)

La H.I. asegura que Tf(n) es finito, por lo tanto Tf(n+1) también es un árbol finito.

Queda probado que Tf(n) es finito para todo n

FUNCIONES DEFINIDAS RECURSIVAMENTE

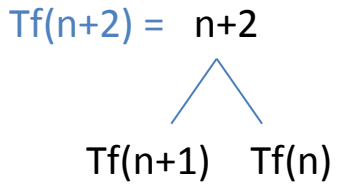
Ejemplo: Usar el método de inducción para probar que la siguiente correspondencia es función.

$$f: N \to N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B.1 \\ 4 & si\ n = 1 \quad R.B.2 \\ f(n-1) + f(n-2) & si\ n \geq 2 \quad R.R. \end{cases}$$

Para ver que es una función, basta probar que el árbol de dependencia es finito en cualquier valor del dominio.

Paso base: f(0) =1 (RB1) por lo tanto Tf(0) = 0 es un árbol de un solo nodo, es finito.
f(1) = 4 (RB2) por lo tanto Tf(1) = 1 es un árbol de un solo nodo, es finito.

Paso de inducción: Sea n≥0, tal que Tf(n) y Tf(n+1) son finitos (H.I.). Hay que probar que Tf(n+2) también es finito. Por la RR se tiene que f(n+2) = f(n+1)*f(n) y por lo tanto el árbol es:



La H.I. asegura que Tf(n) y Tf(n+1) son finitos, por lo tanto Tf(n+2) también es un árbol finito.

Queda probado que Tf(n) es finito para todo n.

FUNCIONES DEFINIDAS RECURSIVAMENTE

Observaciones:

• Si una correspondencia recursiva f:A $\rightarrow$B está definida en todos los elementos de A y f(a) es único, entonces f es función. En lugar de probar estas dos propiedades por inducción, se suele probar que el árbol de dependencia para todo a de A, es finito.

• Si se conoce una función explícita f y se construye g, una definición recursiva de f, para probar que g coincide con f, se suele usar el método de inducción.

Ejemplo: Dar una definición recursiva de $f(n) = 5^n$, $n \in N$, y probar que coincide con f.

Ejemplo: Obtener una forma explícita para la correspondencia recursiva siguiente. Probar que ambas coinciden.

$$f: N \rightarrow N \qquad f(n) = \begin{cases} 1 & si\ n = 0 \quad R.B. \\ n * f(n-1) & si\ n \geq 1 \quad R.R. \end{cases}$$

FUNCIONES DEFINIDAS RECURSIVAMENTE

En la definición de funciones recursivas sobre el conjunto de las listas, se usan algunas funciones básicas. Las más frecuentes son:

CAB: $\text{LIST}(\Sigma) - \{[\ ]\} \rightarrow \Sigma \cup \text{LIST}(\Sigma)$

CAB(L) proporciona el primer elemento de L

RESTO: $\text{LIST}(\Sigma) - \{[\ ]\} \rightarrow \text{LIST}(\Sigma)$

RESTO(L) proporciona la lista obtenida al borrar el primer elemento de L

CONC: $\text{LIST}(\Sigma) \times \text{LIST}(\Sigma) \rightarrow \text{LIST}(\Sigma)$

CONC(L1, L2) proporciona la lista obtenida al colocar los elementos de L1 en primer lugar y después los de L2. Notación: CONC(L1, L2) = L1 || L2

LISTA: { palabras sobre $\{\Sigma, [,], , \}$ } $\rightarrow$ $\{0, 1\}$

LISTA(L) devuelve 1 si L es lista y 0 en otro caso.

FUNCIONES DEFINIDAS RECURSIVAMENTE

Ejemplo: Dadas L1=[2,4,6,8,0] L2=[1,3,5,7] L3=[[0,2],3,4,5]

1. Obtener: CAB(L1), CAB(L3), RESTO(L2), RESTO(L3), CAB(CAB(L3)), CONC(L1,L2), CONC(L2,l3)
2. Usar las funciones elementales adecuadas para obtener a partir de las listas anteriores:
   [6,8,0] [1,3,5,7,2,4,6,8,0] [3,4,5] [2]

Para definir funciones recursivas sobre el conjunto de listas, también se suele utilizar la **función longitud,** que proporciona el número de elementos de una lista L.

$$LONG(L) = \begin{cases} 0 & si\ L = [\,] \\ 1 + LONG(RESTO(L)) & e.o.c. \end{cases}$$

FUNCIONES DEFINIDAS RECURSIVAMENTE

Ejemplo 1: Definir, de manera recursiva, una función que reciba una lista plana de naturales y devuelva la suma de los elementos de la lista

Suma: $\text{LIST}_P(N) \rightarrow N$

$$suma(L) = \begin{cases} 0 & si\ L = [\,] \\ CAB(L) + suma\big(RESTO(L)\big) & e.o.c. \end{cases}$$

Ejemplo 2: Definir, de manera recursiva, una función que reciba una lista plana L de naturales y devuelva la lista que contiene los elementos de L en orden inverso.

inversa: $\text{LIST}_P(N) \rightarrow \text{LIST}_P(N)$

$$inversa(L) = \begin{cases} [\,] & si\ L = [\,] \\ inversa\big(RESTO(L)\big)||[CAB(L)] & e.o.c. \end{cases}$$

FUNCIONES DEFINIDAS RECURSIVAMENTE

Ejemplo 3: Definir, de manera recursiva, una función que reciba una lista de naturales y devuelva la suma de los elementos de la lista

Suma: $\text{LIST}_P(N) \rightarrow N$

Ejemplo1 :

$$suma(L) = \begin{cases} 0 & si\ L = [\,] \\ CAB(L) + suma(RESTO(L)) & e.o.c. \end{cases}$$

Extensión: Suma: $\text{LIST}(N) \rightarrow N$

$$suma(L) = \begin{cases} 0 & si\ L = [\,] \\ CAB(L) + suma(RESTO(L)) & si\ L \neq [\,]\ y\ LISTA(CAB(L)) = 0 \\ suma(CAB(L)) + suma(RESTO(L)) & si\ L \neq [\,]\ y\ LISTA(CAB(L)) = 1 \end{cases}$$

Para probar que una correspondencia recursiva definida sobre el conjunto de listas es una función (Tf(L) finito para toda L) y que proporciona el resultado requerido, se suele usar el método de inducción.