Recursivitat

Tecnologia de la Programació

Sebastià Vila-Marta

Enginyeria de Sistemes TIC
Universitat Politècnica de Catalunya
http://itic.cat

7 d'abril de 2013

1 En el tema anterior. . .

2 Concepte de recursivitat

3 La recursivitat a les matemàtiques

4 De les matemàtiques a la programació

5 Sobre el disseny recursiu

6 Per a la setmana vinent . . .

En el tema anterior...

- UML és un llenguatge gràfic per descriure sistemes.
- Els diagrames UML de classe permeten representar l'estructura estàtica d'un sistema.
- Les diferents classes poden estar interrelacionades per: herència, associació, agregació i composició.
- EL diagrama de classes il·lustra molt bé què podem fer per implementar el sistema.

Concepte de recursivitat

Segons el diccionari...

Una cosa recursiva és aquella que es repeteix un nombre infinit de vegades aplicant la mateixa regla.

Observeu la capsa de la figura. És una imatge recursiva!

Definicions recursives

- Les definicions recursives són corrents a les matemàtiques.
- El cas més típic és el de la funció factorial:

$$n! = \begin{cases} 1 & n = 0 \\ (n-1)! \cdot n & n > 0 \end{cases}$$

- La seva interpretació és senzilla. Calculem, per exemple, el factorial de 5:

$$\begin{aligned} 5! = (5-1)! \cdot 5 &= 4! \cdot 5 = \\ &= (3! \cdot 4) \cdot 5 = \\ &= (2! \cdot 3) \cdot 4 \cdot 5 = \\ &= (1! \cdot 2) \cdot 3 \cdot 4 \cdot 5 = \\ &= (0! \cdot 1) \cdot 2 \cdot 3 \cdot 4 \cdot 5 = \\ &= 1 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = \\ &= 120 \end{aligned}$$

Definicions recursives II

També podem definir recursivament els nombres combinatoris així:

$$\binom{n}{k} = \begin{cases} 1 & k = 0 \\ 1 & k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{altrament} \end{cases}$$

O conjunts com aquest:

Definition

$P \subseteq \mathbb{N}$ és el conjunt definit per les següents regles:

1. $1 \in P$
2. $x \in P \iff x - 2 \in P$

Subprogrames recursius

- Passar de les definicions matemàtiques recursives a subprogrames Python recursius és només un pas.
- En el cas del factorial:

```
def factorial(n):
    if n == 0:
        return 0
    elif n != 0:
        return factorial(n−1)∗n
```

Proveu-ho!!

- Sabrieu definir el càlcul dels coeficients binomials recursivament?

Subprogrames recursius

- Passar de les definicions matemàtiques recursives a subprogrames Python recursius és només un pas.
- En el cas del factorial:

```
def factorial(n):
    if n == 0:
        return 0
    elif n != 0:
        return factorial(n−1)∗n
```

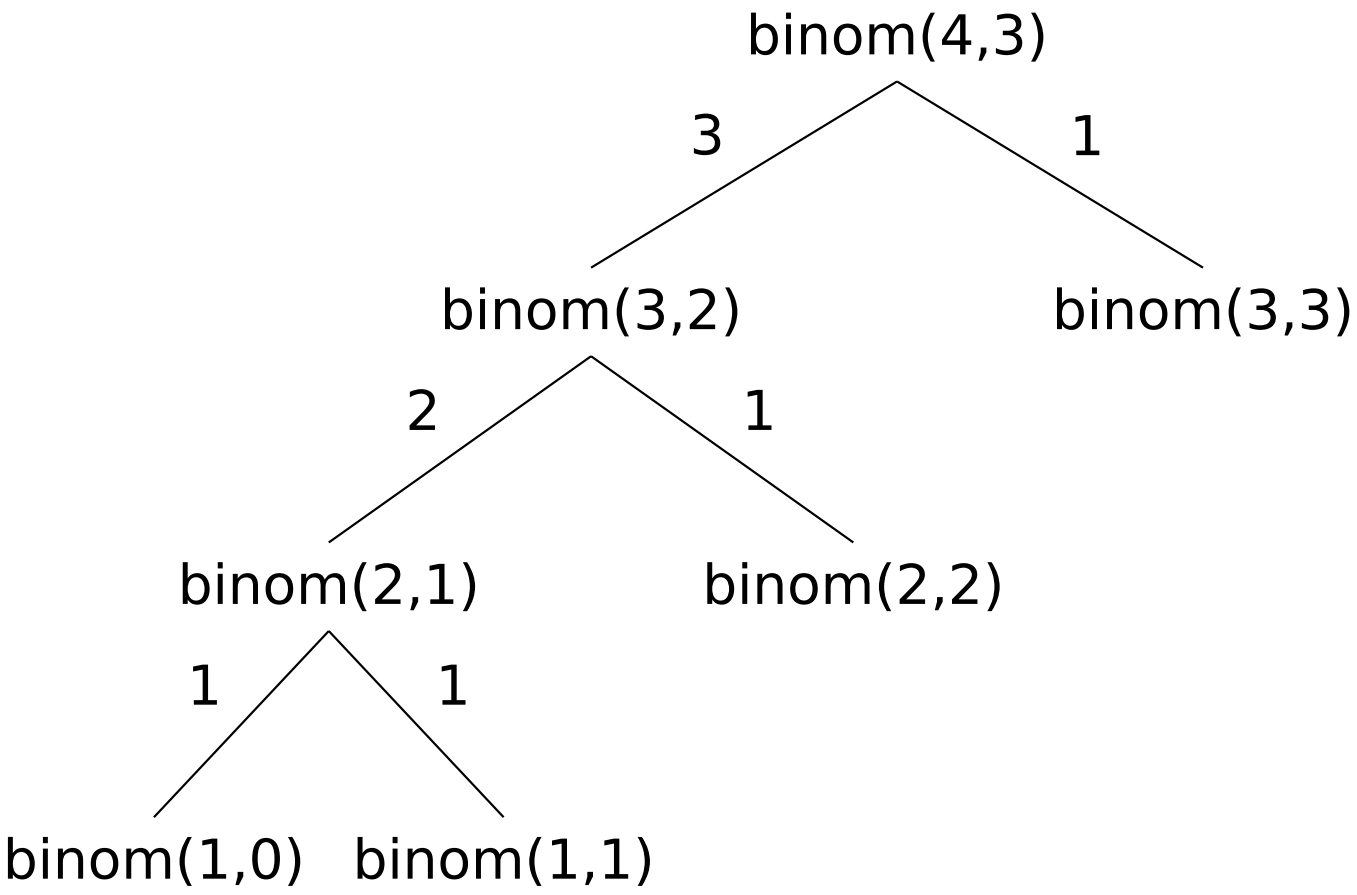
Proveu-ho!!

- Sabrieu definir el càlcul dels coeficients binomials recursivament?

```
def binom(n,k):
    if k == 0:
        return 1
    elif k == n:
        return 1
    else:
        return binom(n−1,k−1) + binom(n−1,k)
```

Traça de «binom»

Observeu la traça corresponent a la crida binom(4,3)



Subprogrames recursius i llistes I

- Hi ha moltes operacions sobre llistes (i altres seqüències) que són molt senzilles d'expressar recursivament.
- Calculem la suma dels elements d'una llista:

```
def suma(l):
    if l == []:
        return 0
    else:
        return l[0] + suma(l[1:])
```

- Si l és una llista no buida a l[0] se l'anomena cap i a l[1:] cua. En el cas d'una llista d'un sol element, la cua és la llista buida.

Subprogrames recursius i llistes II

- Calculem el màxim dels elements d'una llista de longitud superior a zero:

```
def maxim(l):
    if len(l) > 1:
        e = l[0]
        m = maxim(l[1:])
        if e > m:
            return e
        else:
            return m
    else:
        return l[0]
```

Un problema més complicat amb llistes I

- Assumiu que l és una llista d'enters i llistes que, a la seva vegada, són llistes d'enters i llistes i així successivament. Per exemple, l=[2,3,[2,1,[2,3],[1]],[2,3]].
- Volem dissenyar una funció tal que, donada una llista com l'anterior, calculi la suma dels seus elements. Fixeu-vos que aqui, la funció pre-definida **sum**() no serveix de res. . .
- Estudiem amb quins casos base ens podem trobar:
  - Si un element és la llista buida, la suma dels seus elements és 0.
  - Si un element és un enter, la suma és el valor de l'enter.
  - Si un element és una llista de més d'un element, la suma dels seus elements serà igual a la suma del primer (cap) més la suma de la resta (cua).

Un problema més complicat amb llistes II

- Si traslladem això a Python, tenim:

```
def suma(l):
    if isinstance(l,int):
        return l
    elif not l:
        return 0
    else:
        return suma(l[0]) + suma(l[1:])
```

Estructura d'un subprograma recursiu

- Els subprogrames recursius sempre tenen la mateixa estructura:
  - Un o més casos base, que són casos trivials de solució immediata. És el cas del factorial de zero.
  - Un o més casos recursius, que son casos on la solució del problema s'expressa en funció de les solucions del mateix problema aplicat a casos més petits.
- Per assegurar que acaba correctament cal garantir que a cada crida recursiva el problema que es tracta és de talla menor. Tècnicament, hem de poder associar al problema una funció afitada inferiorment que decreix a cada crida recursiva.
- Per entendre o dissenyar un subprograma recursiu intentar resseguir la seva execució és una mala estratègia. És molt més interessant treballar per entendre la definició.

Recursivitat i iteració I

- Es pot demostrar que tot subprograma recursiu també pot escriure's de forma iterativa i viceversa. En el cas del factorial:

```
def fact(n):
    if n>0:
        return n*fact(n-1)
    else:
        return 1
```

```
def fact(n):
    f = 1
    for i in range(2,n+1):
        f *= i
    return f
```

- Així, recursivitat i iteració son dues formes diferents d'expressar el mateix algoritme.
- L'expressió recursiva és molt més abstracta i compacta. Hi ha molts problemes en els que l'expressió recursiva és més simple i natural que la iterativa. La recursivitat és una eina molt potent per raonar.

Recursivitat i iteració II

- Hi ha llenguatges de programació que no admeten la recursivitat i d'altres en que no s'admeten les iteracions i només tenen recursivitat.
- La recursivitat té un paper fonamental a la teoria de la informàtica.

La cara fosca de la recursivitat

- Les implementacions recursives presenten alguns problemes.
  - Són generalment més costoses d'executar. Cada crida recursiva que es fa consumeix nous recursos: crea noves variables locals diferents de les anteriors.
  - Poden repetir càlculs més d'una vegada. Fixeu-vos en el cas dels coeficients binomials.
  - Cal que els suprogrames recursius siguin funcions pures i no modifiquin els arguments corresponents. S'ha de tenir cura amb els tipus mutables.
- Tot i això són una eina de treball irrenunciable quan es tracta d'abordar problemes complicats.

Feina per aquesta setmana

1. Estudi de la teoria. A partir de les transparències i les notes que heu pres.
2. Incrementar el xuletari i el glossari del tema.
3. Solució dels problemes del tema.