Pràctica Estructures de la informació

Codificador - Fase 2

Primer package genèric

Arbres

Biel Moyà

2013-2014

Un cop tenim realitzada la taula de freqüències (Fase 1) que serà l'*input* per construir el nostre codi, es fa necessari realitzar una explicació global del problema per entrendre les pases que realitzam:

Es desitja fixar un sistema de codificació de longitud variable, de tal manera que els caràcters que apareixen amb més freqüència tinguin codis de menys bits. Per tal que no hi hagi ambigüitats, cap caràcter pot tenir una codificació que sigui el prefix de la codificació d'un altre caràcter, aquest problema es pot resoldre de manera senzilla construint un arbre binari on cada node fulla representa un dels símbols a codificar (veure Figura 1).

Es demana:

- Donat programa que rep el nom d'un fitxer de text (*fname*) com a paràmetre d'entrada, construir una taula amb les freqüències d'aparició de cada caràcter en el fitxer.
- Generar n arbres que contenen un sol símbol i tenen com a probabilitat associada la d'aquell símbol.
- Realitzar un procés iteratiu en el que es prenen els dos arbres amb les probabilitats més baixes i unificar-los en un de sol, que tindrà com a probabilitat associada la suma de probabilitats dels seus dos components. Aquest procés acabarà en tenir un únic arbre.
- Generar les estructures que ens facilitin els processos de codificació i decodificació.
- En finalitzar l'execució del programa obtindrem dos fitxers: un on guardarem l'estructura que ens permet codificar (*fname.co*) i un altre on tenim l'estructura que ens permet decodificar (*fname.de*)

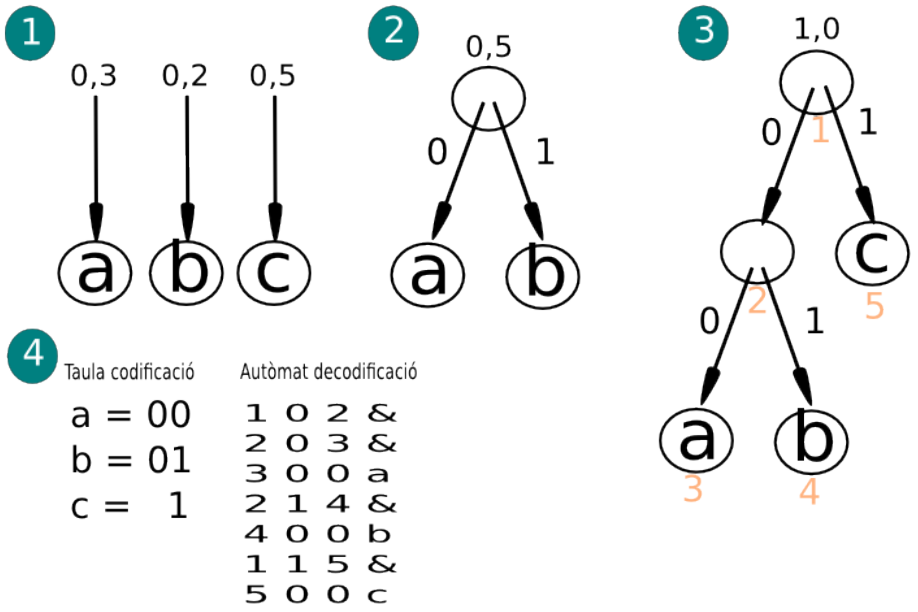

Figura 1: Esquema general

Arbre binari

En aquest cas que ens ocupa, els nostres arbres tenen la particularitat que han de guardar una informació més: la suma de les probabilitats dels caràcters que penjen d'ells. Això també es veu reflectit en certs canvis en les operacions.

En aquesta segona fase es demana determinar l'estructura de l'arbre i implementar les diferents operacions recollides en el fitxer d'especificació *arbreb.ads*

Especificació

```
generic
    type item is private;
package arbre_b is

        -- FER: Declaracio de l'arbre

    bad_use: exception;
    space_overflow: exception;

    procedure buit      (t: out arbre);
    function es_buit    (t: in arbre) return boolean;
    procedure arrel     (t: in arbre; tnd : out tipusdenode);
    procedure construeix(t: out arbre; lr, rt: in arbre; f: out ↩
```

```
        float);
    procedure construeix(t: out arbre; x: in item; f: in float);
    procedure esquerra  (t: in arbre; lt: out arbre);
    procedure dreta     (t: in arbre; rt: out arbre);
    function "<" (x1, x2: in arbre) return boolean;
    function ">" (x1, x2: in arbre) return boolean;

    end arbre_b;
```

Consideracions

- L'arbre ha de permetre emmagatzemar qualsevol estructura, per tant ha de ser genèric.
- Per qüestions de simplicitat es permet que la declaració de l'estructura sigui pública.
- En els apunts de classe podeu trobar una implementació molt similar a la que es demana.
- Aquestes operacions són les habituals en la construcció i recorregut d'arbres binaris, en el cas de necessitar altres operacions en fases futures, es poden implementar i/o modificar segons les vostres necessitats.
- En aquest punt, seria convenient afegir les excepcions necessàries (veure codi dels apunts).