Licence Info
Algorithmique Avancée
TD 6 : novembre 2006
Codage de structures complexes par des entiers

1. Les fonctions code et decode (décrites en cours et données ici en forme Exalgo) effectuent le codage et décodage pour arbres (binaires) d'une taille fixe. Pour traiter tous les arbres binaires jusqu'á la taille *max*, on peut décider d'ordonner tous ces arbres d'abord par leur taille et ensuite dans l'ordre utilisé dans ces fonctions Exalgo.

```
fonction code (ref racine: arbreptr): entier;
var i:entier;
var somme:entier;
debut /* code parmi ceux de la meme taille*/
        si racine = null alors retourne 0 finsi;
        /* compter ceux de la meme taille mais avec un sous-arbre gauche
        plus petit*/
        somme:=0;
        pour i allant de 0 a taille(racine^.gauche)-1 par pas de 1 faire
                somme := somme+T[i]*T[taille(racine)-1-i]
        finpour;
        /*et enfin ceux avec les memes tailles des deux sous-arbres mais qui
        precedent * "racine" dans l'ordre*/
        somme := somme+code(racine^.gauche)*T[taille(racine^.droite)];
        retourne somme+code(racine^.d)
fin
finfonction;

fonction decode(val n,c: entier): arbreptr;
        /* n la taille, c le code*/
var racine: arbreptr;
var i: entier;
debut
        i:=-1;
        si n=0 alors retourne null finsi;
        allouer(racine);
        tantque c>=0 faire i:=i+1;c:=c-T[i]*T[n-1-i] fin tantque;
        c:=c+T[i]*T[n-1-i]; /*i est la taille du sous-arbre gauche*/
        racine^.gauche:=decode(i,c/T[n-1-i]);
        racine^.droite=decode(n-1-i,c mod T[n-1-i]);
        retourne racine
fin
finfonction;
```

(a) Quelle est dans cet ordre, parmi tous les arbres de taille 0 à max, la position de l'arbre de taille n et dont le code c parmi les arbres de cette taille a été établi par la fonction code vue en cours?
Réponse: $c+$ la somme pour i de 0 à $n-1$ du nombre d'arbres de taille i.

(b) Modifier le code Exalgo de façon à utiliser cet ordre. On ajoute:

```
fonction codetous(ref racine: arbreptr): entier;
var i,somme:entier;
début
   somme:=0;
   pour i allant de 0 à taille(racine)-1 par pas de 1 faire
      somme:=somme+T[i]
   finpour;
   retourne somme+code(racine)
finfonction;

fonction decodetous(val n,c: entier): arbreptr;
var i:entier;
debut
   tant que T[i]<=c faire
      c:=c-T[n];
      n:=n+1
   fintantque;
   retourne decode(n,c)
finfonction;
```

2. Justifier la récurrence donnée dans les transparents de cours pour le nombre d'arbres de taille n avec racine et limitation du nombre de fils d'un noeud à 2: $G(n) = G(n-1)$
$+ \; somme_{i=1}^{n/2-1} G(i) * G(n-1-i) \; +$
$(si\ n\ impair) G((n-1)/2) \; + \; \begin{pmatrix} G((n-1)/2) \\ 2 \end{pmatrix}$
(à noter: arrondir vers le bas dans la division $n/2$)
Réponse:

- Le $G(n-1)$ est le nombre d'arbres avec racine et un sous-arbre (forcément de taille $n-1$)
- Le $G(i) * G(n-1-i)$ est le nombre d'arbres avec deux sous arbres de tailles différentes i et $n-1-i$ où $i < n-1-i$
- si n est impair il y a aussi la possibilité de deux sous arbres de la même taille $(n-1)/2$
- soit deux arbres identiques ($G((n-1)/2)$ possibilités)
- soit deux différents (nombre de possibilités le coefficient binomial des choix de 2 parmi $G((n-1)/2)$.

3. Dans le cas d'arbres généraux (avec l'ordre des fils d'un noeud défini), on pourrait les ordonner d'abord par leur taille, ensuite par le nombre de sous-arbres de la racine et enfin par les codes des sous-arbres;
   donner les algorithmes de codage et décodage dans ce cas.
   Réponse dans le fichier arbres.exo.

4. Décrire une méthode de codage et décodage des façons de choisir m objets parmi n. (Toujours avec les codes utilisés constituant une intervalle commençant à 0)
   Réponse: On peut supposer que les n "objets" sont les entiers 1 à n; on les ordonne d'abord selon si n est parmi les m choisis ou non et ensuite récursivement; dans le code donné on représente la combinaison avant et après codage/décodage par un tableau T de $m+1$ entiers où les éléments $T[1]$ à $T[m]$ sont les m choisis rangés en ordre croissant.
   Un programme Exalgo qui contient les fonctions et un petit test:

```
programme coeff;

type vec : tableau 0..20 de entier;
type tab : tableau 0..20 de vec;

var binom: tab;
var T:vec;

procédure init();
var i,j:entier;
debut
   pour i allant de 0 à 20 par pas de 1 faire
      binom[i][0]:=1;
      binom[i][i]:=1;
      pour j allant de 1 à i-1 par pas de 1 faire
         binom[i][j]:=binom[i-1][j-1]+binom[i-1][j]
      finpour
   finpour
finprocedure;

fonction coder(ref T:vec; val n,m:entier):entier;
debut
   selonque
      cas m=0: retourne 0
      cas T[m]=n: retourne coder(T,n-1,m-1)
      autrement: retourne binom[n-1][m-1]+coder(T,n-1,m)
   finselonque
finfonction;

procédure decoder(val c:entier; ref T:vec; val n,m:entier);
```

```
début
   printf("decoder(%i,%i,%i)\n",c,n,m);
   si m=0 alors retourne finsi;
   si c<binom[n-1][m-1] alors
      T[m]:=n;
      decoder(c,T,n-1,m-1)
   sinon
      decoder(c-binom[n-1][m-1],T,n-1,m)
   finsi
finprocedure;

var i:entier;
début
init();
pour i allant de 0 a binom[8][5]-1 par pas de 1 faire
   decoder(i,T,8,5);
   printf("vérification %i=%i\n",i,coder(T,8,5))
finpour
fin.
```