Travaux Dirigés - Corrigé
Séance nr. 5

Exercice 1: Dénombrement sur les arbres binaires

Dans cet exercice nous supposons que l'arbre binaire est non vide.

1. Quelle est le nombre maximal de feuilles d'un arbre de hauteur h.

Si h = 0, c'est-à-dire si l'arbre se réduit à sa racine, f=1. Sinon, on raisonne par récurrence : le sous-arbre gauche (resp. droit) de la racine est un arbre de hauteur h-1 et il a un nombre maximal de feuilles pour un arbre de hauteur h-1. Si on note f(h) le nombre maximal de feuilles d'un arbre de hauteur h on a donc : f(h)=1 si h=0, 2f(h-1) sinon. On a donc f(h) = 2^h. Le nombre maximal de feuilles d'un arbre de hauteur h est donc 2^h.

2. Quel est le nombre maximal de noeuds d'un arbre de hauteur h.

Si h = 0, c'est-à-dire si l'arbre se réduit à sa racine, n=1. Sinon, on raisonne par récurrence : le sous-arbre gauche (resp. droit) de la racine est un arbre de hauteur h-1 et il a un nombre maximal de noeuds pour un arbre de hauteur h-1. Si on note n(h) le nombre maximal de noeuds d'un arbre de hauteur h on a donc : n(h)=1 si h=0, 1+2n(h-1) sinon. On a donc n(h) = 2^(h+1)-1. Le nombre maximal de noeuds d'un arbre de hauteur h est donc 2^(h+1)-1.

3. Montrer que le nombre de branches vides b (nombre de fils gauches et de fils droits vides) d'un arbre à n noeuds est égal à n+1.

On distinguera les noeuds ayant zéro, un et deux fils :

- p le nombre de noeuds ayant zéro fils;
- q le nombre de noeuds ayant un fils;
- r le nombre de noeuds ayant deux fils.

On a les relations suivantes : n = p + q + r car un noeud a soit zéro, soit un, soit deux fils.

On a aussi : 0 × p + 1 × q + 2 × r = n - 1 car tous les noeuds, la racine exceptée, ont un père; on compte ces n-1 noeuds « fils » à partir de leurs pères, 0 × p sont fils d'un noeud sans fils, 1 × q sont fils d'un noeud ayant un unique fils et 2 × r sont fils d'un noeud ayant deux fils.

Enfin, on a : 2 × p + 1 × q + 0 × r = b en comptant les branches vides de chaque noeud.

En additionnant les deux dernières équations on obtient : 2(p + q + r) = b + n - 1 d'où :
2 n = b + n - 1 et finalement : b = n + 1 en utilisant la première équation.

4. Montrer que le nombre de feuilles d'un arbre est égal au nombre de noeuds de degré deux, plus un.

On se sert des relations entre p, q et r de la question précédente : p + q + r = n et q + 2r = n-1. En éliminant q entre les deux équations on obtient : p = r + 1 qui est le résultat recherché.

Exercice 2: Implantation d'un arbre binaire

On utilise la structure de données suivante:

```
typedef struct s_noeud {
        int r;
        struct s_noeud *fg, *fd;
} noeud, *abin;
```

Proposer les opérations spécifiées en cours: enraciner, sous-arbre gauche et droit, racine et vide.

```
abin anouv() { return NULL; }

abin enrac(int x, abin g, abin d)
{
        abin n = (abin)malloc(sizeof(noeud));
        n->r = x;
        n->fg = g;
        n->fd = d;
        return n;
}

abin ag(abin a)
{
        if (a!=NULL) return a->fg;
        return NULL;
}

abin ad(abin a)
{
        if (a!=NULL) return a->fd;
        return NULL;
}

int racine(abin a) { return a->r; } // if not vide(a)
boolean vide(abin a) { return a==NULL; }
```

Exercice 3: Parcours d'un arbre binaire

Compléter les opérations précédentes par: nombre de noeuds, nombre de feuilles, hauteur, longueur de cheminement interne / externe, existence d'un élément.

```
int nnoeuds(abin a)
{
        if (a==NULL) return 0;
        return 1+nnoeuds(a->fg)+nnoeuds(a->fd);
}
```

```
int nfeuilles(abin a)
{
        if (a==NULL) return 0;
        if (a->fg==NULL && a->fd==NULL) return 1;
        return nfeuilles(a->fg)+nfeuilles(a->fd);
}

int hauteur(abin a)
{
        if (a==NULL) return 0;
        int hg = hauteur(a->fg);
        int hd = hauteur(a->fd);
        return 1+(hg>hd?hg:hd);
}

int lce_aux(abin a, int hh)
{
        if (a->fg == NULL && a->fd == NULL) return hh;
        int lceg = 0; if (a->fg != NULL) lceg = lce_aux(a->fg, hh + 1);
        int lced = 0; if (a->fd != NULL) lced = lce_aux(a->fd, hh + 1);
        return lceg + lced;
}
int lce(abin a) { if (a == NULL) return 0; else return lce_aux(a, 1); }

int lci_aux(abin a, int hh)
{
        if (a->fg == NULL && a->fd == NULL) return 0;
        int lcig = 0; if (a->fg != NULL) lcig = lci_aux(a->fg, hh + 1);
        int lcid = 0; if (a->fd != NULL) lcid = lci_aux(a->fd, hh + 1);
        return hh + lcig + lcid;
}
int lci(abin a) { if (a == NULL) return 0; else return lci_aux(a, 1); }

bool existe(abin a, int x)
{
        if (a==NULL) return false;
        if (a->r==x) return true;
        return existe(a->fg,x) || existe(a->fd,x);
}
```