Algorithmique
Correction Contrôle n° 1

Info-Spé – Epita

7 déc. 2009

Solution 1 **(Hachages – *9 points*)**

1. Le tableau 1 représente les valeurs de hachage associées à l'ensemble E suivant :
   E={beck, cale, clapton, hendrix, hooker, king, richards, vaughan, winter, young}

2. Représentation des structures de données dans les cas suivants :

   (a) *Hachage linéaire avec un coefficient de décalage $d = 3$: voir tableau 2.*

Tab. 1 – Valeurs de hachage

beck	21 mod 11	10
cale	21 mod 11	10
clapton	81 mod 11	4
hendrix	82 mod 11	5
hooker	72 mod 11	6
king	41 mod 11	8
richards	80 mod 11	3
vaughan	74 mod 11	8
winter	89 mod 11	1
young	82 mod 11	5

Tab. 2 – Hachage linéaire

0	vaughan
1	
2	cale
3	richards
4	clapton
5	hendrix
6	hooker
7	winter
8	king
9	young
10	beck

(b) *Hachage avec chaînage séparé (voir figure 1),*

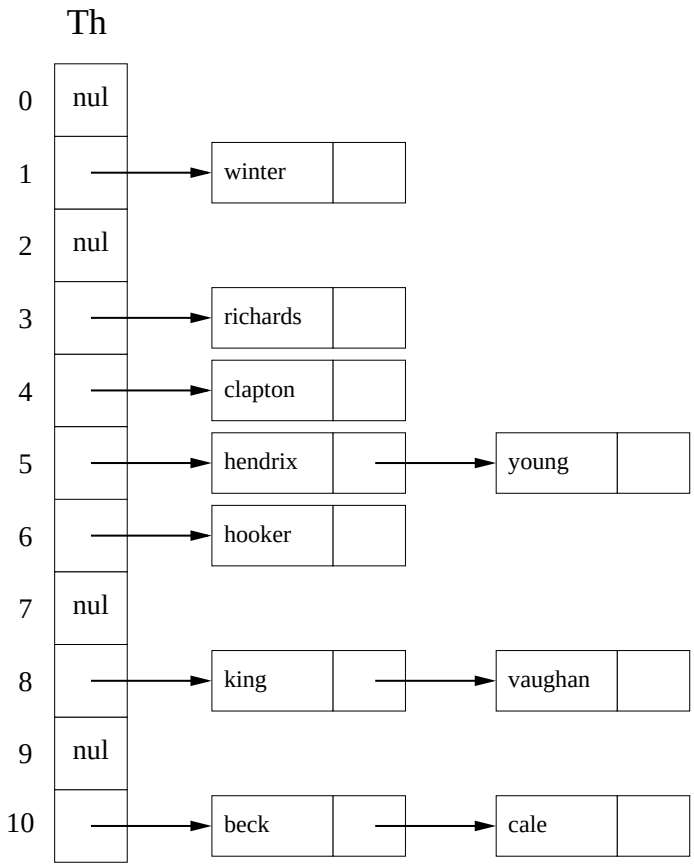


FIG. 1 – Hachage avec chaînage séparé.

3. *Principe :*

Tout d'abord, il faut calculer dans v la valeur de hachage de l'élément recherché *h(x)*.
Ensuite, tant que l'emplacement v du même tableau de hachage n'est pas vide, et que l'élément du tableau de hachage correspondant à ce calcul ($Th[v]$) est strictement inférieur à x, et que l'on a pas effectué les m essais successifs, on applique le principe du hachage linéaire, à savoir $v \leftarrow v \oplus 1$ (ce qui correspond pour une première valeur à 1 à faire $v \leftarrow (v+1)\ mod\ m$).

Si l'on a quitté la boucle, soit :
– parce que l'on était sur une case vide,
– parce que l'on a effectué les m essais successifs sans succès,
– parce que la valeur de la case $th[v]$ était supérieure au x recherché,
alors on retourne *Faux*.

En revanche, si c'est parce que la valeur de la case $th[v]$ était égale à x, alors on retourne *Vrai*.

Algorithme :

L'algorithme correspondant à ce principe est le suivant :

```
Algorithme fonction rechercher_HO : Booléen
   Paramètres locaux
     t_element x
     t_hachage th
   Variables
     entier  v, i

Debut

   v ← h(x)       /* calcul de la valeur de hachage primaire */
   i ← 1          /* compteur d'essais */

    Tant que (i<=m)  Faire
        Si (th[v]>=x)  ou (estvide(th, v))  Alors
           SortieBoucle
        Fin si
       v ← (v+1) mod m
       i ← i+1
    Fin tant que

    Si Th[v]=x  Alors      /* l'élément est présent */
       Retourne (Vrai)
    Fin si

   Retourne(Faux)              /* dans tous les autres cas */

Fin Algorithme Fonction rechercher_HO
```

Solution 2 **Mesure sur les arbres *2-3-4* – *11 points***

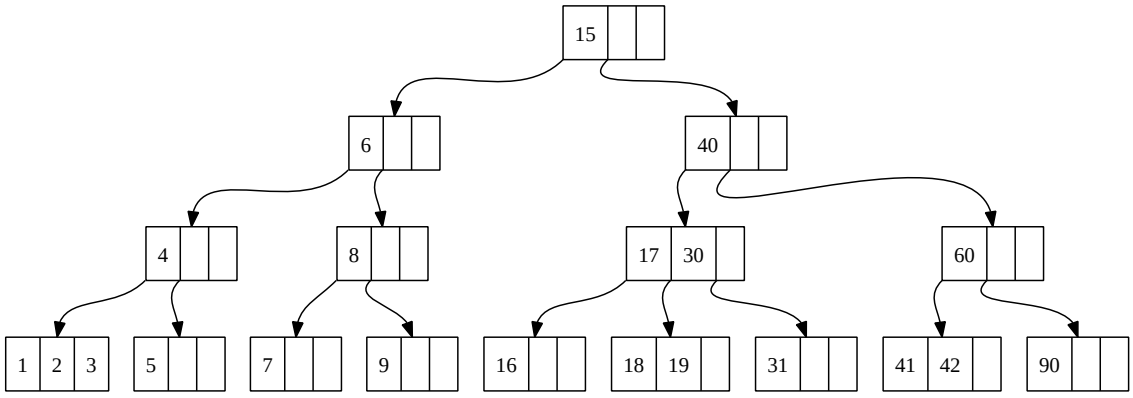
1. Il suffit de calculer le rapport entre la taille de l'arbre en clefs et la taille de l'arbre en nœud.

2. **Principe :** on va calculer (en un seul parcours) les deux tailles : la procédure récursive `occup_rec` va accumuler dans ses deux paramètres globaux les deux tailles.

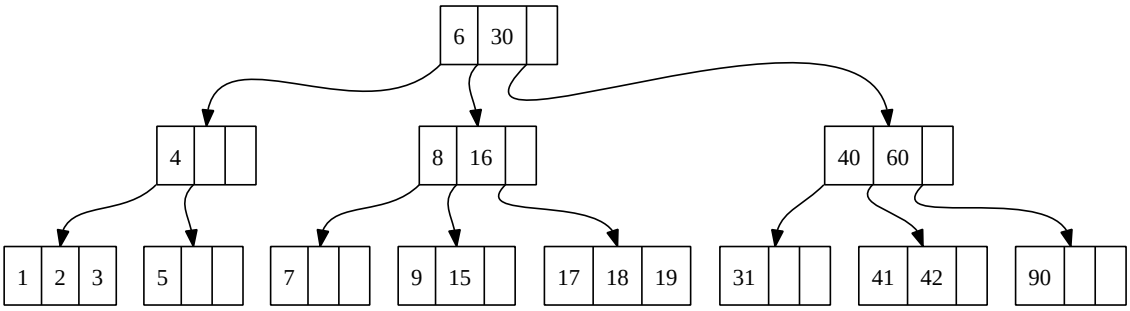
```
algorithme procedure occup_rec
    parametres locaux
        t_a234                  A
    parametres globaux
        entier                  tn, tc
    variables
        entier                  i
debut
    si (A <> NUL) alors
        tc ← (tc + A↑.nbcles)
        tn ← (tn + 1)
        pour i ← 1 jusqu'a (A↑.nbcles + 1) faire
            occup_rec(A↑.fils[i], tn, tc)
        fin pour
    fin si
fin algorithme procedure occup_rec
```

```
algorithme fonction occupation : reel
    parametres locaux
        t_a234                  A
    variables
        entier                  tn, tc
        reel                    r
debut
    si (A = NUL) alors
        retourne (0)
    fin si
    tn ← 0
    tc ← 0
    occup_rec(A, tn, tc)
    r ← tc
    r ← (r / tn)
    retourne (r)
fin algorithme fonction occupation
```

3. Avec Éclatement :



4. Avec Rotation :



5. Nombre moyen de clefs par nœud :
   Avant insertion : $18/12 = 1.5$
   Après insertion avec éclatement : $21/16 = 1.3125$
   Avant insertion avec rotation : $21/12 = 1.75$