Algorithmique
Correction Contrôle n° 1

S3 – Epita

10 Nov. 2014 - 10 :00

Solution 1 **(Quelques résultats différents –** ***6 points*****)**

Représentations des tables de hachages en cas de :

1. hachage coalescent :

Figure 1 – Hachage coalescent

0	5	-1
1	20	-1
2	16	0
3	39	-1
4	11	2
5	44	10
6	94	3
7	12	8
8	23	-1
9	13	-1
10	88	4

2. hachage linéaire :

Figure 2 – Hachage linéaire

0	11
1	39
2	20
3	5
4	16
5	44
6	88
7	12
8	23
9	13
10	94

3. double hachage :

Figure 3 – Double hachage

0	11
1	23
2	20
3	16
4	39
5	44
6	94
7	12
8	88
9	13
10	5

Solution 2 **(Arbres Généraux : sérialisation – *6,5 points*)**

1.

1	2	3	4	5	6	7	8	9	10	11	12
3	3	10	3	3	10	8	9	10	−1	7	7

2.

```
algorithme procedure parent_from_tuples
  parametres locaux
    t_arbre_nuplet   T
  parametres globaux
    t_vect_entiers   parent

  variables
    entier i
debut
  parent[T↑.cle] ← -1
  pour i ← 1 jusqu'a T↑.nbFils faire
    parent_from_tuples (T↑.fils[i], parent)
    parent[T↑.fils[i]↑.cle] ← T↑.cle
  fin pour
fin algorithme procedure parent_from_tuples
```

3.

```
algorithme procedure parent_from_dyn
  parametres locaux
    t_arbre_dyn   T
  parametres globaux
    t_vect_entiers   parent

  variables
    t_arbre_dyn   child
debut
  parent[↑.cle] ← -1
  child ← T↑.fils
  tant que (child <> NUL) faire
    parent_from_dyn (child, parent)
    parent[child↑.cle] ← T↑.cle
    child ← child↑.frere
  fin tant que
fin algorithme procedure parent_from_dyn
```

Solution 3 **(B-tree or not B-tree... – *6 points*)**

Spécifications :

La fonction `test_Btree (t_Btree` B`, t_element` inf`,` sup`)` vérifie si l'arbre B est bien "ordonné" avec ses valeurs dans l'intervalle $]inf, sup[$.

```
algorithme fonction test_Btree : booleen
  parametres locaux
    t_Btree   B
    t_element inf, sup

  variables
    entier    i
debut
  si (B↑.cles[1] <= inf) ou (B↑.cles[B↑.nbcles] >= sup) alors
    retourne faux
  sinon
    pour i ← 1 jusqu'a B↑.nbcles-1 faire
      si B↑.cles[i] >= B↑.cles[i+1] alors
        retourne faux
      fin si
    fin pour
    si B↑.fils[1] = NUL alors
      retourne vrai
    sinon
      pour i ← 1 jusqu'a B↑.nbcles faire
        si non test_Btree (B↑.fils[i], inf, B↑.cles[i]) alors
          retourne faux
        fin si
        inf ← B↑.cles[i]
      fin pour
      retourne test_Btree (B↑.fils[B↑.nbcles+1, inf, sup)
    fin si
  fin si
fin algorithme fonction test_Btree
```

Appel :

```
retourne (B = NUL) ou test_Btree (B, −∞, +∞)
```

Figure 4 – Après suppression

Solution 4 **(B-arbre : suppression – *1,5 points*)**

1. L'arbre de l'énoncé est un B-arbre de degré minimal (ordre) 3.

2. Après suppresion de la valeur 15, avec le principe "à la descente" :