

Algorithmique

Correction Contrôle n° 3 (C3)

INFO-SPÉ S3# – EPITA

6 mars 2018 - 14 : 15

Solution 1 (Hachages – 2 points)

Hachage avec chaînage séparé (voir figure 1),

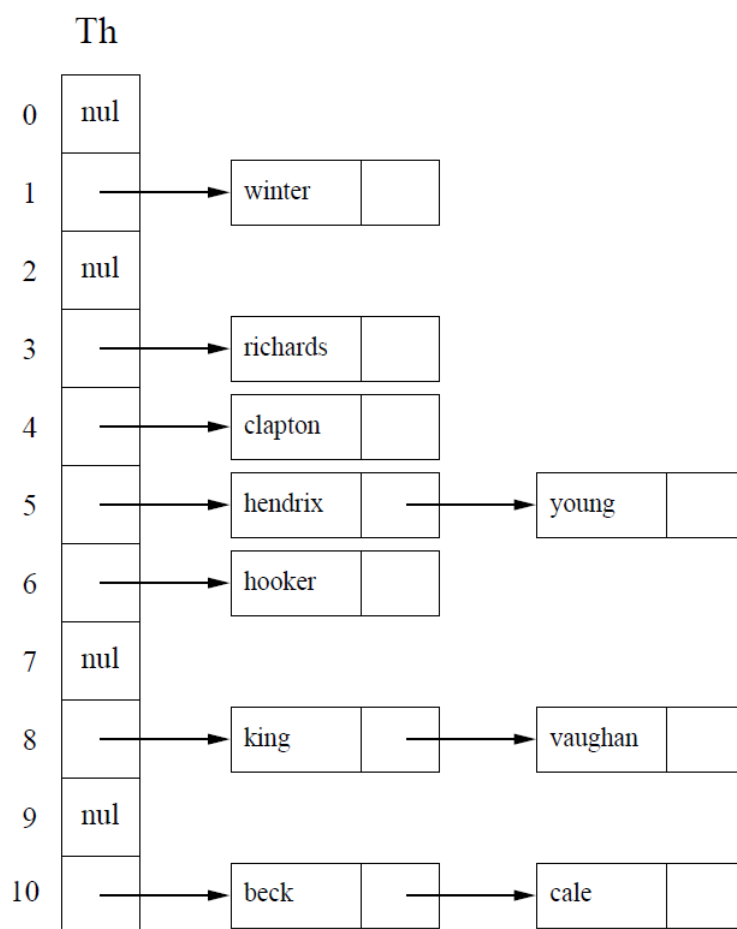


FIGURE 1 – Hachage avec chaînage séparé.

Solution 2 (Hachage : Tableaux valides – 3 points)

Les tableaux ne pouvant pas résulter d’une insertion quelconque des clés sont : A-B-D

Le seul valable est le C qui pourrait correspondre à la séquence d’ajout {B, E, A, C, D, F, G} (il y en a d’autres).

Solution 3 (Arité moyenne d'un arbre général – 5 points)

Spécifications :

La fonction `average_arity(T)` retourne l'arité moyenne de l'arbre T (`TreeAsBin`).

```
1
2 """
3 arity(B)  return (nb links, nb internal nodes)
4 """
5
6 def arity(B):
7     '''
8     with "classical" traversal
9     '''
10    if B.child == None:
11        return (0, 0)
12
13    else:
14        (links, nodes) = (0, 1)
15        child = B.child
16        while child:
17            (l, n) = arity(child)
18            links += l + 1
19            nodes += n
20            child = child.sibling
21
22        return (links, nodes)
23
24
25
26
27 def arity(B):
28     '''
29     "binary" traversal
30     '''
31    if B.child == None:
32        (links, nodes) = (0, 0)
33    else:
34        (l, n) = arity(B.child)
35        (links, nodes) = (l + 1, n + 1)
36
37    if B.sibling != None:
38        (l, n) = arity(B.sibling)
39        links += l + 1
40        nodes += n
41
42    return (links, nodes)
```

```
1
2 def average_arity(B):
3     (links, nodes) = arity(B)
4     return links / nodes if nodes else 0
```

Solution 4 (B-arbres : Représentation linéaire – 5 points)

Spécifications :

La fonction `btree2list(B)` retourne la représentation linéaire (de type `str`) de B s'il est non vide, la chaîne vide sinon.

```

1  def __BtreeToList(B):
2      s = '(<'
3      for i in range(B.nbKeys-1):
4          s += str(B.keys[i]) + ', '
5      s += str(B.keys[B.nbKeys-1]) + '>'
6      for child in B.children:
7          s += __BtreeToList(child)
8      s += ')'
9      return s
10
11 def BtreeToList(B):
12     if B:
13         return __BtreeToList(B)

```

Solution 5 (B-Arbres : insertions – 2 points)

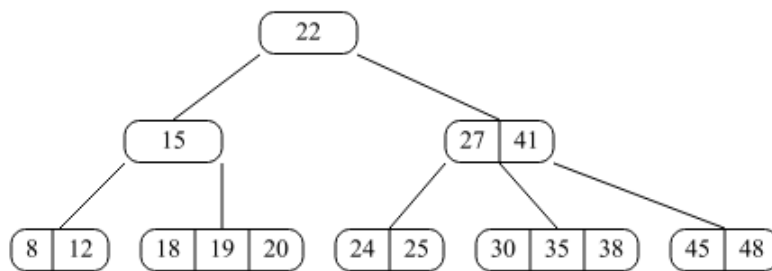


FIGURE 2 – B-arbre d'ordre 2

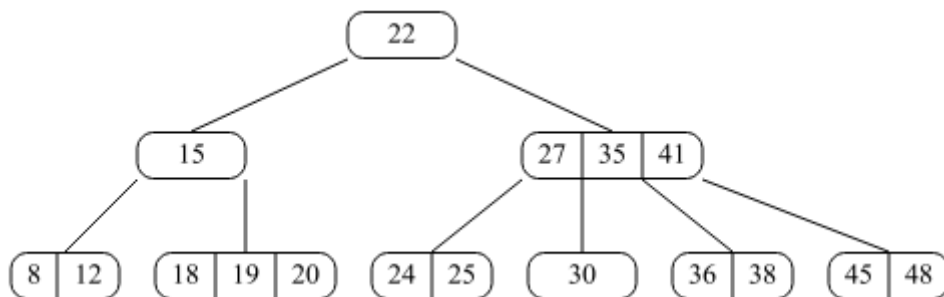


FIGURE 3 – Après insertion de 36

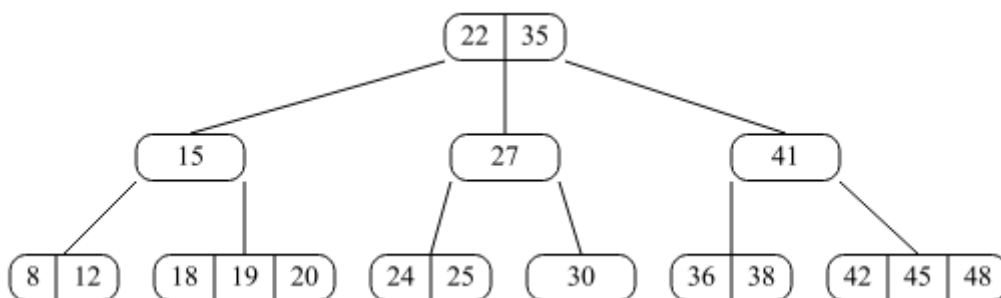


FIGURE 4 – Après insertion de 42

Solution 6 (What ? – 3 points)

1. Résultats :

- (a) `what(B_3 , 2)` retourne 5
- (b) `what(B_3 , 7)` retourne 13
- (c) `what(B_3 , 18)` retourne 20
- (d) `what(B_3 , 39)` retourne 40
- (e) `what(B_3 , 41)` retourne 42
- (f) `what(B_3 , 99)` retourne `None`

2. La fonction `what(B, x)` retourne la clé de B immédiatement supérieure à x. La fonction renvoie `None` si une telle valeur n'existe pas.