

Algorithmique

Correction Contrôle n° 3 (C3)

INFO-SPÉ (S3#) – EPITA

6 avril - 10h

Solution 1 (Quelques questions – 4 points)

1. Les méthodes de hachage de base sont : extraction, compression, division, multiplication.
2. Le hachage linéaire ou le double hachage.
3. Le hachage avec chaînage séparé. En effet les éléments sont chaînés entre eux à l'extérieur du tableau.
4. La recherche par intervalle est incompatible avec la dispersion des éléments effectuée par le hachage.

Solution 2 (Sérialisation – 4 points)

1. Le vecteur de pères :

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. La fonction `buildParentVect(T, n)` :

```
1     def __buildParentVect(T, P):
2         if T != None:
3             C = T.child
4             while C != None:
5                 P[C.key] = T.key
6                 __buildParentVect(C, P)
7                 C = C.sibling
8
9     def buildParentVect(T, n):
10        P = [-1] * n
11        __buildParentVect(T, P)
12        return P
13
```

Solution 3 (Représentation par listes – 5 points)

```
1     def __fromList(s, i=0):
2         i += 1
3         key = ""
4         while s[i] not in "()":
5             key += s[i]
6             i += 1
7         T = Tree(int(key))
8         T.children = []
9         while s[i] != ')':
10            (Child, i) = __fromList(s, i)
11            T.children.append(Child)
12        i = i + 1
13        return (T, i)
14
15    def fromList(L):
16        (T, _) = __fromList(L)
17        return T
```

Solution 4 (B-arbre : suppression – 2 points)

1. L'arbre de l'énoncé est un B-arbre de degré minimal (ordre) 3.
2. Après suppression de la valeur 72, avec le principe "à la descente" :

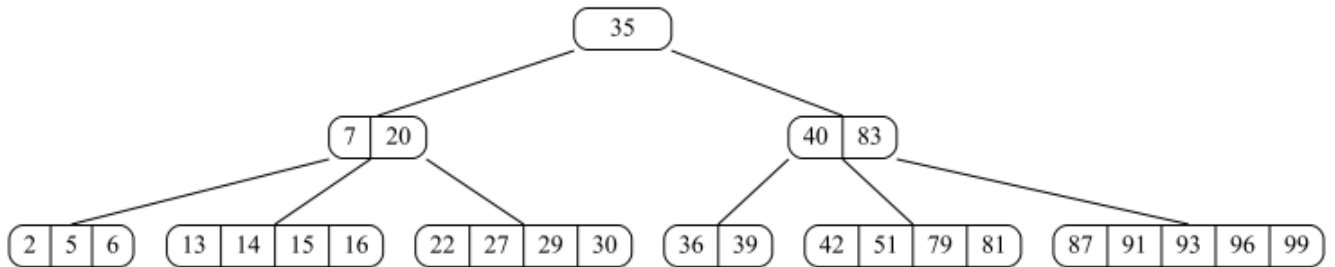


FIGURE 1 – Après suppression

Solution 5 (Au suivant – 5 points)

```

1     def findNextNath(B, x):
2         vNext, i = None, 0
3         while i < B.nbKeys and B.keys[i] <= x:
4             i += 1
5         if B.children == []:
6             if i < B.nbKeys:
7                 return B.keys[i]
8             else:
9                 return None
10        else:
11            vNext = findNextNath(B.children[i], x)
12            if vNext != None:
13                return vNext
14            elif i < B.nbKeys:
15                return B.keys[i]
16            else:
17                return None
18
19        # Student's
20        def findNextStu(B, x, y = None):
21            i = 0
22            while i < B.nbKeys and B.keys[i] <= x:
23                i += 1
24            if i < B.nbKeys:
25                y = B.keys[i]
26            if B.children == []:
27                return y
28            else:
29                return findNextStu(B.children[i], x, y)

```