Algorithmique
Correction Contrôle n° 3 (C3)

Info-spé - S3 – Epita

9 novembre 2020 - 13 : 30

Solution 1 **(Quelques résultats différents – *5 points*)**

Représentations des tables de hachages en cas de :

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 :

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 :

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

***Solution 2* (Cherche la somme – *4 points*)**

Spécifications :

La fonction `find_sum(`B`, sum)` vérifie s'il existe une branche dans l'arbre B (`TreeAsBin`) dont la somme des valeurs (entières) est *sum*.

```
def find_sum_tab(B, sum, s=0):
    if B.child == None:
        return s + B.key == sum
    else:
        C = B.child
        while C:
            if find_sum(C, sum, s + B.key):
                return True
            C = C.sibling
        return False
```

Using the "binary structure" :

```
def find_sum_bin(B, sum, s=0):
    if B.child == None:
        if s + B.key == sum:
            return True
    else:
        if find_sum_bin(B.child, sum, s + B.key):
            return True
    return B.sibling != None and find_sum_bin(B.sibling, sum, s)
```

***Solution 3* (Gap maximum – *4 points*)**

Spécifications :

La fonction `maxgap(`B`)` calcule le gap maximum du B-arbre B.

```
# optimised version: searching in all children is useless,
# first and last child are sufficient!

      def __maxgap(B):
          gap = 0
          for i in range(B.nbkeys-1):
              gap = max(gap, B.keys[i+1] - B.keys[i])
          if B.children:
              gap = max(gap, __maxgap(B.children[0]))
              gap = max(gap, __maxgap(B.children[-1]))
          return gap

# less optimized...

      def __maxgap2(B):
          gap = 0
          for i in range(B.nbkeys-1):
              gap = max(gap, B.keys[i+1] - B.keys[i])

          for child in B.children:
              gap = max(gap, __maxgap2(child))
          return gap

# call function:
      def maxgap(B):
         if B == None:
            return 0
         else:
            return __maxgap(B)
```

Solution 4 **(What ? –** ***4 points*****)**

1. *Résultats des applications :*

`what(`B_3`, 2)`	`what(`B_3`, 7)`	`what(`B_3`, 18)`	`what(`B_3`, 39)`	`what(`B_3`, 41)`	`what(`B_3`, 99)`
5	13	20	40	42	`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.

Solution 5 **(B-arbre : insertion et supression –** ***3 points*****)**

1. Après insertion de la valeur 39, avec le principe "à la descente" :

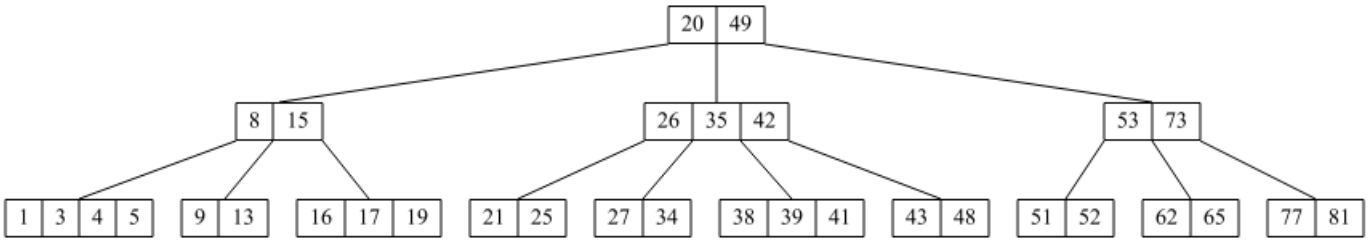

FIGURE 1 – Après insertion

2. Après suppression de la valeur 72, avec le principe "à la descente" :

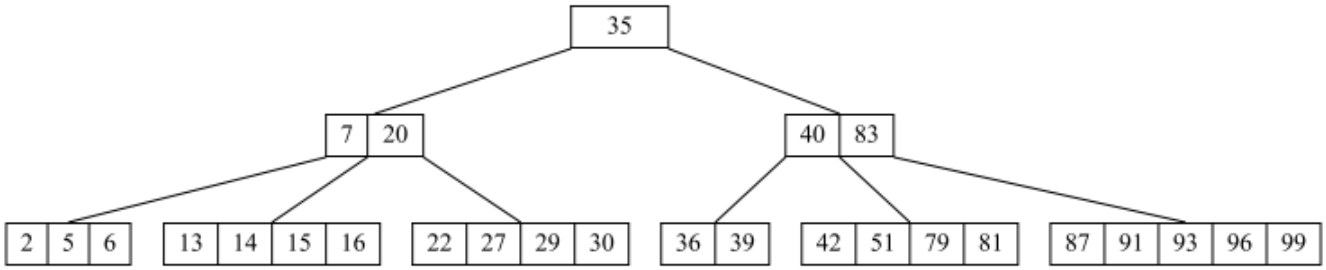

FIGURE 2 – Après suppression