Algorithmics
Correction Midterm #3 (C3)

Undergraduate 2^{nd} year - S3 – Epita

9 November 2020 - 13 : 30

Solution 1 **(Some different results –** ***5 points*****)**

Showing of the hash tables in the cases of:

1. Coalesced hashing

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. Linear probing :

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

3. Double hashing

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

Solution 2 **(Find the sum – *4 points*)**

Specifications:

The function `find_sum(`B`, `sum`)` tests if there exists a branch in the tree B (`TreeAsBin`) such that the sum of its values (integers) is equal to 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 **(Maximum Gap – *4 points*)**

Specifications:

The function `maxgap(`B`)` computes the maximum gap of the B-tree 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. *Application results:*

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. The function `what(B, x)` returns the nearest key bigger than `x` in `B`. The function returns `None` if such a key does not exists.

Solution 5 **(B-tree: insertion and deletion – *3 points*)**

1. After the insertion of the value 39, using the "in going down" principle:

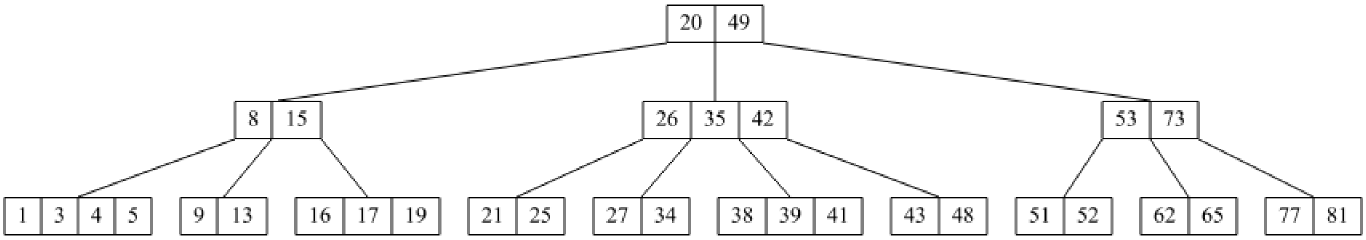

Figure 1: Après insertion

2. After the deletion of the value 72, using the "in going down" principle:

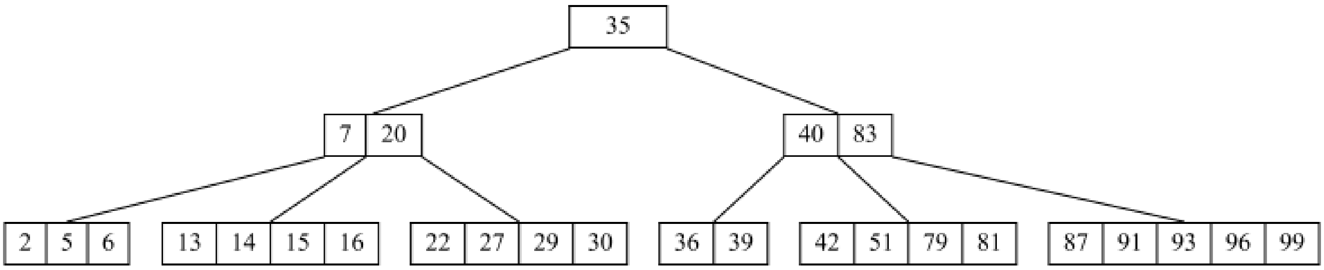

Figure 2: Après suppression