Algorithmique
Correction Contrôle n° 3 (C3)

Info-spé - S3 – Epita

5 novembre 2019 - 9 : 30

Solution 1 **(Haches et graphes... – *5 points*)**

1. Le hachage avec chaînage séparé et le hachage coalescent.
2. La méthode de résolution des collisions faisant apparaître des collisions secondaires est le **hachage coalescent**.
3. Une collision secondaire est une collision sans coïncidence de valeur de hachage entre un x différent d'un y.
4. L'ordre d'un graphe orienté est son nombre de sommets.
5. Un sommet de degré nul est appelé **sommet isolé**.
6. Les sommets de `G` de demi degré extérieur égal à 0 sont : **{6,9}**
7. Les sommets de `G` de demi degré intérieur égal à 1 sont : **{2,7,8}**

Solution 2 **(Arité moyenne d'un arbre général – *5 points*)**

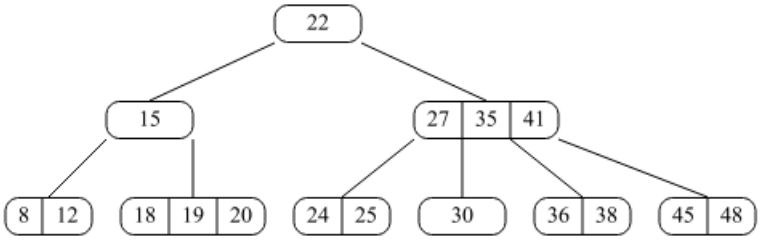
Spécifications :
La fonction `average_arity(`T`)` retourne l'arité moyenne de l'arbre T (`TreeAsBin`).

```
"""arity(B)  return (nb links, nb internal nodes)
"""
def arity(B):  # with "classical" traversal
    if B.child == None:
        return (0, 0)
    else:
        (links, nodes) = (0, 1)
        child = B.child
        while child:
            (lc, nc) = arity(child)
            links += lc + 1
            nodes += nc
            child = child.sibling
        return (links, nodes)

def arity(B):  # "binary" traversal
    if B.child == None:
        (links, nodes) = (0, 0)
    else:
        (lc, nc) = arity(B.child)
        (links, nodes) = (lc + 1, nc + 1)
    if B.sibling != None:
        (ls, ns) = arity(B.sibling)
        links += ls + 1
        nodes += ns
    return (links, nodes)

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

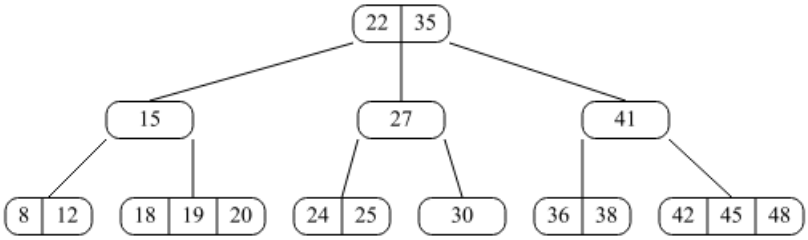
Solution 3 **(B-Arbres : insertions –** ***8 points*****)**

1. *Insertions des clés 36 et 42 :*

Après insertion de 36



Après insertion de 42



2. **Spécifications :**
   La fonction `__insert(`B`, `x`)` insère la clé x dans le B-arbre B, sauf si celle-ci est déjà présente. L'arbre B n'est pas vide, et sa racine n'est pas un nœud complet (pas un $2t$-nœud). Elle retourne un booléen indiquant si l'insertion a eu lieu.

```python
def __insert(B, x):
    i = search_pos(B.keys, x)

    if i < B.nbkeys and B.keys[i] == x:
        return False
    elif B.children == []:
        B.keys.insert(i, x)
        return True
    else:
        if B.children[i].nbkeys == 2 * B.degree - 1:
            if B.children[i].keys[B.degree-1] == x:
                return False
            split(B, i)
            if x > B.keys[i]:
                i += 1
        return __insert(B.children[i], x)
```

Solution 4 **(B-arbres et mystère –** ***2 points*****)**

```
nodes = [[22], [15], [27, 41], [8, 12], [18, 19, 20], [24, 25], [30, 35, 38], [45, 48]]

degree = 2
```