Structures de données avancées :

Arbres 'Left Leaning Red-Black'

Pr ZEGOUR DJAMEL EDDINE
Ecole Supérieure d'Informatique (ESI)
http://zegour.esi.dz
email: d_zegour@esi.dz

Left Leaning Red-Black Tree

Motivation : code actuel difficile à maintenir et à réutiliser (100 à 200 lignes de code)

Objectif : Réduire considérable du code

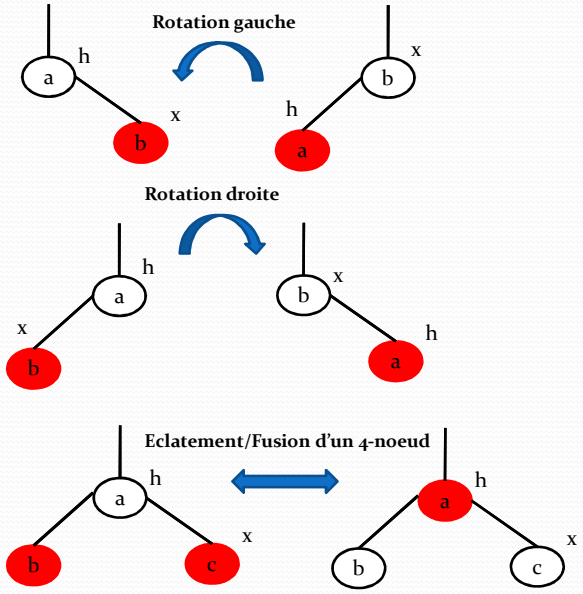
'Left Leaning' : éliminer les fils droits rouges

Solution : Insertion Top down (version récursive)

- **Ajouter 3 lignes de code à l'algorithme d'insertion dans un BST**
- **Ajouter 5 lignes de code pour supprimer le minimum (maximum)**
- **Ajouter 30 lignes de code pour l'opération de suppression**

Garder les opérations : Eclatement (Flipcolor), Rotation gauche (Rotateleft)et rotation droite(Rotateright)

Left Leaning Red-Black Tree



Rotation gauche

```
Node rotateLeft(Node h)
{
x = h.right;
h.right = x.left;
x.left = h;
x.color = h.color;
h.color = RED;
return x;
}
```

Rotation droite

```
Node rotateRight(Node h)
{
x = h.left;
h.left= x.right;
x.right= h;
x.color = h.color;
h.color = RED;
return x;
}
```

Eclatement/Fusion d'un 4-noeud

```
void flipColors(Node h)
{
h.color = !h.color;
h.left.color = !h.left.color;
h.right.color = !h.right.color;
}
```

Left Leaning Red-Black Tree

```
public class LLRB<Key extends Comparable<Key>,
Value>
{
private static final boolean RED = true;
private static final boolean BLACK = false;
private Node root;
private class Node
{
private Key key;
private Value val;
private Node left, right;
private boolean color;
Node(Key key, Value val)
{
this.key = key;
this.val = val;
this.color = RED;
}
}
```

Boolean représente un champ à un bit

Les nouveaux noeuds sont toujours Rouge

Code Java pour implémenter les arbres LLRB (Arbre de recherche binaire sans le code en bleu)

Left Leaning Red-Black Tree

```
public Value search(Key key)
{
Node x = root;
while (x != null)
{
int cmp = key.compareTo(x.key);
if (cmp == 0) return x.val;
else if (cmp < 0) x = x.left;
else if (cmp > 0) x = x.right;
}
return null;
}
```

Left Leaning Red-Black Tree

```
public void insert(Key key, Value value)
{
root = insert(root, key, value);
root.color = BLACK;
}
private Node insert(Node h, Key key, Value value)
{
if (h == null) return new Node(key, value);
if (isRed(h.left) && isRed(h.right)) colorFlip(h);
int cmp = key.compareTo(h.key);
if (cmp == 0) h.val = value;
else if (cmp < 0) h.left = insert(h.left, key, value);
else h.right = insert(h.right, key, value);
if (isRed(h.right) && !isRed(h.left)) h = rotateLeft(h);
if (isRed(h.left) && isRed(h.left.left)) h = rotateRight(h);

return h;
}
}
```

Déplacer cette ligne à la fin pour avoir un RB 2-3 tree

Code Java pour implémenter les arbres LLRB (Arbre de recherche binaire sans le code en bleu)

Left Leaning Red-Black Tree

Généralisation : famille des arbres Red Black

Algorithme d'insertion TOP DOWN (4 modes)

- **RB 2-4 tree**
- **RB 2-3 tree**
- **LLRB 2-4 tree**
- **LLRB 2-3 tree**

Programmation Pascal

Utilisation des opérations (Flipcolor, Rotateleft et Rotateright)

Mode_arbre désigne les 4 modes
Si Mode_arbre est différent des 4 modes, il s'agit d'un arbre de recherche binaire

Left Leaning Red-Black Tree

```
procedure insert ( val : typequelconque; var A : Ptr_noeud);
  begin
   if A = nil
   then
     begin
       New(A); aff_info(A, val);
       aff_fg(A, nil);    Aff_fd(A, nil);    Aff_couleur(A, Rouge)
    end
   else
    begin
     // eclatememement pendant la descente
     if (mode_arbre = Mode_RB24) or  (mode_arbre = Mode_LLRB24)
     then
      if (couleur(fg(A))= rouge) and (couleur(fd(A))= rouge)
      then  Flipcolors(A);

     if val= info(A)
     then aff_info(A, Val)
     else
      if Val < info(A)
      then insert( val, A.fg)
      else insert( val, A.fd)   ;
```

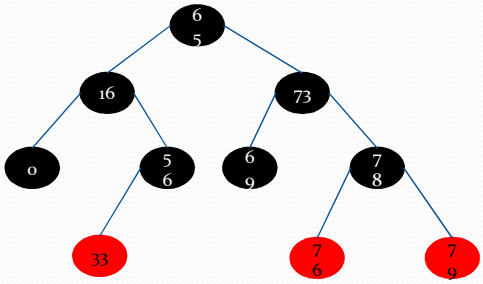
Left Leaning Red-Black Tree

```
// Restructuration pendant la remontée
if (mode_arbre = Mode_LLRB24) Or (mode_arbre = Mode_LLRB23)
Then
     if (couleur(fd(A))= rouge) and (couleur(fg(A))= noir)  then  A := rotateLeft(A);
if (mode_arbre = Mode_RB24) or (mode_arbre = Mode_RB23)
 then
     if (couleur(fg(A))= rouge) and (couleur(fd(fg(A)))= rouge) then  A.Fg :=
rotateleft(fg(A));

if (couleur(fg(A))= rouge) and (couleur(fg(fg(A)))= rouge)    then  A := rotateRight(A);

if (mode_arbre = Mode_RB24)or (mode_arbre = Mode_RB23)
Then
   if (couleur(fd(A))= rouge) and (couleur(fg(fd(A)))= rouge) then  A.fd :=
Rotateright(fd(A));
if (mode_arbre = Mode_RB24) or (mode_arbre = Mode_RB23)
Then
    if (couleur(fd(A))= rouge) and (couleur(fd(fd(A)))= rouge) then  A := rotateleft(A);

if (mode_arbre = Mode_LLRB23) or  (mode_arbre = Mode_RB23)
Then
     if (couleur(fg(A))= rouge) and (couleur(fd(A))= rouge)  then  Flipcolors(A);
    end
```

Left Leaning Red-Black Tree

Arbre RB 2-4

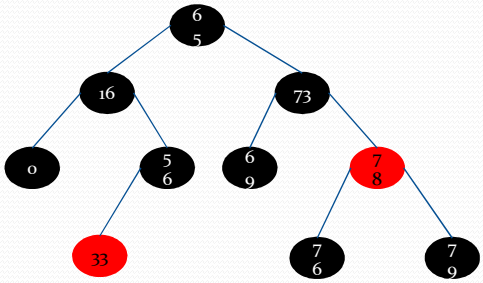
Propriétés de l'arbre RB
Noir avec un fils gauche rouge =1
Noir avec un fils droit rouge =0
Noir avec deux fils rouges =1
Noir feuille =2



Left Leaning Red-Black Tree

Arbre RB 2-3

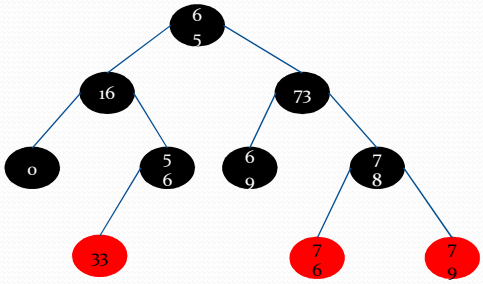
Propriétés de l'arbre RB
Noir avec un fils gauche rouge =1
Noir avec un fils droit rouge =1
Noir avec deux fils rouges =0
Noir feuille =4



Left Leaning Red-Black Tree

Arbre LLRB 2-4

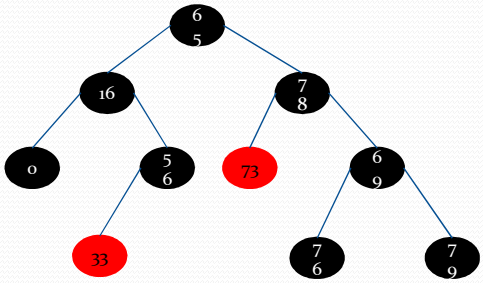
Propriétés de l'arbre RB
Noir avec un fils gauche rouge =1
Noir avec un fils droit rouge =0
Noir avec deux fils rouges =1
Noir feuille =2



Left Leaning Red-Black Tree

Arbre LLRB 2-3

Propriétés de l'arbre RB
Noir avec un fils gauche rouge =2
Noir avec un fils droit rouge =0
Noir avec deux fils rouges =0
Noir feuille =4



Left Leaning Red-Black Tree

Suppression pour les arbres LLRB 2-3

Basé sur l'approche inverse de l'algorithme d'insertion Top down d'un arbre 2-3-4

Pendant la recherche: entreprendre des rotations et les fusions (Flipcolors) des 4-noeuds pour s'assurer que la recherche ne s'arrete pas sur un 2-noeuds.

De cette manière, il suffit de supprimer l'élément au niveau de la feuille.

Utiliser ensuite la méthode *Fixup*
Pour vérifier les propriétés LLRB

Fixup

```
if (isRed(h.right) && !isRed(h.left))
    h = rotateLeft(h);
if (isRed(h.left) && isRed(h.left.left))
    h = rotateRight(h);
if (isRed(h.left) && isRed(h.right))
    colorFlip(h);
```

Suppression pour les arbres LLRB 2-4

Exige une rotation supplémentaire (voir technique ascendante pour les arbres 2-4)