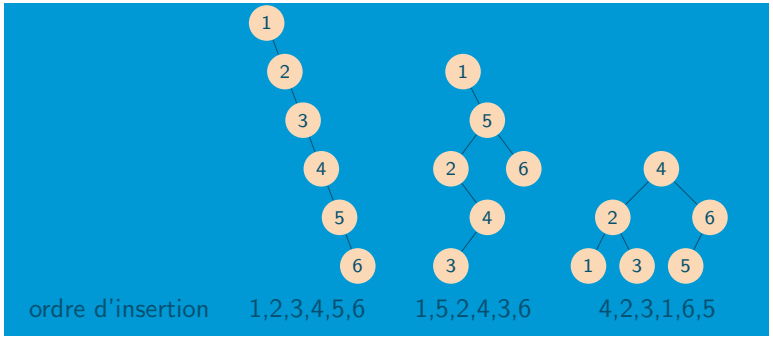
Construction d'un arbre binaire

même algorithme que la recherche + création d'une nouvelle feuille



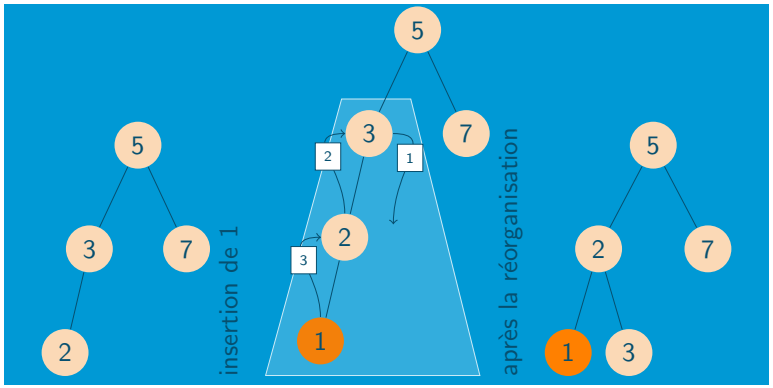
ne conserve pas une hauteur minimale !

Maintien de la hauteur minimale

- nécessite de conserver une différence de hauteur de au plus un entre les deux sous-arbres gauche et droit
- ceci ne peut être fait avec la construction classique d'un ABR : on n'a pas le choix du sous-arbre dans lequel on insère
- on pourrait réorganiser les valeurs avant de construire l'arbre, mais cela revient à trier, l'utilisation d'un ABR par la suite devient inutile

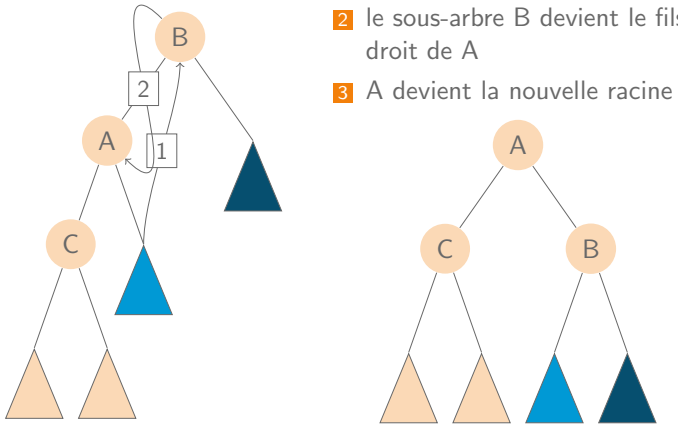
inventer une nouvelle manière d'insérer les valeurs

Exemple de réorganisation



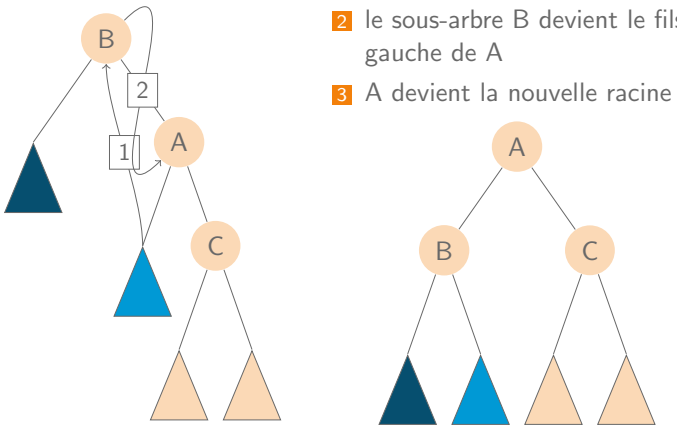
La rotation droite

1. le fils droit de A remplace le fils gauche de B
2. le sous-arbre B devient le fils droit de A
3. A devient la nouvelle racine



La rotation gauche

1. le fils gauche de A remplace le fils droit de B
2. le sous-arbre B devient le fils gauche de A
3. A devient la nouvelle racine



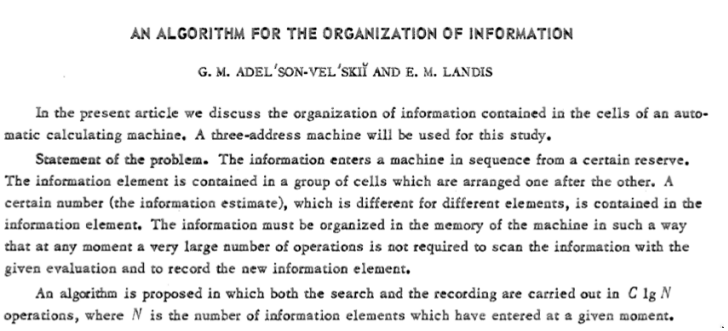
Peut-on toujours rééquilibrer un arbre ?

Peut-on le faire en temps constant ? en temps linéaire ? en temps logarithmique ?

Peut-on construire un ABR qui maintienne l'équilibre sans que cela coûte ?

Les AVL

AVL = G.M. **A**del'son-**V**elski et E.M. **L**andis,
"An algorithm for the organization of information".
Proceedings of the USSR Academy of Sciences (in Russian). 146 : 263–266. English translation by Myron J. Ricci in Soviet Math. Doklady, 3 :1259–1263, 1962.



AN ALGORITHM FOR THE ORGANIZATION OF INFORMATION

G. M. ADEL'SON-VEL'SKIĬ AND E. M. LANDIS

In the present article we discuss the organization of information contained in the cells of an automatic calculating machine. A three-address machine will be used for this study.

Statement of the problem. The information enters a machine in sequence from a certain reserve. The information element is contained in a group of cells which are arranged one after the other. A certain number (the information estimate), which is different for different elements, is contained in the information element. The information must be organized in the memory of the machine in such a way that at any moment a very large number of operations is not required to scan the information with the given evaluation and to record the new information element.

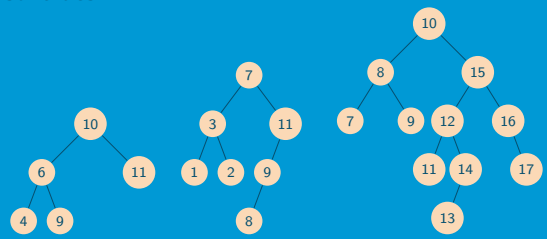
An algorithm is proposed in which both the search and the recording are carried out in $C \lg N$ operations, where N is the number of information elements which have entered at a given moment.

Les AVL

Définition : un AVL est un ABR, éventuellement vide, tel que :

- la différence de hauteur entre les sous-arbres gauche et droit d'un nœud est au plus un
- les sous-arbres gauche et droit sont des AVL

Lesquels sont des AVL ?



Propriétés dans les AVL

- si n est la taille de l'arbre, alors la hauteur est $\lfloor \log_2(n) \rfloor$
- l'insertion d'une valeur dans un AVL aboutit à l'ajout d'une feuille et donc à un déséquilibre de hauteur d'au plus 2
- la suppression d'une valeur dans un AVL aboutit à la suppression d'une feuille ou d'un nœud et donc à un déséquilibre de hauteur d'au plus 2

On définit pour chaque nœud une valeur de déséquilibre :

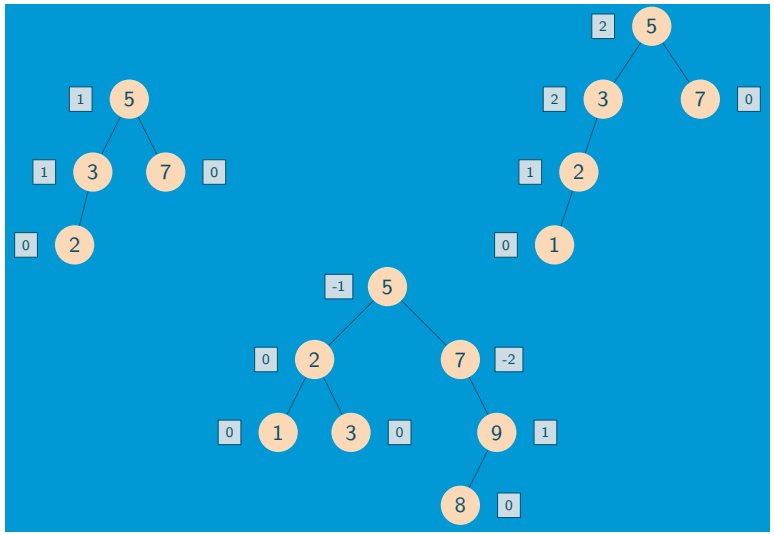
$$d(n) = h(\text{fils gauche}(n)) - h(\text{fils droit}(n))$$

qui vaut zéro pour une feuille.

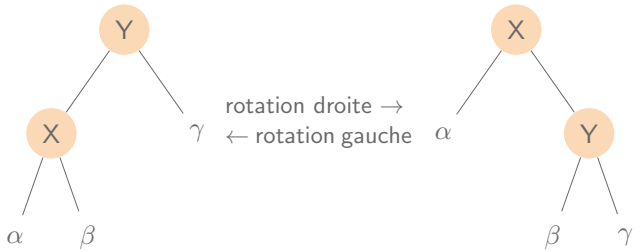
Dans un AVL, le déséquilibre de chaque nœud vaut -1, 0 ou $+1$.

- déséquilibre positif : h de gauche $>$ h de droite
- déséquilibre négatif : h de gauche $<$ h de droite

Exemples



Propriétés des rotations



- Après une rotation droite autour du sommet Y, on a :
  - $d'(X) = d(X) - 1 + \min(d'(Y), 0)$
  - $d'(Y) = d(Y) - 1 - \max(d(X), 0)$
- Après une rotation gauche autour du sommet X, on a :
  - $d'(X) = d(X) + 1 - \min(d(Y), 0)$
  - $d'(Y) = d(Y) + 1 + \max(d'(X), 0)$

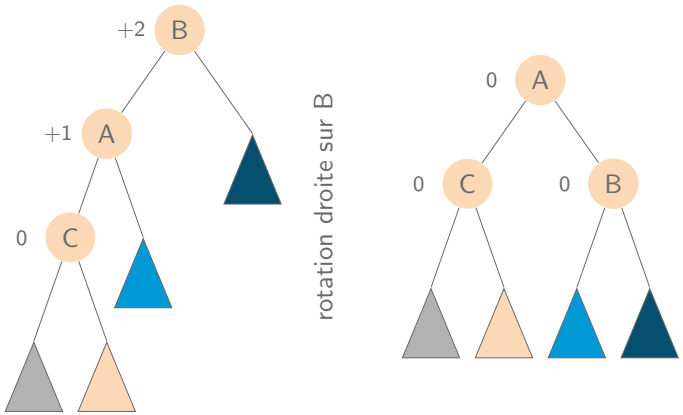
Insertion dans un AVL, cas simples

- si il n'y a pas de déséquilibre, alors la hauteur du sous-arbre gauche ou droit est augmentée au plus de 1
- si le déséquilibre de la racine est $+1$ et que l'ajout se fait dans le sous-arbre droit, alors la hauteur du sous-arbre droit après insertion est augmentée d'au plus 1
- si le déséquilibre de la racine est -1 et que l'ajout se fait dans le sous-arbre gauche, alors la hauteur du sous-arbre gauche après insertion est augmentée d'au plus 1

Dans tous ces cas, l'arbre reste équilibré

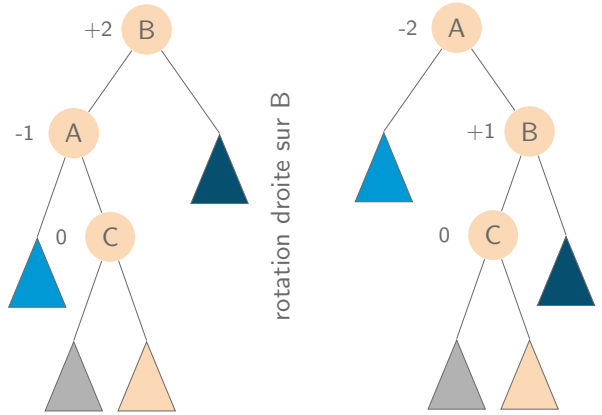
Insertion dans un AVL, cas 1a

Déséquilibre de +1 avant l'ajout dans le fils gauche du fils gauche
déséquilibre résultant de +2



Insertion dans un AVL, cas 1b

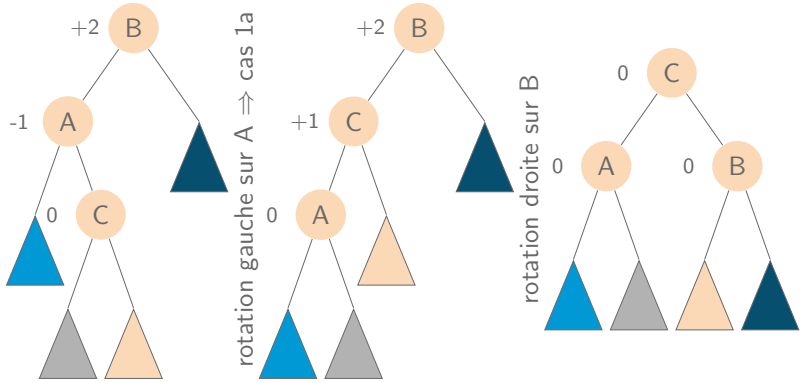
Déséquilibre de +1 avant l'ajout dans le droit du fils gauche
déséquilibre résultant de +2



Ne fonctionne pas !

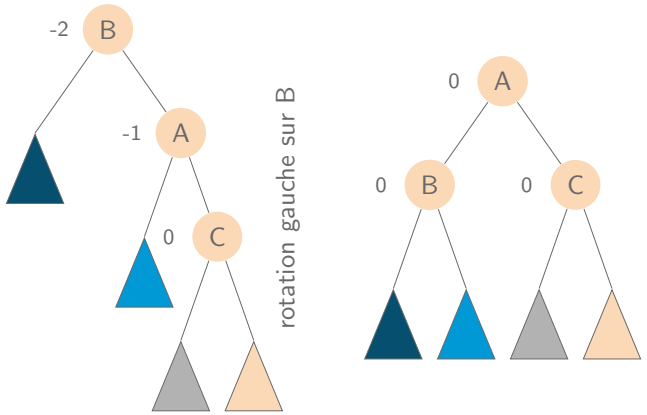
Insertion dans un AVL, cas 1b

Déséquilibre de +1 avant l'ajout dans le fils droit du fils gauche
déséquilibre résultant de +2



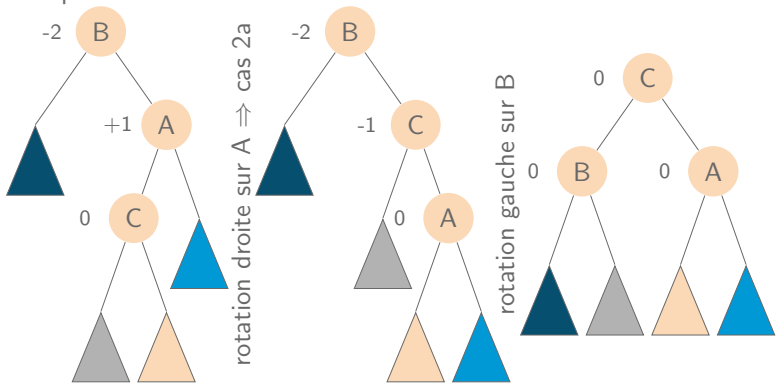
Insertion dans un AVL, cas 2a

Déséquilibre de -1 avant l'ajout dans le fils droit du fils droit
déséquilibre résultant de -2



Insertion dans un AVL, cas 2b

Déséquilibre de -1 avant l'ajout dans le fils gauche du fils droit
déséquilibre résultant de -2



Rotations

```
procedure ROTATIONGAUCHE(b)
    a ← fils droit de b
    (* le fils gauche de A remplace le fils droit de B *)
    fils droit de b ← fils gauche de a
    (* le sous-arbre B devient le fils gauche de A *)
    fils gauche de a ← b
    (* A devient la nouvelle racine *)
    b ← a
    (* mise à jour de la nouvelle racine *)
    hauteur du fils gauche de b ← max(h(b.filsg.filsg),h(b.filsg.filsd)) + 1
    hauteur de b ← max (h(b.filsg),h(b.filsd)) + 1
end procedure
```


Insertion

- insérer comme dans les ABR
- mettre à jour la hauteur de chaque nœud visité
- ré-équilibrer chaque nœud visité

```
Require: a est un AVL et e l'élément à insérer
Ensure: a est un AVL contenant e
  procedure INSERER(a, e)
      if a est vide then
          return nouvel arbre
      else
          if e < valeur de la racine de a then
              INSERER (fils gauche de a, e)
          else
              INSERER (fils droit de a, e)
          end if
          (* mettre à jour la hauteur de a puis ré-équilibrer *)
          hauteur de a ← max h(fils gauche de a),h(fils droite de b)
          EQUILIBRER (a)
      end if
  end procedure
```


Ré-équilibrage

```
procedure EQUILIBRER(a)
    if  h(fils gauche de a) - h(fils droit de a) = 2 then
        if h(gauche de gauche de a) < h(droit de gauche de a) then
            (* cas 1b *)
            ROTATIONGAUCHE (fils gauche de a)
        end if
        ROTATIONDROITE (a) (* cas 1a *)
    else if  h(fils gauche de a) - h(fils droit de a) = -2 then
        if h(droit de droit de a) < h(gauche de droit de a) then
            (* cas 2b *)
            ROTATIONDROITE (fils droit de a)
        end if
        ROTATIONGAUCHE (a) (* cas 2a *)
    end if
end procedure
```

Complexité de l'insertion

- complexité d'une rotation : $\Theta(1)$
- complexité d'un équilibrage d'un nœud : $\Theta(1)$ (au maximum 2 rotations)
- complexité de l'insertion : $\mathcal{O}(\log n)$ (au maximum un équilibrage par nœud traversé)