(Séquence 7.5

Suppression d'un élément

Suppression d'un élément

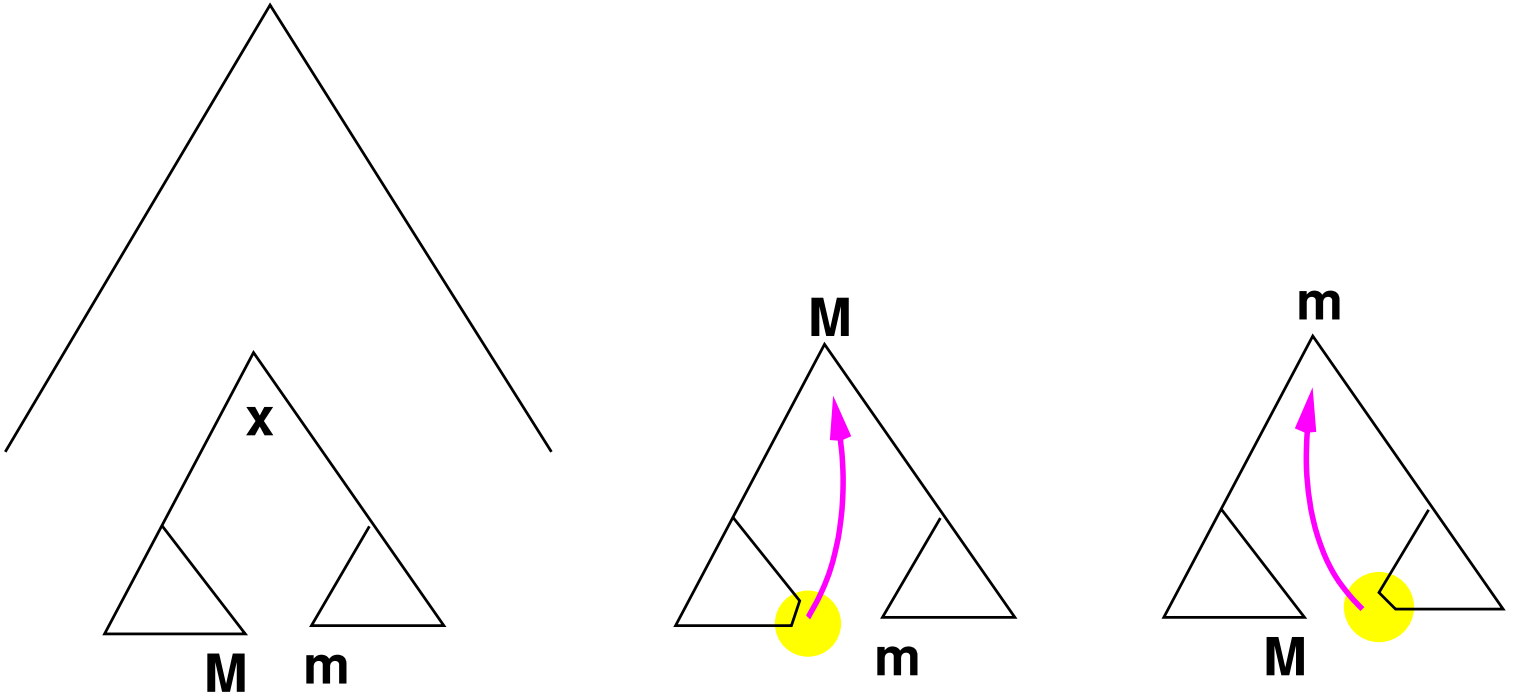
Principe :

- rechercher le sous-arbre dont l'élément à supprimer est la racine
- effectuer la suppression
  - la suppression est simple si l'élément est une feuille
  - la suppression est plus complexe sinon : modifier le sous-arbre en remplaçant la racine par son « précédent » dans le sous-arbre. *« précédent »* $\equiv$ *le plus grand des éléments inférieurs à la racine*
    - les éléments inférieurs forment le sous-arbre gauche
    - le plus grand des inférieurs est au bout de la branche droite

Remarque : autre façon pour supprimer la racine d'un ABR : la remplacer par son « suivant » (le plus petit des éléments supérieurs à la racine)

Suppression d'un élément

Pour supprimer l'élément *x*



Fonction de suppression

```
;;; abr-moins: Nombre * ArbreBinRecherche
;;;     -> ArbreBinRecherche
;;; (abr-moins x ABR) rend ABR lorque x n'y est pas
;;; et sinon rend un arbre bin. de rech. qui contient
;;; toutes les étiquettes de ABR sauf x
(define (abr-moins x ABR)
  (if (ab-noeud? ABR)
      (let ((e (ab-etiquette ABR)))
        (cond
          ((= x e) (moins-racine ABR)) ; supprimer racin
          ((< x e) (ab-noeud
                    e
                    (abr-moins x (ab-gauche ABR))
                    (ab-droit ABR)))
          (else (ab-noeud
                 e
                 (ab-gauche ABR)
                 (abr-moins x (ab-droit ABR))))))
      (ab-vide)))
```

Fonction de suppression de la racine

```
;;; moins-racine: ArbreBinRecherche -> ArbreBinRecherche
;;; (moins-racine ABR) rend l'arbre bin. de rech. qui
;;; contient toutes les étiquettes qui apparaissent
;;; dans ABR hormis l'étiquette de sa racine.
;;; HYPOTHÈSE: ABR non vide
(define (moins-racine ABR)
  (cond ((not (ab-noeud? (ab-gauche ABR)))
         (ab-droit ABR))
        ((not (ab-noeud? (ab-droit ABR)))
         (ab-gauche ABR))
        (else
          défaire le ss-ab-g -> max + ABR privé du max )))
```

Le max d'un arbre binaire de recherche

Cette fonction renvoie le max d'un arbre binaire de recherche

```
(define (max-abr ABR)
  (if (ab-noeud? (ab-droit ABR))
      (max-abr (ab-droit ABR))
      (ab-etiquette ABR)))
```

Arbre binaire de recherche sans son max

Cette fonction renvoie un ABR

```
(define (abr-sauf-max ABR)
  (if (ab-noeud? (ab-droit ABR))
      (ab-noeud (ab-etiquette ABR)
                (ab-gauche ABR)
                (abr-sauf-max (ab-droit ABR)))
      (ab-gauche ABR)))
```

Le max ET de l'arbre privé du max

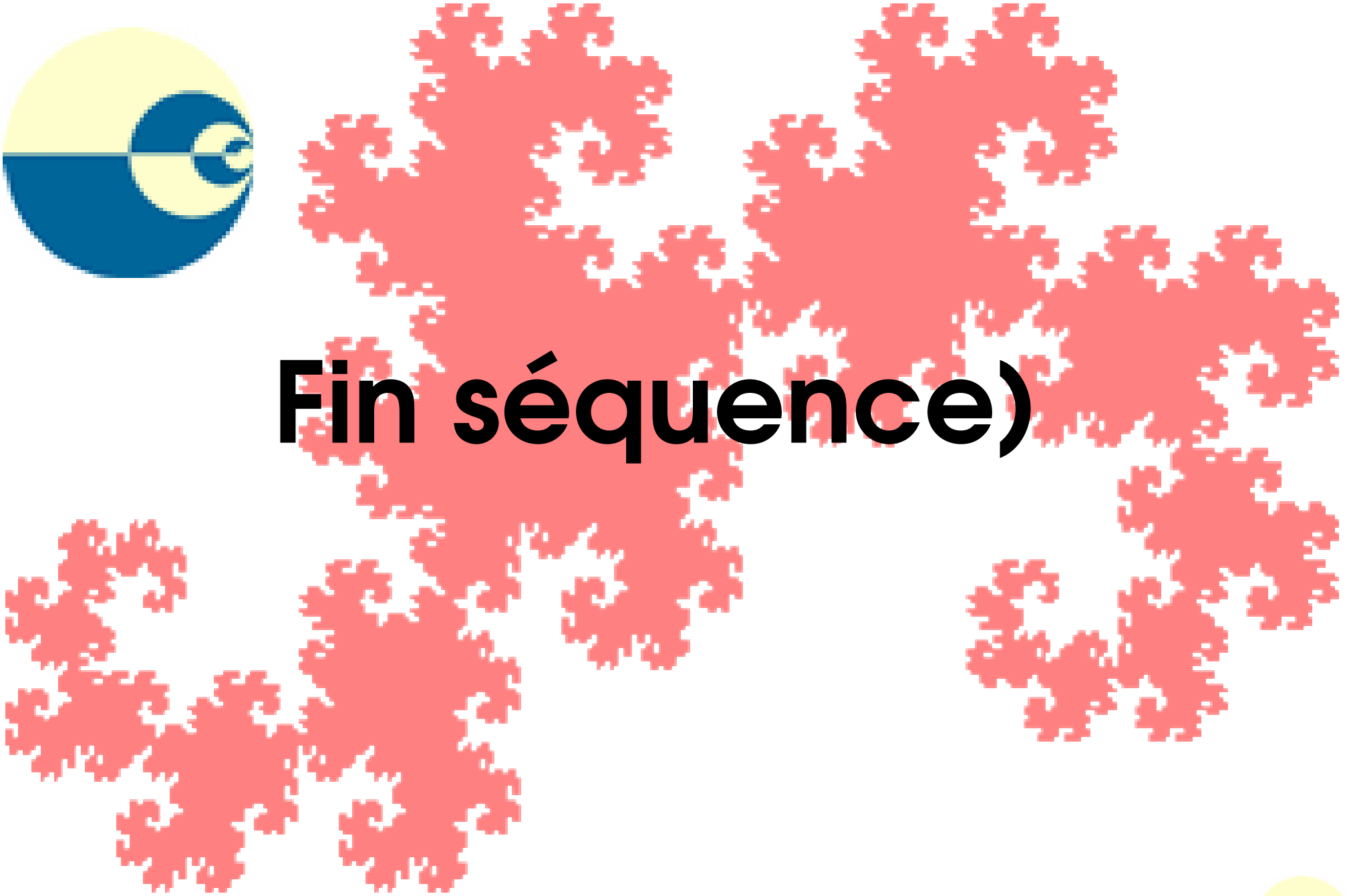
```
(define (max-sauf-max ABR)
  (if (ab-noeud? (ab-droit ABR))
      (let ((m-sm-ss-ab-d (max-sauf-max (ab-droit ABR))))
        (list (car m-sm-ss-ab-d)
              (ab-noeud (ab-etiquette ABR)
                        (ab-gauche ABR)
                        (cadr m-sm-ss-ab-d))))
      (list (ab-etiquette ABR)
            (ab-gauche ABR) ) ) )
```

Fonction de suppression de la racine

```
(define (moins-racine ABR)
  (cond ((not (ab-noeud? (ab-gauche ABR)))
         (ab-droit ABR))
        ((not (ab-noeud? (ab-droit ABR)))
         (ab-gauche ABR))
        (else ; defaire le ss-ab-g -->  max + ABR prive
         (let ((m-sm-ss-ab-g (max-sauf-max (ab-gauche AB
           (ab-noeud (car m-sm-ss-ab-g)
                     (cadr m-sm-ss-ab-g)
                     (ab-droit ABR))))))))
```

Perspectives

- D'autres implantations avec des propriétés plus fortes comme les AVL (Georgii Adelson-Velsky et Evguenii Landis, 1962) : arbres automatiquement quasi-équilibrés (la différence des profondeurs entre fils gauche et droit est au plus de 1).
- Introduction d'aléa pour prévenir (statistiquement) la fabrication d'arbres dégénérés.



Fin séquence)