(Séquence 7.4

Ajout d'un élément

Ajout d'un élément

Principe : ajouter « à l'endroit où » la recherche s'est terminée en échec
Remarque : la fonction d'ajout rend comme résultat un arbre binaire de recherche qui est (re)construit au fur et à mesure (constructeur `ab-noeud`)

Fonction d'ajout

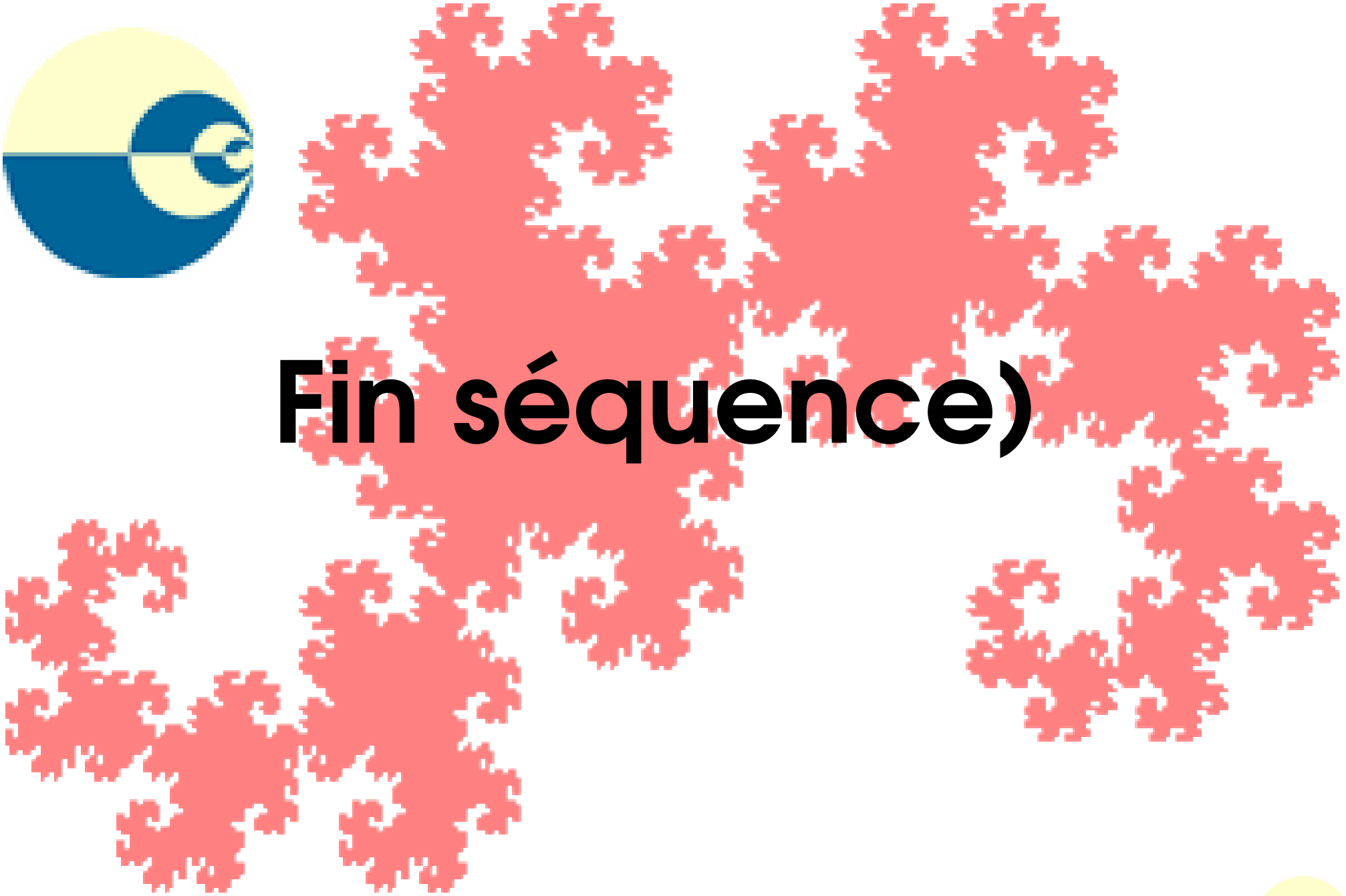
```
;;; abr-ajout: Nombre * ArbreBinRecherche
;;;     -> ArbreBinRecherche
;;; (abr-ajout x ABR) rend l'arbre ABR lorque x est
;;; dans ABR et sinon rend un arbre bin. de rech.
;;; qui contient x et toutes les étiquettes d'ABR
(define (abr-ajout x ABR)
  (if (ab-noeud? ABR)
      (let ((e (ab-etiquette  ABR)))
        (cond
          ((= x e) ABR)
          ((< x e)
           (ab-noeud e (abr-ajout x (ab-gauche ABR))
                       (ab-droit ABR)))
          (else (ab-noeud
                 e
                 (ab-gauche ABR)
                 (abr-ajout x (ab-droit ABR))))))
      (ab-noeud x (ab-vide) (ab-vide))))
```

Suppression d'un élément

C'est la fonction la plus compliquée. Voir le code commenté.

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)