(Séquence 7.2

Récursion sur ABR

Principe des algorithmes

Une *unique comparaison* avec l'étiquette de l'arbre permet d'aiguiller le traitement

- *soit* dans le sous-arbre gauche
- *soit* dans le sous-arbre droit

Le principe d'aiguillage permet de « laisser tomber » un sous-arbre entier (gauche ou droit) et de recommencer récursivement le traitement sur *un seul* sous-arbre.

CC-BY-NC-SA by C.Queinnec

Schéma de fonction

```
;;; f-abr: Nombre * ArbreBinRecherche -> ...
(define (f-abr x ABR)
  (if (ab-noeud? ABR)
      (let ((e (ab-etiquette ABR)))
        (cond
          ((= x e) ...)
          ((< x e) (combinaison
                     (f-abr x (ab-gauche ABR))))
          (else (combinaison
                  (f-abr x (ab-droit ABR)))) ) )
      cas-arbre-vide ))
```

Efficacité

Si chaque sous-arbre contient à peu près la moitié des nœuds de l'arbre entier (arbre équilibré), alors le nombre de noeuds restant à examiner est *divisé par deux* à chaque fois : **dichotomie**

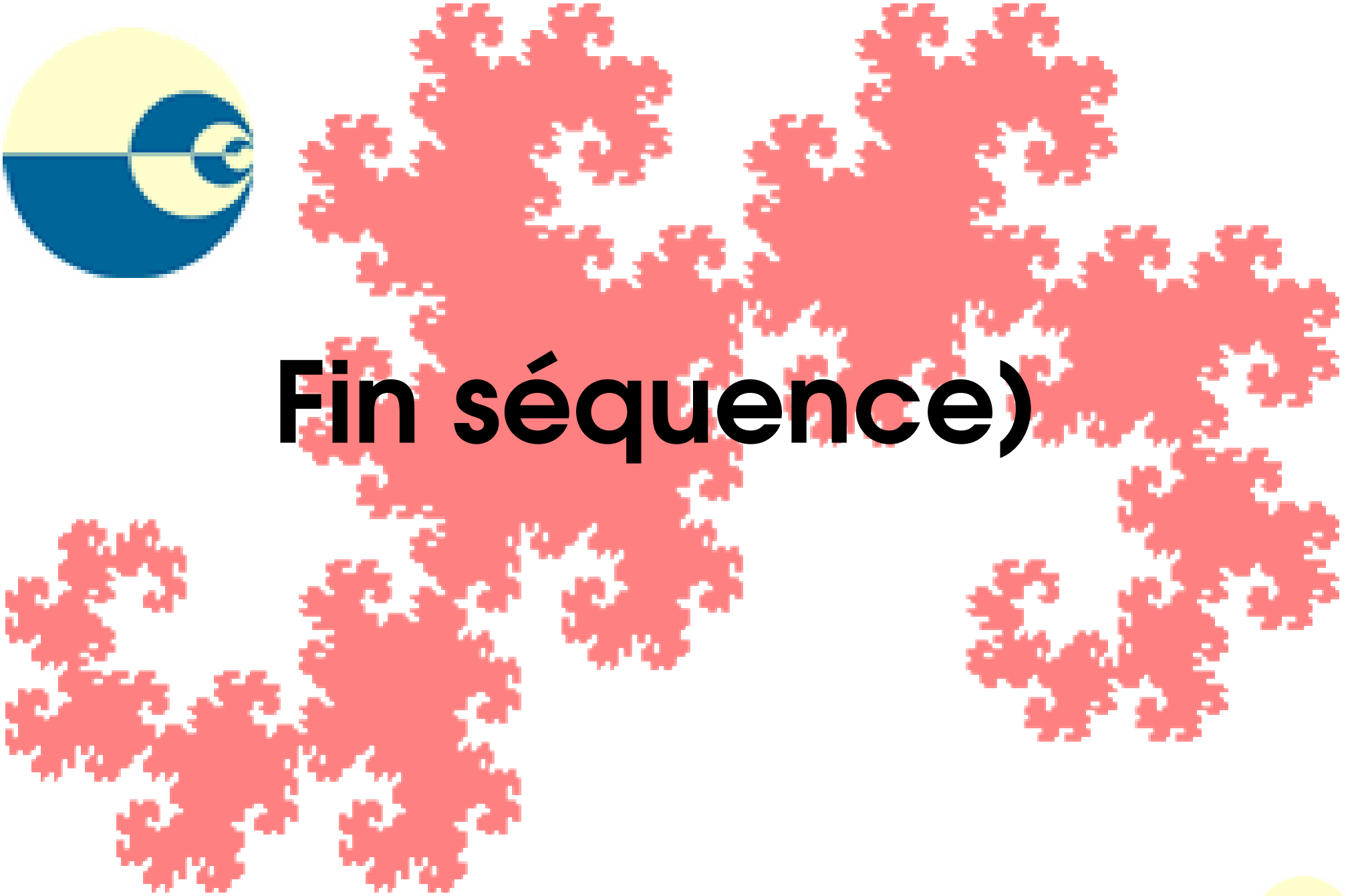
$$n \to n/2 \to n/4 \to \ldots \to n/2^p \Longrightarrow p \approx \log_2 n$$

Dichotomie $\to$ temps d'exécution logarithmique $0(\log n)$

MAIS si l'arbre est dégénéré le nombre de noeuds restant à examiner est seulement *diminué de 1* à chaque fois

$$n \to n-1 \to n-2 \to n-3 \ldots$$

$\to$ temps d'exécution linéaire $0(n)$



Fin séquence)

CC-BY-NC-SA by C.Queinnec