(Séquence 8.3

Exemples

Nombre de nœuds

```
;;; nombre-noeuds-arbre: ArbreGeneral[α] -> nat
;;; (nombre-noeuds-arbre G) rend le nombre de
;;; noeuds de G
(define (nombre-noeuds-arbre G)
  ;; nombre-noeuds-foret: Foret[α] -> nat
  ;; (nombre-noeuds-foret F) rend le nombre de
  ;; noeuds de F
  (define (nombre-noeuds-foret F)
    (if (pair? F)
        (+ (nombre-noeuds-arbre (car F))
           (nombre-noeuds-foret (cdr F)))
        0 ) )
  (+ 1 (nombre-noeuds-foret (ag-foret G))) )
```

Avec un `map`

```
;;; nombre-noeuds-arbre: ArbreGeneral[α] -> nat
;;; (nombre-noeuds-arbre G) rend le nombre de
;;; noeuds de G
(define (nombre-noeuds-arbre G)
  ;; nombre-noeuds-foret: Foret[α] -> nat
  ;; (nombre-noeuds-foret F) rend le nombre de
  ;; noeuds de F
  (define (nombre-noeuds-foret F)
    (reduce + 0 (map nombre-noeuds-arbre F)))
  (+ 1 (nombre-noeuds-foret (ag-foret G))))
```

ou encore:

```
(define (nombre-noeuds-arbre G)
  (reduce + 1 (map nombre-noeuds-arbre (ag-foret G))))
```

Profondeur

```
;;; ag-profondeur: ArbreGeneral[α] -> nat
;;; (ag-profondeur G) rend la profondeur de
;;; l'arbre G
(define (ag-profondeur G)
  ;; profondeurForet: Foret[α] -> nat
  ;; (profondeurForet F) rend la profondeur de F
  (define (profondeurForet F)
    (if (pair? F)
        (max (ag-profondeur (car F))
             (profondeurForet (cdr F)))
        0))
  (+ 1 (profondeurForet (ag-foret G))))
```

ou avec des itérateurs:

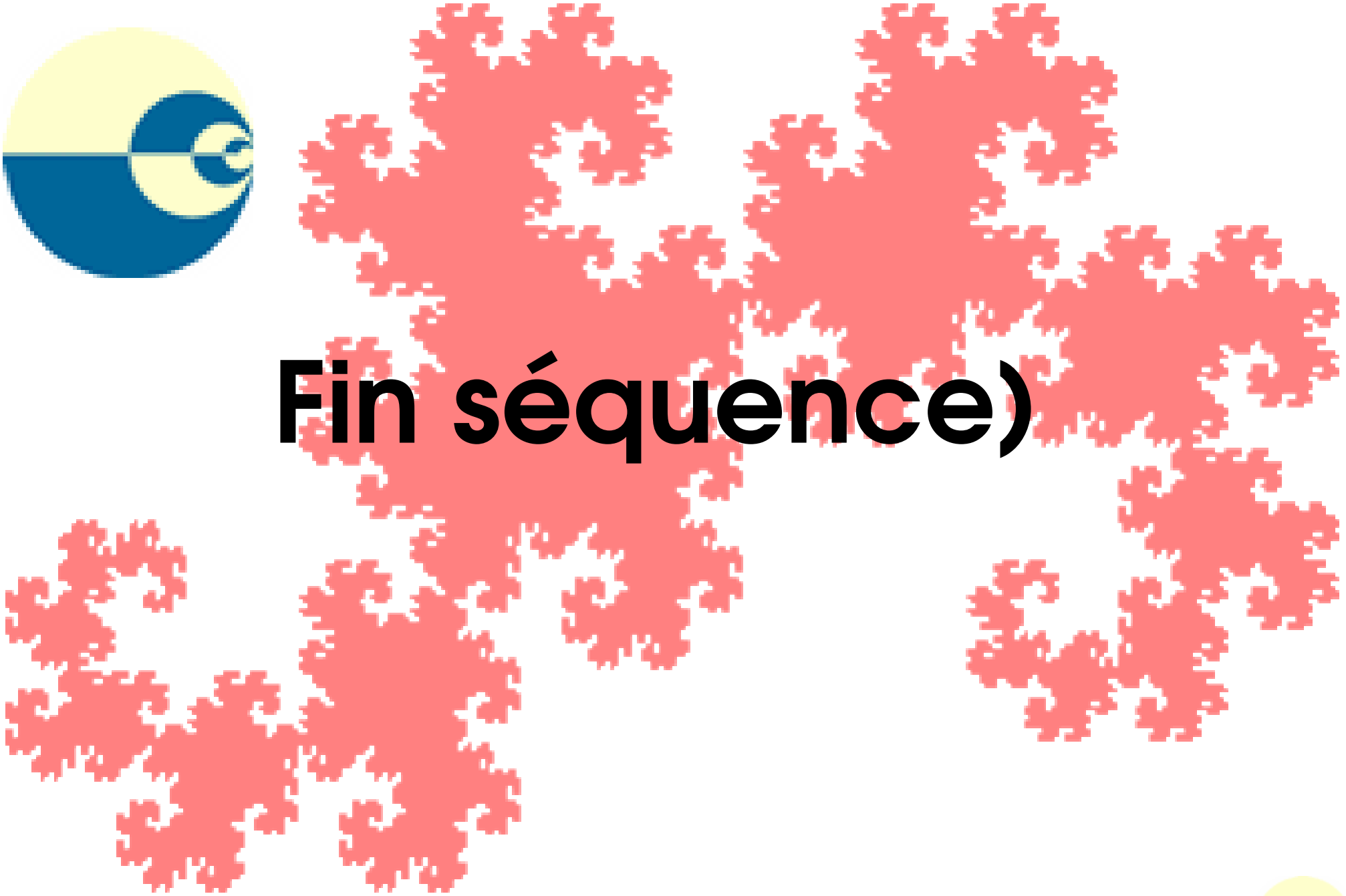
```
(define (ag-profondeur G)
  (+ 1 (reduce max 0 (map ag-profondeur (ag-foret G)))))
```

Liste préfixe

```
;;; ag-liste-prefixe: ArbreGeneral[α] -> LISTE[α]
;;; (ag-liste-prefixe G) rend la liste préfixe
;;; des étiquettes de l'arbre G
(define (ag-liste-prefixe G)
  ;; liste-prefixe-foret: Foret[α] -> LISTE[α]
  ;; rend la concaténation des listes préfixes
  ;; des étiquettes des arbres de F
  (define (liste-prefixe-foret F)
    (if (pair? F)
        (append (ag-liste-prefixe (car F))
                (liste-prefixe-foret (cdr F)))
        '() ) )
  (cons (ag-etiquette G)
        (liste-prefixe-foret (ag-foret G)) ) )
```

Avec des itérateurs

```
(define (ag-liste-prefixe G)
  (cons (ag-etiquette G)
        (reduce append
                '()
                (map ag-liste-prefixe (ag-foret G)))))
```



Fin séquence)