

Listes

travailler localement

Persistence : copies inutiles ?

→ accroche de la structure à la position de travail

Point de vue local → zipper (sur liste)

```
type 'a zipper = {  
  left : 'a list ;  
  pos : int ;  
  right : 'a list ; }
```

Fonctions :

```
go_left : 'a zipper -> 'a zipper  
go_right : 'a zipper -> 'a zipper  
value : 'a zipper -> 'a pos : 'a zipper -> int
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 1

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 2

Programmation fonctionnelle 5–6

Xavier Urbain

ensie

2011/2012

Listes

travailler localement

Persistence : copies inutiles ?

→ accroche de la structure à la position de travail

Point de vue local → zipper (sur liste)

```
type 'a zipper = Zip of ('a list * int * 'a list)
```

Organisation → constructeur futé

Fonctions :

```
go_left : 'a zipper -> 'a zipper  
go_right : 'a zipper -> 'a zipper  
value : 'a zipper -> 'a pos : 'a zipper -> int
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 3

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 4

Ensembles

ABR

Structure dynamique encore

Aspects recherche et opérations ensemblistes

Organisation : relation d'ordre total

Savoir où chercher

Fonctions :

- Comparaison compare : 'a -> 'a -> int
- Ajout
- Appartenance
- #, U, ∩

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 3

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 4

Ensembles

Comparison x, y :

- Positive si x strictement plus grand que y
- Nulle x est égal à y
- Négative si x strictement plus petit que y

Organisation :

- D'un côté : les plus petits
- De l'autre : les autres

→ Arbres binaires

{21; 3; 42; 666; 73; 12; 37} ?

XU - ensieie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 5

XU - ensieie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 6

ABR

Ensembles

Arbres de 'a :

compare : 'a → 'a → int

type 'a abr = Empty | Node **of** (abr * 'a * abr)

Opérations de recherche → bornées par **profondeur** mieux que listes
Opérations de construction → fonction de la **profondeur** moins bien
Coûts ? Réduire = réduire **profondeur**

ABR

Ensembles

Réduire profondeur → **équilibre**

Invariant supplémentaire : prof. à droite = prof. à gauche ±1

Information supplémentaire : **profondeur**

AVL de 'a :

compare : 'a → 'a → int

type 'a avl = Empty | Node **of** ('a avl * 'a * 'a avl * **int**)

height : 'a avl → int

[Adelson-Velsky, Landis 62]

AVL

Ensembles

Réduire profondeur → **équilibre**

Invariant supplémentaire : prof. à droite = prof. à gauche ±1

Information supplémentaire : **profondeur**

Constructeur **futé** pour maintenir invariant

node : 'a avl → 'a → 'a avl → 'a avl

AVL

XU - ensieie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 7

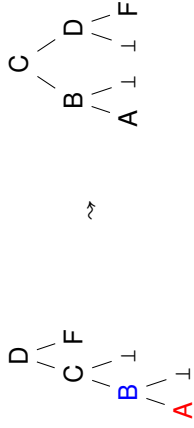
XU - ensieie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 8

Ensembles

- Minimum ?
- Recherche ?
→ comme ABR
- Ajout ?

Invariant → savoir équilibrer



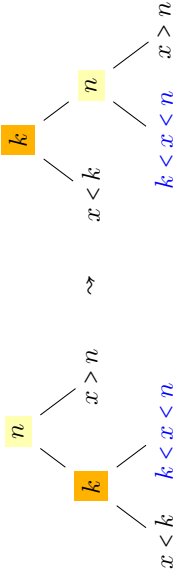
AVL

bon marché !

(déséq. de 1)

Ensembles

Rotation droite :

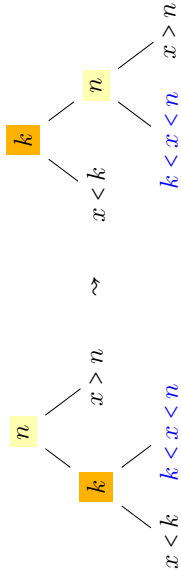


↔ Rotation gauche

AVL

Ensembles

Rotation droite :

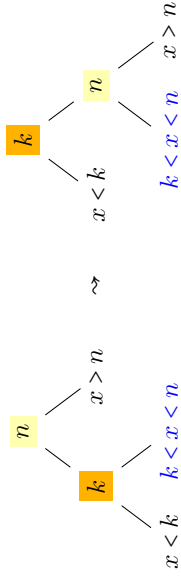


↔ Rotation gauche

AVL

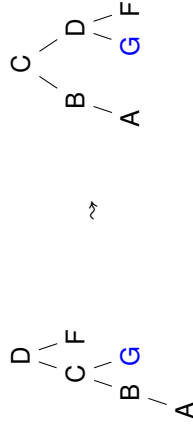
Ensembles

Rotation droite :



↔ Rotation gauche

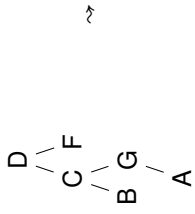
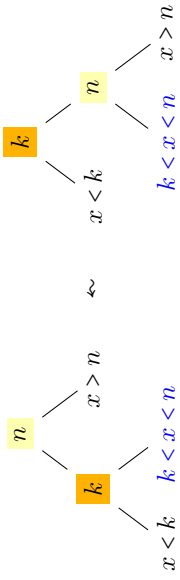
AVL



Échec

Ensembles

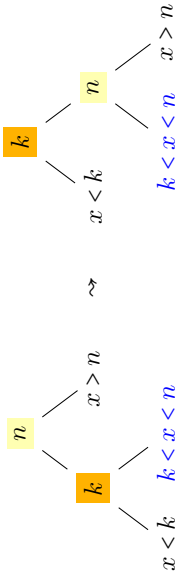
Rotation droite :



Échec

Ensembles

Rotation droite :



Rotation gauche-droite

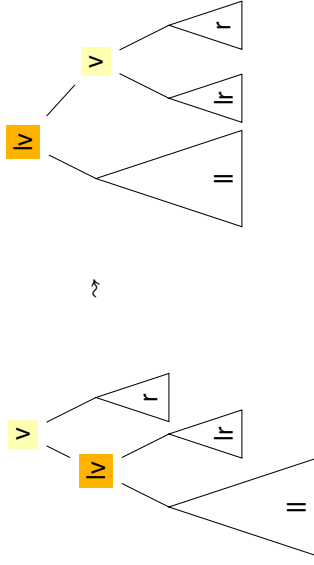
OK

AVL

↔ Rotation gauche

Ensembles

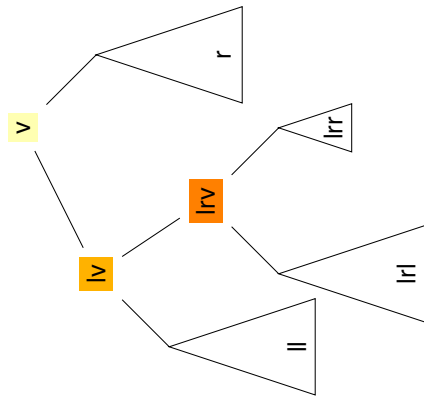
Simple : droite



AVL

Ensembles

Double : gauche-droite



AVL

Ensembles

AVL

Rotations simples et rotations doubles suffisantes

Garantir l'équilibrage à la construction : constructeur (encore plus) futé

```
balance : 'a avl → 'a → 'a avl → 'a avl
```

Par cas suivant le déséquilibre

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 17

Ensembles

AVL

```
let balance l v r =
```

```
  let hl = height l and hr = height r in
```

```
  if hl > hr + 1 then (match l with
```

```
    | Node(1l,lv,lr,_) →
```

```
      if (height 1l ≥ height lr) then node 1l lv (node lr v r)
```

1l plus grand que lr \rightsquigarrow Rot. droite suffit

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 19

Ensembles

AVL

```
let balance l v r =
```

```
  let hl = height l and hr = height r in
```

```
  if hl > hr + 1 then ( ... )
```

1l plus grand que lr \rightsquigarrow Rot. droite suffit

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 18

Ensembles

AVL

```
let balance l v r =
```

```
  let hl = height l and hr = height r in
```

```
  if hl > hr + 1 then (match l with
```

```
    | Node(1l,lv,lr,_) →
```

```
      if (height 1l ≥ height lr) then node 1l lv (node lr v r)
```

1l plus petit que lr \rightsquigarrow gauche-droite

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 20

Ensembles

AVL

```
let balance l v r =  
  let hl = height l and hr = height r in  
  if hl > hr + 1 then (match l with  
    | Node(ll,lv,lr,_) →  
      if (height ll ≥ height lr) then node ll lv (node lr v r)  
      else (match lr with  
        | Node(lrl,lrv,lrr,_) →  
            node (node ll lv lrl) lrv (node lrr v r)
```

ll plus petit que lr ⇒ gauche-droite

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 21

Ensembles

AVL

```
let balance l v r =  
  let hl = height l and hr = height r in  
  if hl > hr + 1 then (match l with  
    | Node(ll,lv,lr,_) →  
      if (height ll ≥ height lr) then node ll lv (node lr v r)  
      else (match lr with  
        | Node(lrl,lrv,lrr,_) →  
            node (node ll lv lrl) lrv (node lrr v r)
```

Pas d'autre cas pour ce déséquilibre !

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 22

Ensembles

AVL

```
let balance l v r =  
  let hl = height l and hr = height r in  
  if hl > hr + 1 then (match l with  
    | Node(ll,lv,lr,_) →  
      if (height ll ≥ height lr) then node ll lv (node lr v r)  
      else (match lr with  
        | Node(lrl,lrv,lrr,_) →  
            node (node ll lv lrl) lrv (node lrr v r)  
        | _ → raise Les_maths_sont_inconsistentes)  
        | _ → raise Les_maths_sont_inconsistentes)
```

Pas d'autre cas pour ce déséquilibre !

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 23

Ensembles

AVL

```
let balance l v r =  
  let hl = height l and hr = height r in  
  if hl > hr + 1 then (match l with  
    | Node(ll,lv,lr,_) →  
      if (height ll ≥ height lr) then node ll lv (node lr v r)  
      else (match lr with  
        | Node(lrl,lrv,lrr,_) →  
            node (node ll lv lrl) lrv (node lrr v r)  
        | _ → raise Les_maths_sont_inconsistentes)  
        | _ → raise Les_maths_sont_inconsistentes)  
      else if hr > hl + 1 then ...
```

Cas de déséquilibre à droite symétrique

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 24

Ensembles

```
let balance l v r =
  let hl = height l and hr = height r in
  if hl > hr + 1 then (match l with
    | Node(ll,lv,lr,_) →
      if (height ll ≥ height lr) then node ll lv (node lr v r)
      else (match lr with
        | Node(lrl,lr,rr,_) →
            node (node ll lv lrl) lrv (node lrr v r)
        | _ → raise Les_maths_sont_inconsistantes)
      | _ → raise Les_maths_sont_inconsistantes)
  else if hr > hl + 1 then ...
  else node l v r
```

Enfin cas sans déséquilibre

AVL

Ensembles

- Recherche
- Ajout
- Retrait
- ...

AVL

Efficace

Léger surcoût

Associations

On veut : $x \mapsto E_x$

Trouver rapidement E_x étant donné x

Opération : recherche

Ranger les E en fonction des x

Recherche efficace \rightsquigarrow Base AVL

Efficacement

Associations

type key

type 'a pmap

empty : 'a pmap

add : key → 'a → 'a pmap

mem : key → 'a pmap → bool

find : key → 'a pmap → 'a

remove : key → 'a pmap → 'a pmap

(* Type des x *)

(* 'a type des E_x *)