Programmation fonctionnelle

ensIIE

Semestre 3 – 2026/27

Institut Mines-Télécom
ensIIE école associée
université PARIS-SACLAY

ensIIE

www.ensiie.fr

Dictionnaire

Aussi appelés tableaux associatifs ou tables d'association

On veut associer des clefs $k \in \mathcal{K}ey$ à des valeurs $v \in \mathcal{V}al$

Il est possible

- de créer un dictionnaire vide
- d'insérer une nouvelle association
- de rechercher à quelle valeur est associée une clef
- de supprimer une association
- de mettre à jour une association
- de fusionner deux dictionnaires
- de parcourir un dictionnaire (*fold*)

Dans la suite, on supposera qu'à une clef n'est associée qu'une seule valeur au maximum (le dictionnaire est une fonction au sens mathématique)

Exemples d'utilisation

Omniprésent en informatique :

- Table des symboles dans un compilateur
  $\mathcal{K}ey=$ symboles, $\mathcal{V}al=$ informations (type, visibilité, ...)
- Système de fichier
  $\mathcal{K}ey=$ chemins, $\mathcal{V}al=$ emplacements disque
- Mémoïsation
  $\mathcal{K}ey=$ arguments, $\mathcal{V}al=$ résultats
- Moteur de recherche
  $\mathcal{K}ey=$ mots-clefs, $\mathcal{V}al=$ pages associées
- Représentation d'un ensemble
  $\mathcal{K}ey=$ élément, $\mathcal{V}al=\{est_dedans\}$
- ...

Spécification algébrique d'un dictionnaire

```
rechercher(creer, k) = ERR
rechercher(inserer(d,k,v),k) = v
rechercher(inserer(d,k,v),k') = rechercher(d,k')
rechercher(supprimer(d,k),k) = ERR
rechercher(supprimer(d,k),k') = rechercher(d,k')
```

Interface

```
module Make(O : OrderedType) : sig
  type 'a map (* maps from O.t to 'a *)

  val empty : 'a map

  val add : O.t -> 'a -> 'a map -> 'a map

  val find: O.t -> 'a map -> 'a

  val remove : O.t -> 'a map -> 'a map

  val update : O.t -> ('a option -> 'a option) -> 'a map
     -> 'a map

  val merge : (O.t -> 'a option -> 'b option -> 'c option)
      'a map -> 'b map -> 'c map

  val fold : (O.t -> 'a -> 'b -> 'b) -> 'a map -> 'b -> 'b
end
```

Implémentation

Utilisation d'arbres AVL

Utilisation des clés comme élément de comparaison

Champ supplémentaire `value` dans les nœuds

Exemple d'utilisation : graphes

Graphe = ensemble de sommets + arêtes (orientées)

Association entre sommets et ensemble des voisins

Exemple : graphes dont les nœuds sont des `string`

```
module StringSet = Set.Make(String)

module StringMap = Map.Make(String)

type graph = StringSet.t StringMap.t
```

Utilisation

Graphe vide :

- `StringMap.empty`

Ajout d'un sommet `v` :

- `StringMap.add v StringSet.empty`

Ajout d'une arête de `u` vers `v` :

- `StringMap.find` + `StringSet.add`

Tester si `v` successeur de `u` :

- `StringMap.find` + `StringSet.mem`

Traitement sur tous les sommets :

- `StringMap.fold`

Traitement sur toutes les arêtes :

- `StringMap.fold` + `StringSet.fold`

Parcours

Trouver un chemin

- attention aux cycles !

Parcourir le graphe

- en profondeur, en largeur
- garder en mémoire les sommets visités

Pour optimiser :

- interruption à l'aide d'une exception