

Manipulation de tableaux

Christophe Fiorio

On souhaite implémenter un type concret TInt dont le type abstrait est le suivant :

```
ullet init: \operatorname{Int} 	imes \operatorname{Int} 	o \operatorname{TInt} / \operatorname{init}(a,n) crée un tableau de n éléments initialisés
    avec la valeur a
• count : TInt \rightarrow Int // donne le nombre d'éléments du tableau ;
// count(init(a,n)) == n
ullet []: Tlnt 	imes Int 	o Int 	o/ retourne l'élément à l'indice donné en paramètre
ullet [\ ]:\mathsf{TInt}	imes\mathsf{Int}	imes\mathsf{Int}	o\mathsf{TInt} // change la valeur de l'entier à l'indice donné par
    l'entier donné en paramètre
// T=init(a,n)\Rightarrow \forall 0 \leq j < n, T[j]==a ; T[j]=k\RightarrowT[j]==k
ullet contains: Tlnt 	imes lnt 	o bool // True si l'entier donnée en paramètre appartient
    au tableau; contains(T,a) \Rightarrow \exists i, T[i] == a
ullet firstIndex: Tlnt 	imes Int|Vide // indice de la première valeur (dans l'ordre
    des indices) donnée en paramètre
// firstIndex(T,a)==i \Rightarrow T[i]==a et \forall 0 \leq j < i, T[j] \neq a
// firstIndex(T,a)==Vide \Rightarrow \forall 0 \leqslant i < n, T[i]\neqa
ullet lastIndex: Tlnt 	imes Int|Vide // indice de la dernière valeur (dans l'ordre
    des indices) donnée en paramètre
// lastIndex(T,a)==i \Rightarrow T[i]==a et \forall i < j < n, T[j] \neq a
// lastIndex(T,a)==Vide \Rightarrow \forall 0 \leqslant i < n, T[i]\neqa
ullet nbOccur: TInt 	imes Int 	imes Int // nombre d'occurrences de la valeur passée en
    paramètre
// nbOccur(T,a)==p\Rightarrow \exists i_0,\ldots,i_p tel que \forall i_{k,O\leqslant k\leqslant p},i_k\in [0,n-1] et T[i_k]==a
ullet min: TInt 	o Int // plus petite valeur de T, min(T)==m \Rightarrow \forall \ 0 \leqslant i < n, T[i] \geqslant m
ullet max: Tint 	o int // plus grande valeur de T, max(T)==m \Rightarrow \forall \ 0 \leqslant i < n, T[i] \leqslant m
```

Exercice 1 structure de données

Proposez une structure de données pour le type abstrait TInt

Exercice 2 algorithmes des fonctions

Implémentez les fonctions du type, en prouvant à chaque fois vos algorithmes et en évaluant leur complexité.

* Exercice 3 Ordre lexicographique

Écrire un algorithme Compare qui, à partir de deux tableaux T1 et T2 à n éléments, a pour résultat :

— la valeur -1 si tous les éléments de T1 sont respectivement avant les éléments de T2 dans l'ordre lexicographique.

```
Exemple: T1=[1,2,1] et T2=[1,2,3]
```

— la valeur 0 si T1 et T2 sont égaux

Exemple: T1=[4,2,5] et T2=[4,2,5]

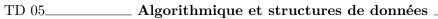
— la valeur 1 sinon.

Exemple : T1=[1,4,1] et T2=[1,2,3]

** Exercice 4 Recherche de sous-tableau

Écrire un algorithme isSub qui, étant donné deux tableaux T1 et T2, indique si T1 est soustableau de T2

 \bullet isSub: TInt \times TInt \to bool // isSub(T1,T2) retourne true si T1 est sous-tableau de T2, et false sinon. T1 est sous-tableau de T2 si





```
// \exists \; 0 \leqslant k < count(T2)\text{, tel que } \forall \; 0 \leqslant i < count(T1), T1[i] == T2[k+i]
// et tel que k + \text{count}(T1) \leqslant \text{count}(T2)
{\bf Exemples}:
- isSub([3,4,5],[1,2,3,4,5,6]) = = true
— isSub([3,4,5],[1,3,2,4,5,6]) == false
— isSub([3,4,5],[1,3,4]) == false
```