## TD3: Arbres binaire de recherche (ABR)

La plupart des structures de données "efficaces" nécessitent la mise en place d'invariants structurels, i.e. des propriétés locales à chaque nœud qui ne peuvent être exprimées par le type seul et qui garantissent globalement l'efficacité des opérations. De telles propriétés sont par exemple :

- des propriétés numériques portant sur la taille, la profondeur, etc.
- des propriétés d'ordre entre éléments.
- des propriétés portant sur des données auxiliaires ajoutées (arbres colorés Rouge-Noir par exemple). Lorsque les invariants sont suffisamment forts, nous pouvons obtenir une représentation canonique.

Enfin, ces invariants doivent obligatoirement être maintenus par les constructeurs abstraits (à supposer que leurs arguments les respectent : programmation offensive), si l'on veut que les structures créées soient toujours correctes. Les types doivent donc être abstraits/privés, afin que l'utilisateur ne puisse pas manipuler "à la main" les constructeurs "concrets" et casser les invariants.

## 1 Spécification

On définit le type récursif suivant :

```
type 'a abr = Vide | Noeud of 'a abr * 'a * 'a abr;;
```

Ce type permet de définir des <u>arbres binaires</u> de <u>recherche</u> (en abrégé abr). Ces arbres permettent de représenter des multi-ensembles sur lequels les opérations ensemblistes sont efficaces par rapport aux mêmes opérations sur des structures de données linéaires comme les listes ou les tableaux. Un abr est :

- Soit Vide
- Soit un Noeud(g,n,d) qui contient un élément n, un sous-arbre gauche g dont tous les éléments sont strictement inférieurs à n, et un sous-arbre droit d dont tous les éléments sont supérieurs ou égaux à n.

Dans le cas des abr, le principe des opérations comme le test d'appartenance, l'ajout ou le retrait d'un élément e à un multi-ensemble est le suivant :

- Soit l'arbre est Vide, il s'agit alors d'un cas terminal;
- Soit l'arbre est un Noeud(g,n,d), alors si e < n (respectivement e > n), on applique récursivement l'opération sur g (respectivement d). Enfin, le cas e=n dépend de la fonction considérée.

## 2 Exercices

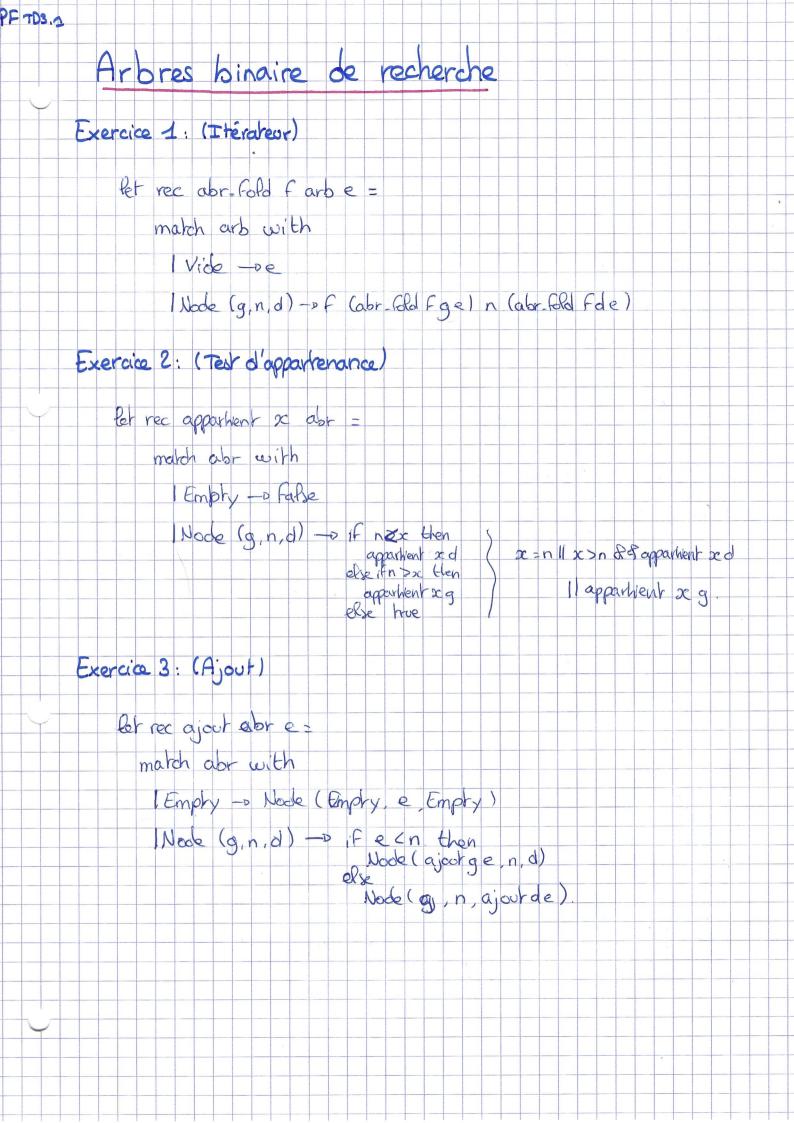
▷ Exercice 1 (Itérateur) Nous souhaitons posséder un itérateur structurel simple qui permet d'implanter rapidement de nombreux algorithmes nécessitant un parcours exhaustif des éléments. Cet itérateur est similaire à l'itérateur sur les arbres binaires standards tree\_fold.

Écrire abr\_fold : (('b -> 'a -> 'b -> 'b) -> 'a abr -> 'b -> 'b) qui parcours tous les éléments d'un abr et leur applique une fonction donnée. Le troisième paramètre correspond à la valeur par défaut pour les abr vides. Par exemple, l'expression suivante :

```
abr_fold f (Noeud(Noeud(Vide, 1, Vide), 2, Noeud(Vide, 3, Vide))) e
```

f (f e 1 e) 2 (f e 3 e)

- Exercice 2 (Test d'appartenance) Écrire une fonction qui teste l'appartenance d'un élément à un multiensemble représenté par un abr. Quelle est la complexité de cette opération?
- Discrete 3 (Ajout) Écrire une fonction qui ajoute un élément à un multi-ensemble représenté par un abr. Cette fonction renvoie le multi-ensemble résultat.
- > Exercice 4 (Cardinal) Écrire une fonction permettant de calculer le cardinal d'un multi-ensemble représenté par un abr, à l'aide de l'itérateur précédent.
- Districte 5 (Aplatissement) Écrire une fonction permettant de construire la liste triée des éléments d'un multi-ensemble représenté par un abr, à l'aide de l'itérateur précédent.
- Exercice 6 (Tri d'une liste) Écrire une fonction triant une liste dans l'ordre croissant par l'intermédiaire d'un abr, en utilisant un itérateur sur les listes et l'itérateur sur les abr.
- Exercice 7 (Retrait de l'élément minimum) Afin de faciliter l'écriture de l'opération de retrait, il est utile de définir une opération auxiliaire qui, à partir d'un abr, renvoie le couple formé de l'élément minimum et de l'arbre privé de cet élément. Écrire une fonction réalisant cette opération.
- ▷ Exercice 8 (Retrait) Écrire une fonction qui retire un élément à un multi-ensemble représenté par un abr, en utilisant la fonction précédente (dans les cas terminaux uniquement). Cette fonction renvoie le multiensemble résultat.



Retrieve a country abor =  march abor with  Wate = 0 (3)  Nobe (g, n, d) = d + cardinal g + cardinal d  Extra cardinal abor = abor fold (for large g = halfle d = exhaulteg forther d) d  Exercise 5: (Aplahissement)  Extra case (i (Tri d'one liste)  abor fold (from light abor p vide in aplathic exempt abor area aport: 'a = a abor = a abor = as a abor = a abor = a abor = area aport: 'a = a a abor = a abor = area aport: 'a = a abor = a abor = area aport: 'a = a abor = a abor = area aport: 'a = a abor = a abor = a abor = a abor = area aport: 'a = a abor = a abor = a abor = a abor = area abor			25																						
Nobe (g, n, d) = 1 + cardinal g + cardinal d    2		Zer	rec	CO	urd	non		abr	-																
Note (g, n, d): 1+ cardinal g + cardinal d.  2- cardinal abor: aby fold (fun bull g - halled = 1+halleg italled) do  Exercise 5: (Aplahissement)  Et aplahissement abor =  abor fold (fun lg n ld = o lg@n::ld) abor CJ.  Exercise 6: (Tri d'une liste)  at his liste l=  let abor = list fold-right ajour l vide in aplahissement abor  auec ajour: a => a abor => a abor.  Exercise 7: (Petrait de l'alternat minimum)  Organini abor:  march abor with  Vide = foldwith ('Adocuside')  (Node (Vide, n, d) = (Node (Innightst, n, d) (Ming. snot)  Coercice 8: (Petrait).  Exercice 8: (Petrait).		}	make	h	als	- w	Jh																		
Exercice 5: (Aplahissement)  Et aplahissement obr =  abr Bla (Fun lg n ld -a lg C n: ld ) abr CJ.  Exercice 6: (Tri d'une Piste)  at hi-liste l=  Ch abr - list Blo right ajour P vide in aplathicement abr  auec ajour: 'a -> 'a abr -> 'a abr.  Exercice 7: (Petrait de Pélénant minimum)  Chamic abr =  march abr with  Vide = Rai Cuith ('Albertide')  (Node (Vide, n d) -> (Node (minighter, n, d)) ming snot)  Cxercice 8: (Petrait)			1	Note	2	2) (	0																		
Exercice 5: (Aplah sement)  Ch aphathisenant obr =  abr-Gold (fun fg n ld -o lg@n:ld) abr []  Exercice 6: (Tri d'une Piste)  Ch thi-liste l=  Cet abr = Pist Gold-right ajout P vide in aplathisenent abrace ajout: `a -> `a abr -> `a abr.  Exercice 7: (Petroit de Pélénant minimum)  Cetra minimum)  Cetra minimum  (Node (Vide, n, d) -e (			1	50€	2 (	3, h	, 01	)	= 1	1+	ca	rde	nof	a	+0	ard	ino	John (	\$.						
Exercice 5: (Aplahissement)  Ch aplahissement obr =  abr Gld (Fun lg n ld -o lg@n:ld) abr CJ.  Exercice 6: (Tri d'une Piste)  Ch hi-liste l=  Ch abr - list fld-right ajour l vide in aplathiserent abraves ajour: 'a -> 'a abor -> 'a abr.  Exercice 7: (Petrait de l'alternit minimum)  Chamin abr =  march abr with  (Node (Vide, n, d) + (d, n)  Node (g, n, d) -> (Node (mini guist n, d) minig snot)  Chercice 8: (Petrait).																									
Charcia 8: Cherrait obre  abrablissement obre  abrablissement obre  abrablissement obre  abrablissement obre  abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  Exercice 7: (Petrait de l'éléctunt minimum)  Chamai abrablissement obre  (Node (Vide, n. d) - ( d , n ).  (Node (S, n. d) - ( Node (minight, n. d) / minight, sod)  (Node (S, n. d) - ( Node (minight, n. d) / minight, sod)  Charcia 8: (Petrait).	er	Car	din	d i	abr		ah	V-h	dd	(	fon	la	arte	3		hat	Sk.	d	0 1	+ rc	u.llo	9-	· had	کی رم	( ) d
Charcia 8: Cherrait obre  abrablissement obre  abrablissement obre  abrablissement obre  abrablissement obre  abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  auex ajact : 'a -> 'a abrablissement obre  Exercice 7: (Petrait de l'éléctunt minimum)  Chamai abrablissement obre  (Node (Vide, n. d) - ( d , n ).  (Node (S, n. d) - ( Node (minight, n. d) / minight, sod)  (Node (S, n. d) - ( Node (minight, n. d) / minight, sod)  Charcia 8: (Petrait).	-von	in	5.	(4	100	h's	Coh	امرما							-										
abr Fold (Fun la n ld - o la C n!: ld) abr CJ.  Exercice 6: (Tri d'une liste)  Out hi-liste l=  Out abr = list Fold-right ajour le vide in aplathisterent abr  auec ajour: 'a -> 'a abr -> 'a abr  Exercice 7: (Petrait de l'élément minimum)  Othermini abr = inh.  I vide -> Fa (unim ('Aborquée').  I Node (yide, n, d) - (Node (iminig) fist, n, d), minig, sno)  Exercice 8: (Petrait).					1000																				
Exercice 6, (Tri d'une Piste)  Cet tri-liste l:  Cet abr = list fold-right ajour l'vide in aplathisterent abraves avec ajour: 'a -> 'a abr -> 'a abr  Exercice 7: (Petrait de l'élément minimum)  Corporate abravith  Vide -> fall (with a l'alore vide)  (Node (Vide n. d) -> (Node (miniglist n. d) / minig. sno))  Exercice 8: (Petrait).  Corroice 8: (Petrait).	1	er c	plat	hisse	e wen	l al	or .	dange												-					
Exercice 6, (Tri d'une Piste)  Cet tri-liste l'=  Cet abr = list Ald-right ajout l'Vide in aplathisterent abr  auec ajout: 'a -> 'a abr -> 'a dor.  Exercice 7: (Petrait de l'élément minimum)  Chamin abr =  march abr with  I Vide -> fai (au M ( Albre vide ).  I Node ( o, n, ol ) -> (Node ( miniglist, n, ol ) ( minigl. snot )  Exercice 8: (Petrait).  Choroice 8: (Petrait).			abr	-Fe	Rd .	Fur	7 {	q r	1 6	4-	-0	Pa	@ 1	1!!	Pol	)	di	rC	J.						
Cet ri-liste l=  Cet abr = list fold-right ajout l Vide in aplathisterent abr  auex ajout: 'a -> 'a abr -> 'a abr  Exercice 7: (Petrait de l'élérant minimum)  Commin abr =  march abr with.  (Noce Vide, n, d) -> (Noce (miniglist, n, d) (miniglist, n, d))  (Noce (s, n, d) -> (Noce (miniglist, n, d) (miniglist, n, d))  Coraice 8: (Petrait).																									
Chri-liste f=  Cet abr = list fold-right ajout P Vide in aplathisterent abr  auex ajout: 'a -> 'a abr -> 'a abr  Exercice 7: (Petrait de Pélérunt minimum)  Chamir abr =  march abr with  Vide > fallwith ('Albrevide')  (Node (Vide, n, d) -> (Node (miniglist, n, d) (minig) snd)  Node (g, n, d) -> (Node (miniglist, n, d) (minig) snd)  Craraice 8: (Petrait).	Exer	ci ce	16,	(-	Tra'	do	ne	ei:	she)								5 9								
Corcia 8: Cheliant abre a land to land the land the land to land the land the land to land the land the land to land the land the land to land the l																									
Evernice 7: (Petrait de l'élécturit minimum)  Cirpemini abr :  march abr yith.  Vide - fai (with (Abbeyide)).  [Node (Vide, n, d) - (Mode (miniglist, n, d), miniglist, n, d)]  [Node (g, n, d) - (Mode (miniglist, n, d), miniglist, n, d)]  Evernice 8: (Petrait).							CO	d i	h		nio	ch	0	Vo	6	10	0	O Ca	W'c	CON	200		abr		
Exercice 7: (Petrait de l'élément minimum)  Cetremni abre with.  Vide = fai (with ( Abrevide ).  ( Node ( Vide, n, d) - ( d, n).  ( Node ( g, n, d) - ( Node ( (mini g) fot, n, d) , minig). sno))  Crercice 8: (Petrait).									T		7			-					,,,,,		ev i i				
Cheroice 8: Cherout.					evec	a y	201				~V	4			P	0. (	240	r.,							
Chercia 8: Chercuit.						4	ļ.	011	Oc																
march dor with.  Vibe - fallwith ( Abrevide ).  ( Mode (Vide, n, d) - ( d), n).  ( Node ( g, n, d) - ( Mode ( mini g), sr, n, d) , minig), sno )  Exercica 8: (Retrait).	txer	rcice	. 1:		erro	rit	de	*e	161	Len	r	ni	nın	NUV	n										
marcin dor with.  Vibe - Fallwith ( Abrevide).  ( Mode (Vide, n, d) - ( d), n).  ( Node (g, n, d) - ( Mode (mini g), sr, n, d), minig), sno)  Exercica 8: (Refractio).	Q	et you'v	ninc	ab	r =																				
Node (Vide, n, d) - (Mode (minigless, n, d), minigl. sno)  Exercice 8: Crestrait.																									
Noce (Vide, n, d) - (Node (minight, n, d), minight, snd)  Correice 8: Cremair abre =			marc	M	apr	Ust	111				-														
Exercice 8: Crestrair).								0.7	(h		Ado	A (	ndo	1)											
Exercice 8: Cretrair).			1	V.6	6 -	o Fc	u G			•						10	1								
de rec remair abre = .				V (5)	e (	Vida Vida	1. G	n, d	) -				d			7		,	}			The second section of the sect		-	
et rec retrait dor e = .				V (5)	e (	Vida Vida	1. G	n, d	) -				d			7		, 0	) /	M,	714	g).	Snc	ر ارد	
				V (S)	e (	Vide 3	e,	n, d	) -				d			7		, 0		M.	nia	g).	Snc	-5/ )	
if apparhient e alor then match about then match about with 1 Node (a, mini d) and mini allest).	Even			V (S)	e (	Vide 3	e,	n, d	) -				d			7		, 0	) (	M.	nia	3).	Snc	3/ )	
match aby with  1 Nodo (g n, d) to if (n = e) (en Node (a, min) d snd, mini all (st)	Even	cu'co.	8:	VIO NOO	e (	Sit)	2 , ) m	or			Yoc	6	( (n	ini	9),		n								
	Even	cu'co.	8:	VIO NOO	e (	Sit)	2 , ) m	or			Yoc	6	( (n	ini	9),		n								