Contents

1 Module `Arn_doc` : Arbres Rouge-Noirs.

Ce module realise des arbres rouges-noirs (auto-equilibrant) fonctionnels. Les operations de Recherche, Insertion et Suppression y sont garanties en temps logarithmique en le nombre d'elements.

Toutes les fonctions presentes dans l'implementation sont documentees ici, y compris celles qui ne sont que des auxiliaires a d'autres fonctions plus utiles.

NB : Ceci n'est donc **pas** un bon exemple d'interface : absolument rien n'est cache, rien n'est abstrait. Pire encore, certains commentaires sont des explications pour aider a implementer alors que l'interface doit justement abstraire l'implementation. Mais cela me permet de garder le .ml relativement court, en deplacant la documentation en dehors.

En resume, ceci n'est pas une bonne interface, mais c'est un bon enonce de TP/DM.

```
type couleur =
  | Rouge
  | Noir
```

Type des couleurs des noeuds

```
type 'a arn =
  | Nil
  | Node of couleur * 'a arn * 'a * 'a arn
```

Type des Arbres Rouges-Noirs d'etiquettes de type `'a`

1.1 Initialisateurs

```
val empty : unit -> 'a arn
```

Renvoie un Arbre Rouge-Noir correct vide.

1.2 Accesseurs

Proprietes d'un noeud

Ces fonctions confondent noeud et arbre : autrement dit, elles s'appliquent au noeud-racine.

```
val etiquette : 'a arn -> 'a
```

Renvoie l'etiquette d'un noeud.

```
val gauche : 'a arn -> 'a arn
```

Renvoie l'enfant gauche d'un noeud.

```
val droite : 'a arn -> 'a arn
```

Renvoie l'enfant droit d'un noeud.

```
val est_rouge : 'a arn -> bool
```

Teste si un noeud est Rouge.

`val est_noir : 'a arn -> bool`

Teste si un noeud est Noir. Par convention, les Nil sont aussi Noirs.

Proprietes d'un arbre

`val hauteur : 'a arn -> int`

Renvoie la hauteur d'un (sous-) arbre rouge-noir (presque) correct.

`val hauteur_noire : 'a arn -> int`

Renvoie la hauteur noire d'un (sous-) arbre rouge-noir (presque) correct.

`val recherche : 'a -> 'a arn -> bool`

Recherche si un element est present dans un (sous-) arbre rouge-noir (presque) correct.

`val minimum : 'a arn -> 'a`

Renvoie le minimum d'un (sous-) arbre rouge-noir (presque) correct.

`val maximum : 'a arn -> 'a`

Renvoie le maximum d'un (sous-) arbre rouge-noir (presque) correct.

1.3 Transformateurs

Recoloriages

`val devient_rouge : 'a arn -> 'a arn`

Colorie un noeud en Rouge.

`val devient_noir : 'a arn -> 'a arn`

Colorie un noeud en Noir.

Rotations

`val rotation_droite : 'a arn -> 'a arn`

Applique une rotation droite.

`val rotation_gauche : 'a arn -> 'a arn`

Applique une rotation gauche.

Insertion

`val corrige_rouge : 'a arn -> 'a arn`

Corrige un sous-arbre dont un enfant est presque correct en rouge en un sous-arbre rouge-noir correct.

Plus precisement, sous l'hypothese qu'il existe au plus un enchainement de deux rouges, et qu'ils sont situes a profondeur 1 puis 2, `corrige_rouge` corrige ce probleme.

NB : la racine peut devenir rouge, et donc creer un enchainement de rouges plus haut dans l'arbre.

`val insere_aux : 'a -> 'a arn -> 'a arn`

Insere l'element dans un arbre. et corrige l'enchainement de rouges grace a `corrige_rouge`.

Invariant : apres un appel a insere_aux , l'element est insere ET il n'y a pas d'enchainement de rouge. Par contre la racine a pu devenir rouge.

`val insere : 'a -> 'a arn -> 'a arn`

Insere un element dans un arbre a l'aide de `insere_aux`. Renvoie un arbre rouge-noir correct, i.e. garantit le maintien des proprietes d'ARN.