Tas min gauchers

Contexte Les arbres binaires de recherche sont des structures qui permettent de représenter des ensembles finis, et où les opérations efficaces sont la recherche, l'ajout et la suppression. En revanche, l'union de deux ensembles ne bénéficie pas de la structure d'arbre binaire de recherche.

Dans cet exercice nous allons nous intéresser à une structure de donnée où, au contraire, l'union est l'opération que l'on cherche à réaliser efficacement: les *tas min gauchers*, qu'on appellera simplement TMG. Contrairement à la structure d'ABR, où le coût des opérations est typiquement proportionnelle à la hauteur (c'est-à-dire le plus long chemin) de l'arbre, dans les TMG le coût des opérations efficace est proportionnel au rang : la longueur du *plus court chemin* de la racine à un sous-arbre vide. Donc là où un ABR a intérêt à être bien équilibré, les TMG sont des arbres très déséquilibrés.

Question 1 Écrivez une une *spécification* et une *définition* pour la fonction `ab-rang` qui, étant donné un arbre binaire (quelconque) `A`, calcule le rang de `A`. C'est-à-dire la longueur du plus court chemin entre la racine de `A` et un de ses sous-arbres vides (si `A` est vide, on prendra la convention `ab-rang A` vaut `0`).

Question 2 On définit un TMG comme étant un arbre binaire `A` de type `ArbreBinaire[Nombre]`, tel que :

- L'étiquette de la racine de `A` est le *plus petit* élément de `A`
- Le rang du sous-arbre droit de `A` est plus petit que le rang du sous-arbre gauche de `A`.
- Les sous-arbres droit et gauche de `A` sont tous les deux des TMG

On appellera `Tas` le type des TMG.

Notez comme dans un TMG, il est plus efficace de calculer le rang que dans un arbre binaire quelconque. Écrivez une *spécification* et une *définition* pour la fonction `tas-rang` qui utilise les propriétes de TMG si dessus pour calculer efficacement le rang d'un TMG.

Question 3 Donnez une *spécification* et une *définition* pour une fonction `tas-noeud` qui étant donné un nombre `r` et deux TMG `X` et `Y` construit un TMG dont l'étiquette est `r` et les deux sous-arbres sont `X` et `Y`. Veillez à bien respecter la condition de rang des `tmg`, en remarquant qu'il s'agit de la seule condition sur l'ordre des deux sous-arbres.

Question 4 Soit `X` et `Y` deux TMG. Supposons qu'ils ont tous les deux une étiquette et que l'étiquette de la racine de `X` est plus petite que celle de `Y`. Pour prendre l'union de `X` et `Y`, on doit garder comme étiquette celle de la racine de `X`. On a trois arbres pour constituter les deux sous-arbres de l'union: `(ab-gauche X)`, `(ab-droit X)` et `Y`. Comme le rang de `(ab-gauche X)` est plus grand que celui de `(ab-droit X)`, on choisit de prendre comme sous-arbres de l'union, `(ab-gauche X)` et l'union de `(ab-droit X)` et de `Y`.

Écrivez une *spécification* et une *définition* de la fonction `tas-union` qui calcule l'union de deux TMG en utilisant la méthode ci-dessus. Pensez à utiliser la fonction `tas-noeud` définie à la question précédente pour vous assurer de ne pas violer la condition de rang.

Question 5 Dans un `tmg`, l'élément minimum est directement accessible. Mais grâce à la fonction `tas-union` de la question précédente, il est facile aussi de retirer l'élément minimum d'un `tmg`: en effet, il suffit de prendre l'union des sous-arbres droit et gauche.

Donnez une *spécification* et une *définition* de la fonction `tas-extrait-min` qui étant donné un TMG `X` renvoit un *couple* dont la première composante est l'élément minimum de `X` et la seconde composante le TMG `X` privé de son élément minimum.

Question 6 Écrivez une *spécification* et une *définition* de la fonction `liste-de-tas` qui, étant donné un TMG `X`, renvoit la liste *triée* des éléments de `X`. Utilisez la fonction `tas-extrait-min` de la question précédente.

Question 7 On peut se servir de la fonction `liste-de-tas` pour trier une liste. Mais pour cela, il faut tout d'abord convertir une liste (de nombres) en un TMG. On procédera comme suit :

- On part d'une liste d'arbres singletons (c'est-à-dire de la forme `(tas-noeud x (ab-vide) (ab-vide))`). Obtenue, par exemple, en utilisant `map` sur la liste d'origine.
- On prend les éléments de la liste deux à deux, et on en prend l'union. C'est-à-dire que si on a une liste $(X_1\ X_2\ X_3\ X_4\ \ldots\ X_{2n-1}\ X_{2n})$, on calcule $(X_1 \cup X_2\ X_3 \cup X_4\ \ldots\ X_{2n-1} \cup X_{2n})$. Si il y a un $2n+1$-ième élément, on le laissera tel quel.
- On recommence jusqu'à ce que la liste n'est plus qu'un élément, qu'on retourne.

Écrivez une *spécification* et une *définition* de la fonction `tas-de-liste` qui implémente la méthode ci-dessus. Vous pouvez en profiter pour constater, en utilisant `ab-affiche`, que les TMG sont effectivement très déséquilibrés.

Question 8 En utilisant les fonctions `liste-de-tas` et `tas-de-liste`, écrivez une *spécification* et une *définition* de la fonction `trie-liste` qui, étant donnée une liste `l` de nombres, calcule la liste triée correspondante.

Pour créer de grandes listes à triées, vous pouvez utiliser la fonction `random` de votre carte de référence.

Question subsidiaire Cette question d'approfondissement se fait dans une copie du fichier où vous avez répondu aux questions précédentes.

Remarquez qu'on calcule le rang des deux sous-arbres à chaque application de `tas-noeud`. Ce calcule a beau être rapide, il est néanmoins redondant: on calcule souvent le rang d'arbre dont on a déjà calculé le rang. Pour améliorer cet état de fait, il y a une solution simple: garder, en plus de l'étiquette, le rang dans les noeuds du TMG. C'est-à-dire que les TMG sont désormais représentés comme des `ArbreBinaire[COUPLE[nat Nombre]]` : dans chaque nœud, la première composante de l'étiquette est le rang du nœud, et la seconde composante est l'étiquette en tant que tas.

Nous allons modifier les réponses des questions précédentes pour utiliser cette représentations. Commencez par définir la fonction `tas-etiquette` qui accède à la seconde composante de l'étiquette de la racine. Remplacez `ab-etiquette` pas `tas-etiquette` dans le reste de votre fichier.

Modifiez la fonction `tas-rang` pour utiliser l'information de rang qui est désormais stockée dans la racine.

Il ne reste plus qu'à modifier `tas-noeud` pour garder le rang dans la racine. Vous pouvez vérifier que seuls ces trois changement étaient nécessaires en exécutant votre fichier et en vérifiant que les tests y passent toujours.