TD 3

1 Files

Une *file* est une liste linéaire où les insertions se font toutes d'un même coté et les suppressions toutes de l'autre côté (contrairement aux piles où les insertions et les suppressions se font du même coté).

1. Définir le type abstrait `file`.

2. Décrire une implantation du type `file`.

3. Implanter une file à l'aide de deux piles. Calculer le coût amorti d'une opération.

4. Implanter une pile à l'aide de deux files. Calculer le coût amorti d'une opération.

2 Expressions arithmétiques

1. La grammaire des expressions arithmétiques en notation préfixe est :

$$\text{exp} : := \text{réel} \mid \text{op\_bin exp exp} \mid \text{op\_un exp} \mid (\text{exp})$$

- Ecrire une fonction récursive `eval` pour évaluer une expression en notation prefixe.
- Donner une version itérative de la fonction `eval`.

2. Considérons maintenant les expressions en notation infixe données par la grammaire :

$$\text{exp} : := \text{réel} \mid \text{exp op\_bin exp} \mid \text{op\_un(exp)} \mid (\text{exp})$$

Ecrire une fonction qui construit l'arbre représentant une expression infixe en respectant les règles de priorité et d'associativité.

3 Arbres binaires de recherche

1. Ecrire une fonction `supprimerABR` qui prend en argument une clé `k` et un arbre binaire de recherche `t` et qui supprime `k` de `t`.

2. Ecrire une fonction `scinderABR` qui prend en argument un arbre binaire de recherche `t` et une clé `k` et qui retourne une décomposition `(t1,t2)` de `t` telle que :

$$\texttt{clés(t)} = \texttt{clés(t1)} \cup \texttt{clés(t2)}$$
$$\wedge\ \texttt{clés(t1)} = \{\texttt{c} \in \texttt{clés(t)} \mid \texttt{c} \leq \texttt{k}\} \wedge \texttt{clés(t2)} = \{\texttt{c} \in \texttt{clés(t)} \mid \texttt{c} > \texttt{k}\}$$

3. Ecrire une fonction `fusionnerABR` qui prend en argument deux arbres binaires de recherche `t1` et `t2`, tels que

$$\max\{\texttt{clés(t1)}\} < \min\{\texttt{clés(t2)}\}$$

et qui retourne un arbre binaire de recherche `t` tel que

$$\texttt{clés(t)} = \texttt{clés(t1)} \cup \texttt{clés(t2)}.$$

4 Arbres binaires de recherche balisés

Dans un *ABR balisé*, les couples `(clé,info)` ne sont qu'aux feuilles et les nœuds internes ont tous exactement 2 fils et contiennent des balises qui permettent d'orienter la recherche. Si x est un nœud interne, alors

$$\{\text{clés de } x.g\} < x.\text{balise} \leq \{\text{clés de } x.d\}.$$

1. Montrer que $n(a) = 2f(a) - 1$ si a est un ABR balisé.

2. Ecrire les procédures d'insertion et de suppression dans un ABR balisé.

3. Ecrire une procédure pour transformer en temps linéaire un ABR balisé en un ABR classique ayant la même structure.

4. Ecrire la procédure inverse.

5 Dénombrement sur les arbres binaires

Dans cet exercice, n dénote le nombre de nœuds d'un arbre binaire, f son nombre de feuilles et h sa hauteur. Tous les arbres considérés seront supposés non vides.

1. Quelle est la hauteur maximale d'un arbre à n nœuds ?

2. Quel est le nombre maximal de feuilles d'un arbre de hauteur h ?

3. Quel est le nombre maximal de nœuds d'un arbre de hauteur h ?

4. Quelle est la hauteur minimale d'un arbre de n nœuds ?

5. Montrer que le nombre de branches vides − nombre de fils gauches et de fils droits vides − d'un arbre à n nœuds est égal à $n+1$.

6. Montrez que le nombre de feuilles est inférieur ou égal à $\frac{n+1}{2}$ ($f \leq \frac{n+1}{2}$) et qu'il y a égalité si et seulement si chaque nœud de l'arbre est soit une feuille, soit a deux fils.

7. Montrez que le nombre de feuilles d'un arbre est égal au nombre de nœuds de degré deux, plus 1.

6 ABR – Complexité en moyenne

1. Calculer la hauteur moyenne d'un ABR construit aléatoirement.
Univers : $\mathfrak{S}_n$ avec distribution uniforme.
Hauteur : $H_n : \mathfrak{S}_n \to \mathbb{N}$ où $H_n(\sigma)$ pour $\sigma \in \mathfrak{S}_n$ est la hauteur de l'arbre obtenu par insertions successives de $\sigma(1), \ldots, \sigma(n)$

2. Calculer le coût moyen d'une recherche fructueuse/infructueuse dans un ABR construit aléatoirement.