

## Solution du devoir 2

**Auteur : (Indiquer votre nom ici)**

Question	1	2	3	4	Total
Sur	30	15	15	40	100
Note					

1. (30 points) Nous avons vu en classe que la complexité amortie d’une insertion dans un tableau redimensionnable est  $\mathcal{O}(1)$ . Supposons que nous souhaitions également compresser le tableau de moitié lorsque  $1/4$  ou moins des cases sont occupées.

Soit  $\alpha(T) = T.\text{utilisé} / T.\text{capacité}$ . On définit une fonction potentielle

$$\Phi(T) = \begin{cases} 2 \cdot T.\text{utilisé} - T.\text{capacité}, & \text{si } \alpha(T) \geq 1/2; \\ T.\text{capacité}/2 - T.\text{utilisé}, & \text{si } \alpha(T) < 1/2. \end{cases}$$

(a) (4 points) Montrez que  $\Phi(T)$  est toujours positive ou nulle.

(b) (6 points) Montrez que

$$\Phi(T) = \begin{cases} 0, & \text{si } \alpha(T) = 1/2; \\ T.\text{utilisé}, & \text{si } \alpha(T) = 1 \text{ ou } \alpha(T) = 1/4. \end{cases}$$

- (c) (20 points) Montrez que le coût amorti d’une suppression est constant. *Remarque :* Vous devez distinguer plusieurs cas, selon la valeur de  $\alpha(T_{i-1})$  et selon qu’il y a une compression ou non.

**Solution:** Ajouter votre solution ici

2. (15 points) En géométrie algorithmique, il est possible de représenter des solides à l’aide d’opérations ensemblistes. Par exemple, à la figure 1, on construit un solide à l’aide de la réunion  $\cup$ , l’intersection  $\cap$  et la différence  $-$ . Cet objet est appelé *solide de géométrie de construction*. On suppose que les feuilles sont des solides simples, qui offrent des fonctions  $S.\text{ISINSIDE}(P)$ ,  $S.\text{ISOUTSIDE}(P)$  et  $S.\text{ISONBOUNDARY}(P)$  qui retournent vrai respectivement selon que le point  $P$  se trouve à l’intérieur, à l’extérieur ou sur la frontière du solide  $S$ .

Étendez les fonctions  $S.\text{ISINSIDE}(P)$ ,  $S.\text{ISOUTSIDE}(P)$  et  $S.\text{ISONBOUNDARY}(P)$  au cas où  $S$  est un solide de géométrie de construction. Vous pouvez supposer que les seuls opérateurs ensemblistes permis sont  $\cup$ ,  $\cap$  et  $-$ . De plus, vous pouvez supposer

qu'il existe une fonction  $S.\text{IsATOMIC}()$  qui retourne vrai si  $S$  est un solide de base (autrement dit, c'est une feuille dans la représentation), une fonction  $S.\text{OPERATOR}()$  qui retourne la valeur `INTER`, `UNION` ou `DIFF` (comme pour un type énumératif) selon que le solide courant est obtenu par l'opération ensembliste  $\cap$ ,  $\cup$  ou  $-$ , ainsi que des fonctions  $S.\text{LEFT}()$  et  $S.\text{RIGHT}()$  qui retournent le fils gauche et le fils droit de  $S$  lorsqu'il n'est pas un solide de base (autrement dit, c'est un noeud interne dans la représentation).

**Solution:** Ajouter votre solution ici

3. (15 points) Soit  $n$  un entier positif et  $N = \{1, 2, \dots, n\}$ . Dans cette question, on s'intéresse à la représentation d'une famille de sous-ensembles de  $N$  à l'aide d'un arbre et/ou. Par exemple, on peut représenter la famille

$$\{\{1, 3\}, \{1, 4\}, \{2, 4\}, \{2, 3\}, \{2, 5\}\}$$

à l'aide de l'arbre illustré dans la figure 2. Ainsi, un noeud  $\vee$  signifie qu'on peut choisir n'importe quel enfant alors qu'un noeud  $\wedge$  indique qu'on doit choisir toutes les combinaisons possibles de chacun des enfants. Supposez qu'aucun sous-ensemble n'est répété dans l'arbre et/ou, et que les valeurs qui apparaissent dans chaque sous-arbre d'un noeud  $\wedge$  sont distinctes. De plus, supposez que les fonctions suivantes sont déjà implémentées et s'effectuent toutes en  $\mathcal{O}(1)$  :

- Pour tout arbre  $T$ ,  $T.\text{CONTENU}()$  retourne la valeur contenue ( $\wedge$ ,  $\vee$  ou un élément de  $N$ ) dans le noeud racine de  $T$ ;
- Pour tout arbre  $T$ ,  $T.\text{SOUSARBRES}()$  retourne une liste ordonnée des sous-arbres de  $T$ ;
- Pour tout arbre  $T$ ,  $T.\text{CARDINALITE}()$  retourne le nombre total de sous-ensembles représentés par  $T$ ;
- Une fonction  $\text{UNIFORME}()$  qui retourne un nombre réel aléatoire entre 0 inclusivement et 1 exclusivement.

Écrivez une fonction dont la complexité est  $\mathcal{O}(n)$  qui retourne un sous-ensemble aléatoire représenté par un arbre et/ou quelconque, où  $n$  est le nombre de noeuds dans l'arbre. Chaque sous-ensemble doit avoir la même probabilité d'être choisi (dans l'exemple ci-haut, chaque ensemble a donc une probabilité  $1/5$  d'être choisi).

**Solution:** Ajouter votre solution ici

4. (40 points) En utilisant votre langage de programmation préféré parmi Java, C/C++ et Python, implémentez

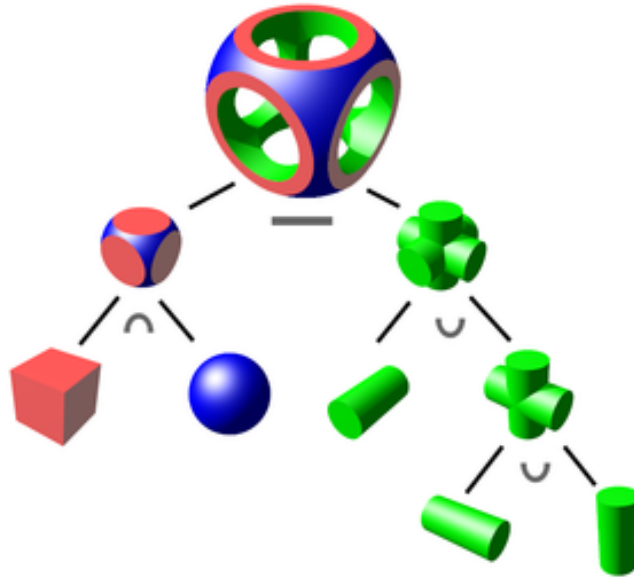


Figure 1: Un solide de géométrie de construction.

- (a) (10 points) un algorithme permettant de transformer un arbre rouge-noir en un arbre 2-3-4;
- (b) (10 points) un algorithme permettant de transformer un arbre 2-3-4 en un arbre rouge-noir.

Dans les deux cas, la transformation doit préserver le contenu de l'arbre (c'est-à-dire que les mêmes clés doivent apparaître). Illustrez le fonctionnement de vos transformations sur au moins 2 exemples chacune.

*Note :* Il n'est pas nécessaire d'implémenter les opérations d'insertion et de suppression dans les arbres rouge-noir et 2-3-4 pour répondre à cette question. Vous pouvez représenter les structures de données simplement par des tuples (en Python) ou à l'aide de classes (tous les langages) sans les valider (on suppose que l'utilisateur qui construit les arbres respecte la spécification).

- (c) (20 points) Dites si vos transformations sont injectives/surjectives/bijectives, en le démontrant dans chaque cas ou en fournissant un contre-exemple.

**Solution:** Ajouter votre solution ici

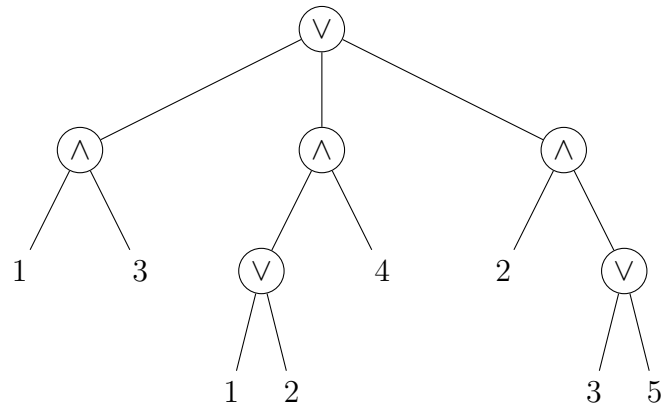


Figure 2: Un arbre et/ou représentant une famille de 5 sous-ensembles.