Contrôle continu

La durée du contrôle continu est 1h30. Il est constitué de trois exercices indépendants. Toute question non résolue peut être admise dans la suite. Les réponses doivent être correctement rédigées. Aucun document n'est autorisé.

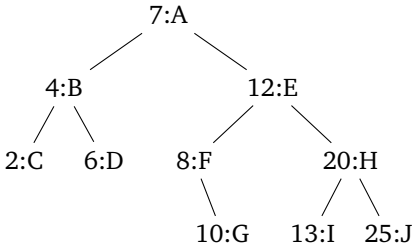
Exercice 1 (sur 5 pts). *Dessins*

1. Soit $T = [16, 12, 1, 3, 10]$.

i. Dessiner ce tas sous forme d'arbre binaire.

ii. On insère dans T les valeurs suivantes (dans cet ordre) : 5, 7, 9. Donner la valeur de T après chaque insertion, et dessiner le tas final sous forme d'arbre binaire.

2. On considère l'arbre binaire de recherche ci-contre où les clefs sont des entiers et les valeurs des lettres. Dessiner l'arbre binaire de recherche obtenu en appliquant l'algorithme de suppression de la racine.



3. Quelles sont les deux grandes méthodes de résolution des collisions dans une table de hachage ? Donner leurs noms et décrire leur principe en une phrase chacune.

Exercice 2 (sur 6 pts). *Tas gauchistes*

On étudie dans cet exercice une réalisation possible des files de priorités.

Soit $\mathcal{A}$ un arbre binaire. On appelle *poids* d'un nœud x, noté $w(x)$, le nombre de nœuds dans le sous-arbre dont x est la racine. Un arbre binaire $\mathcal{A}$ est dit *gauchiste* si pour tout nœud x, $w(\text{ENFANTG}(x)) \geq w(\text{ENFANTD}(x))$ (en notant $w(\emptyset) = 0$ pour l'arbre vide).

Un *tas gauchiste* est un arbre binaire gauchiste dont chaque nœud contient un triplet (v, p, w) où v est la *valeur* du nœud, p sa priorité et w son poids, et qui *tassé*, c'est-à-dire tel que la priorité d'un nœud est toujours inférieure ou égale à celle de son parent.

1. Montrer que pour tout nœud x d'un arbre binaire gauchiste, $w(\text{ENFANTD}(x)) \leq \frac{1}{2}(w(x) - 1)$.

Étant donnés deux tas gauchistes $\mathcal{A}$ et $\mathcal{B}$, on définit une opération de FUSION qui construit un nouvel arbre gauchiste contenant l'union des valeurs de $\mathcal{A}$ et $\mathcal{B}$. L'algorithme est le suivant : 1. on choisit l'arbre dont la racine a la plus grande priorité, disons que c'est $\mathcal{A}$, et on note r sa racine et $\mathcal{G}$ et $\mathcal{D}$ ses sous-arbres gauche et droit ; 2. on appelle FUSION$(\mathcal{D}, \mathcal{B})$ pour obtenir $\mathcal{C}$; 3. on construit l'arbre de racine r et de sous-arbres $\mathcal{G}$ et $\mathcal{C}$, en mettant à jour le poids de r.

2. **i.** Écrire l'algorithme en détail, en utilisant le TAD arbre binaire [1]. *Attention, la description de l'algorithme est incomplète : déterminer des cas de base ; traiter aussi le cas où $\mathcal{B}$ a la racine de plus grande priorité ; choisir le sous-arbre gauche et le sous-arbre droit entre $\mathcal{G}$ et $\mathcal{C}$ pour que l'arbre renvoyé soit gauchiste.*

ii. Montrer que $\mathcal{C}$ est tassé suite à l'opération FUSION.

iii. Montrer que le nombre de comparaisons effectuées par FUSION$(\mathcal{A}, \mathcal{B})$ est $O(\log(1 + mn))$ où m est le nombre de nœuds dans $\mathcal{A}$ et n celui dans $\mathcal{B}$.

3. On utilise les tas gauchistes pour réaliser une file de priorité. On réalise toutes les opérations à l'aide de FUSION.

i. Écrire l'opération INSÉRER(F, v, p) à l'aide de FUSION et analyser sa complexité.

ii. Écrire l'opération EXTRAIRE(F) à l'aide de FUSION et analyser sa complexité.

1. Rappel : NVARBRE$(r, \mathcal{G}, \mathcal{D})$ permet de créer un nouvel arbre.

Exercice 3 (sur 9 pts). *Conversion en décimal*

Dans cet exercice, on souhaite convertir un entier binaire en sa représentation décimale. *On suppose avoir accès à des algorithmes d'addition (*ADD*) et de multiplication (*MUL*) pour les entiers écrits en décimal. Sauf mention contraire explicite, les complexités seront exprimées en nombre d'opérations chiffre-à-chiffre.*

1. Soit A et B deux entiers de n chiffres décimaux.

- **i.** Quel est le coût du calcul de $\text{ADD}(A, B)$?
- **ii.** Quel est le coût du calcul de $\text{MUL}(A, B)$ en utilisant l'algorithme *naïf* ? et en utilisant l'algorithme de Karatsuba ?

Dans la suite, on suppose qu'on utilise l'algorithme de Karatsuba pour MUL.

2. Soit w un mot binaire de longueur n, qui représente l'entier $N(w) = \sum_{i=0}^{n-1} w_{[i]} 2^i$. Pour calculer sa représentation décimale $\text{DÉC}(w)$, on utilise une approche « diviser pour régner » en écrivant $w = w_0 \cdot w_1$ où w_0 est de longueur $\lfloor n/2 \rfloor$ et w_1 de longueur $\lceil n/2 \rceil$ (« $\cdot$ » représente la concaténation).

- **i.** Exprimer $N(w)$ en fonction de $N(w_0)$, (w_1) et d'une puissance de 2.
- **ii.** En déduire un algorithme « diviser pour régner » pour calculer $\text{DÉC}(w)$. *Indication. On peut aussi appliquer* DÉC *à une puissance de* 2.
- **iii.** Analyser sa complexité.

3. On veut améliorer l'algorithme précédent en calculant directement la représentation décimale d'une puissance de 2 à l'aide de l'exponentiation rapide.

- **i.** Écrire un algorithme qui calcule 2^n par exponentiation rapide.
- **ii.** Combien d'appels à MUL sont effectués par cet algorithme ?
- **iii.** Analyser la complexité de l'algorithme.
- **iv.** Quelle est la complexité de $\text{DÉC}(w)$ si on remplace un appel récursif par l'exponentiation rapide ?