

# Programmation avancée

## Répétition 4: Tas, file à priorité et arbre

Romain MORMONT

15 novembre 2019

### Exercice 1

- (a) Combien y a-t-il au minimum (resp. maximum) d'éléments dans un tas de hauteur  $h$  ?
- (b) Où l'élément le plus petit réside dans un tas-max ?
- (c) Quelle est la relation entre un tas-min et à tableau trié (par ordre croissant) ?
- (d) Est-ce que le tableau  $[23, 17, 14, 6, 13, 10, 1, 5, 7, 12]$  est un tas-max ?

### Exercice 2

Dessiner tous les tas-MIN possibles avec l'ensemble des clés  $\{A, B, C, D, E\}$ , où chaque clé n'apparaît qu'une seule fois.

### Exercice 3

Soit un tas-MIN  $H$ . Illustrer l'exécution des opérations suivantes :

```
heapInsert(H, 5)
heapInsert(H, 7)
heapInsert(H, 4)
heapInsert(H, 8)
heapExtractMin(H)
heapInsert(H, 3)
heapInsert(H, 4)
heapInsert(H, 10)
heapExtractMin(H)
heapInsert(H, 3)
```

### Exercice 4

- (a) Comment implémenter une pile au moyen d'une file à priorité ?
- (b) Comment implémenter une file au moyen d'une file à priorité ?
- (c) Comment implémenter une file aléatoire au moyen d'une file à priorité ?
- (d) Comment implémenter une file à priorité maximum avec un tas-min ?
- (e) Comment implémenter une file à priorité qui soit  $\Theta(1)$  pour l'insertion et  $O(n)$  pour l'extraction ?

## Exercice 5

Le parcours préfixe d'un arbre binaire donne :

A B C - - D - - E - F - -

En donner les parcours infixes et postfixes. Les lettres correspondent aux noeuds internes et les tirets à des feuilles de l'arbre.

## Exercice 6

Soit un arbre  $T$ .

- (a) Ecrire une méthode `size(T)` qui détermine le nombre d'éléments dans l'arbre  $T$ .
- (b) Ecrire une méthode `isBalanced(T)` qui détermine si l'arbre  $T$  est équilibré  
**Note :** Un arbre est équilibré si pour chacun de ses noeuds, la hauteur du sous-arbre de droite de ce noeud ne diffère pas de la hauteur du sous-arbre de gauche de plus d'une unité ;
- (c) Ecrire une méthode `isComplete(T)` qui détermine si l'arbre  $T$  est complet ou non.
- (d) Ecrire une méthode `mirror(T)` qui retourne l'arbre miroir de l'arbre  $T$ .
- (e) Ecrire une méthode `printLowerThan(T, x)` qui imprime les valeurs contenues dans  $T$  inférieures ou égales à  $x$ . Quelle est la complexité de cette opération ?

*Bonus :* Pour chaque fonction, décrivez un algorithme a) récursif et b) non-récursif.

*Bonus :* Pourquoi les implémentations récursives sont-elles faciles ?

## Bonus

### Bonus 1

Les files à priorité implémentées sur base de tas offrent de bonnes performances mais supposent une capacité bornée *a priori*. On souhaiterait dépasser cette limitation en utilisant un tas extensible.

- (a) Proposer le pseudo-code correspondant.
- (b) A-t-on conservé les performances de l'implémentation à base de tas ?  
Implémenter une file à priorité se basant sur un tas-max extensible.

### Bonus 2

Un tas d'arité  $d$  ( $d$ -ary heap), est similaire à un tas binaire mais où chaque noeud (interne) peut posséder jusqu'à  $d$  fils.

- (a) Comment représenter un tel tas dans un tableau ?
- (b) Dessiner l'arbre ternaire correspondant au tableau suivant :  
[25, 12, 24, 15, 8, 7, 4, 13, 6, 3, 5, 9, 10, 1, 2]. S'agit-il d'un tas-max ?
- (c) Quelle est la hauteur d'un tas d'arité  $d$  de  $n$  éléments en fonction de  $n$  et de  $d$  ?