TD2: Structures de données abstraites

2025-2026

On rappelle qu'on dispose de 4 opérations pour manipuler les piles, les fils, les files de priorité, à savoir : création, test de vacuité, insertion, extraction. Par exemple, pour les piles le format de ces opérations est le suivant :

- `creer_pile(): Pile` : créé et retourne une pile vide ;
- `pile_vide(p: Pile): booleen` : teste si une pile est vide ;
- `insere_pile(p: Pile, x: entier)` : insère l'entier x dans la pile ; la pile est modifiée par effet de bord ;
- `extrait_pile(p: Pile): entier` : retire l'entier "sur le dessus de la pile" et retourne sa valeur ; la pile est modifiée par effet de bord ; la pile en entrée ne doit pas être vide ;

Exercice 1. Manipulations diverses

Pour traiter les questions suivantes, vous pouvez utiliser des variables locales de type `Pile` ou `File`.

a. Écrire un algorithme qui inverse l'ordre des éléments d'une pile.

b. Écrire un algorithme qui supprime un élément sur deux d'une pile. L'ordre des éléments doit être conservé.

c. Écrire un algorithme qui, étant donnée une pile, retourne deux piles qui contiennent respectivement tous les entiers pairs et tous les entiers impairs.

d. Même question mais en n'utilisant qu'une seule pile supplémentaire. Combien d'appels à la fonction `insere_pile` faites vous dans le pire cas sur une pile de taille n ? Comparer par rapport à la solution de la question précédente.

e. Même question mais en utilisant une file à la place de la pile supplémentaire.

Exercice 2. Tri par file

Soit l'algorithme `fusion(f1: File, f2: File): File` qui prend en argument deux files triées et renvoie la fusion (triée) de ces deux files.

a. Écrire l'algorithme `fusion` en pseudo-langage. Quelle est sa complexité ?

On va utiliser cet algorithme afin de trier n objets de la manière suivante :

- initialement, chaque objet est placé seul dans une file, et ces n files sont elles-mêmes dans une file notée f ; donc f est une file contenant n files ;
- tant que f contient au moins deux files, on extrait 2 files de f, on les fusionne, puis on insère le résultat dans f.
- quand la boucle s'arrête, f ne contient plus qu'une seule file, dans laquelle les n objets sont triés.

b. Donner les étapes de l'algorithme pour trier les 8 caractères suivants : C, A, R, I, B, O, U, S.

c. Écrire cet algorithme de tri en pseudo-langage. Quelle est sa complexité ?

d. Quel type d'algorithme obtient-on en remplaçant la file de files par une pile de files ?

Exercice 3. Implémentation des files de priorité par un tas binaire

Un *tas binaire* est une structure de données généralement représentée par un arbre binaire presque complet dans lequel les sommets ont des valeurs qui satisfont la propriété suivante : la valeur d'un sommet est supérieure (dans le cas du tas-max que nous utiliserons ici) à la valeur de ses enfants.

Un arbre binaire presque complet de hauteur h est un arbre qui vérifie les deux conditions suivantes :

— l'arbre est complet jusqu'à la profondeur $h-1$;

— les feuilles de profondeur h sont "tassées à gauche".

a. Dessiner des arbres binaires presque complets de taille 7, 9, et 14.

b. Quel est le nombre de feuilles d'un arbre binaire presque complet de taille n ?

Un tas peut être écrit dans un tableau : la racine de l'arbre est stockée à l'indice 0, puis le sommet stocké à l'indice i a son fils gauche à l'indice $2i+1$ (si $\leq t.n-1$) et son fils droit à l'indice $2i+2$ (si $\leq t.n-1$). Dans cette représentation, un tableau t de taille n est un tas si il satisfait la propriété suivante pour tout indice $i \leq n-1$:

$$\begin{cases} t[i] \geq t[2i+1] & \text{si } 2i+1 \leq n-1 \\ t[i] \geq t[2i+2] & \text{si } 2i+2 \leq n-1 \end{cases}$$

c. Représenter les tableaux suivants sous forme d'arbre binaire. Quels sont ceux qui satisfont la propriété du tas ?

indices	0	1	2	3	4	5	6	7	8
t_1	16	14	10	8	7	9	3	2	4
t_2	11	6	10	8	7	5	3	2	4
t_3	18	16	15	13	14	1	2	12	11

d. Implémenter les opérations du type file de priorité en utilisant un tas. Donner la complexité de ces opérations.