Implémentations fonctionnelles de files

Exercices de TD

1 Structure de file

Une file est une structure de données modélisant un contenant d'objets et dans lequel les objets sont retirés dans le même ordre que celui dans lequel on les a insérés (on parle de *FIFO* pour *First In, First Out*).

Une telle structure doit offrir les 4 opérations suivantes :

```
file_vide : 'a file
est_file_vide : 'a file → bool
enfile : 'a → 'a file → 'a file
defile : 'a file → ('a * 'a file)
```

Exercice 1 *Une première idée est de représenter une file à l'aide d'une liste Caml, dans laquelle les éléments sont enfilés par la gauche et « défilés » par la droite.*

1. *Sans détailler leur implémentation, quelles seront les opérations efficaces et les opérations lentes ? (Préciser la complexité de chacune.)*
2. *Que se passe-t-il si on préfère enfiler les éléments par la droite et les défiler par la gauche ?*

Exercice 2 *Pour améliorer l'implémentation précédente, on procède de la façon suivante :*
- *Une file est représentée par* deux *listes, une d'entrée et une de sortie.*
- *Les éléments sont toujours enfilés en tête de la liste d'entrée et défilés depuis la tête de la liste de sortie.*
- *On transfère les éléments de la liste d'entrée vers la liste de sortie lorsqu'on essaye de défiler mais que cette dernière liste est vide,* ***et seulement dans ce cas****.*

1. *Représenter le comportement de cette file par des schémas*
2. *Expliquer en quoi cette représentation est plus efficace : quelle est la complexité de chacune des opérations (encore une fois sans détailler leur implémentation) ?*

2 File à Priorités

Une File à Priorités (FàP) est une structure de données similaire à la file, mais dans laquelle chaque élément est pourvu d'une *priorité* (par exemple, un entier), qui définit dans quel ordre les éléments seront extraits de la FàP. La priorité d'un élément est spécifiée lors de son insertion, et lors d'une extraction on choisit l'élément ayant la priorité la plus haute.

Les opérations sont similaires à celles de la file, mais il faut préciser la priorité d'un élément lors de son insertion :

```
fap_vide: 'a fap
est_fap_vide: 'a fap → bool
insere: 'a → int → 'a fap → 'a fap
extrait: 'a fap → ('a * 'a fap)
```

Exercice 3 *Si on représente une FàP à l'aide d'une liste Caml de couples* (élément, priorité) *ordonnée par priorité décroissante, quelle sera la complexité de chacune des 4 opérations ?*

Exercice 4 *Connaissez-vous une structure de données fonctionnelle qui permette d'améliorer ces coûts ? Quel sera le gain ? Est-ce que toutes les opérations sont améliorées ?*

Exercice 5 *On propose ici d'utiliser un Arbre (binaire) partiellement Ordonné : chaque nœud porte un élément de priorité supérieure à celle de ses deux fils.*

1. *Schématiser un exemple d'un tel arbre contenant les priorités* $\{5, 2, 12, 7, 4, 10\}$.
2. *Où est situé dans l'arbre l'élément de priorité maximale ?*
3. *On souhaite insérer un nouvel élément :*
   — *si sa priorité est supérieure à celle de la racine, où faut-il l'insérer ?*
   — *et si elle est inférieure, comment faire ?*
   Illustrer les différents cas par des schémas.
4. *On souhaite maintenant extraire l'élément de priorité maximale :*
   — *Par quel autre élément doit-on le remplacer ?*
   — *Que reste-t-il à faire ensuite ?*
5. *Quelle est la complexité des opérations ci-dessus ?*
6. *L'arbre obtenu après plusieurs insertions et suppressions sera-t-il équilibré en général ?*
   Sans complètement chercher à résoudre ce problème, peut-on trouver une solution simple pour le mitiger ?

Exercice 6 *Encore mieux, on peut utiliser un Tas, c'est-à-dire un arbre partiellement ordonné* tassé, *autrement dit dont tous les niveaux sont complets sauf le dernier, et ce dernier doit avoir toutes ses feuilles à gauche.*

Comme tous les nœuds internes sont complets, un tel arbre peut alors se représenter sans ambiguïté par un tableau `('a * int) array` *qui contient le parcours en largeur du tas.*

1. *Quel est l'indice de la racine dans le tableau ?*
2. *Si un nœud est à l'indice n, à quels indices sont son fils gauche et son fils droit ? et son père ?*
3. *Comment déterminer si un nœud est une feuille ou un nœud interne ?*
4. *Pour conserver l'aspect « tassé », l'opération d'insertion doit être faite un peu différement de l'exercice précédent. Il est suggéré de partir d'une feuille et de remonter vers la racine.*
   Illustrer par des schémas.
5. *De même, l'opération d'extraction doit conserver l'aspect tassé de l'arbre. La stratégie conseillée est cette fois-ci de remplacer l'élément de priorité maximale par un autre élément, et de faire descendre celui-ci dans l'arbre autant que nécessaire.*
6. *Quel est alors l'inconvénient de ce tableau par rapport à une représentation purement fonctionnelle ?*