INF4100
Laboratoire no. 9

02/04/08

(Files de priorité et algorithmes voraces)

1. Algorithme vorace de tri

Concevez un algorithme de tri qui utilise *une approche vorace*, c'est-à-dire, qui correspond le plus possible précisément possible à une instance de l'algorithme glouton générique présené à l'Algorithme 2 (p. 85) du cahier de notes de cours.

Utilisez l'interface suivante, où on suppose que le `Sac` et la `Sequence` contiennent des valeurs, comparables, de même type — `ok` n'est pas nécessaire, puisqu'on peut toujours trouver une solution :

```
PROCEDURE trierVorace( C: Sac ) S: Sequence
```

À quel algorithme classique de tri cet algorithme vorace correspond-il?

2. Files de priorité réalisées avec monceau

Que contiendra le tableau `elements` pour la file de priorité `fp` réalisée avec un monceau après les appels suivants :

a.
```
cap FilePriorite fp = create FilePriorite( 8 );
fp.inserer( 15, 15 );
fp.inserer( 18, 18 );
fp.inserer(  1,  1 );
fp.inserer( 12, 12 );
fp.inserer(  6,  6 );
fp.inserer( 11, 11 );
int p;
fp.retirerMin( p, p );
```

b.
```
int prios[6] = ( 15, 18, 1, 12, 6, 11 );
cap FilePriorite fp = create FilePriorite( 8 );
fp.insererN( prios, prios, 6 );
int p;
fp.retirerMin( p, p );
```

3. Files de priorité

On veut ajouter, au type `FilePriorite` (fichier `fp.mpd`), une opération `maxPrio` ayant l'interface suivante :

```
op maxPrio() returns Priorite prio;
# PRECONDITION
#   ~estVide()
# POSTCONDITION
#   (prio, elem) IN fp & prio = MAXIMUM( p :: (p, e) IN fp )
```

En d'autres mots, l'opération permet de trouver la (pas nécessairement unique) priorité *est maximum*, et ce sans retirer l'élément associé de la file de priorité.

Pour chacune des mises en oeuvre vues en cours — séquence non-ordonnée d'items, séquence ordonnée d'items et (min-)monceau :

a. Donnez le code MPD mettant en oeuvre `maxPrio`, et ce de la façon la plus efficace possible (tant de façon pratique qu'asymptotique).

b. Quelle serait la complexité asymptotique de `maxPrio`? Plus exactement, combien d'éléments serait-il nécessaire d'examiner dans le tableau `elements`? Expliquez birèvement.

4. Analyse de `insererN`

Soit un arbre binaire comportant k niveaux — la racine compte aussi pour un (1) niveau — et où tous les niveaux sont pleins, donc tel que le nombre de noeuds $n = 2^k - 1$

Soit la propriété suivante, valide pour $-1 < r < 1$:

$$\sum_{i=1}^{+\infty} i \cdot r^i = \frac{r}{(1-r)^2}$$

En utilisant cette propriété, montrez que l'expression suivante, qui découle de l'analyse asymptotique de l'opération `insererN` du type `FilePriorite`, est dominée par $O(n)$:

$$T(n) = \sum_{i=1}^{k-1} i \cdot 2^{k-i-1}$$

Exercices tirés des notes de cours :

- Série 5 (p. 226, 229) : 1, 2, 6

Exercices tirés des notes de cours :

- Série 5 (p. 230) : Exercices traduits du manuel de Neapolitan & Naimipour (Chap. 4) : 39