# IFT2010 Hiver 2006 — Devoir 3

Miklós Csűrös

16 mars 2006

À remettre lors du cours du 30 mars.

## 1 Les disques (100 points)

Supposons qu'on se donne un ensemble de $n$ fichiers, où la taille $t_i$ du $i$ème fichier vérifie $0 < t_i \leq M$ avec un seuil $M$ quelconque. On souhaite sauvegarder les fichiers sur un minimum de disques de taille $M$. Chaque disque peut contenir un sous-ensemble de fichiers dont la taille globale n'excède pas $M$. Imaginez par exemple que vous voulez créer un archive de tous vos fichiers MP3 sur des CDs ($M = 700$kb).

Le problème est difficile (NP-ardu) mais il existe beaucoup d'heuristiques qui mènent à des solutions acceptables. Vous devez implanter l'heuristique *worst-fit*. Cette heuristique prend chaque fichier l'un après l'autre et le place sur un disque qui peut l'acceuillir : soit un nouveau disque si aucun des disques utilisés n'ont assez d'espace libre, soit sur le disque qui a le *maximum* d'espace libre. Par exemple, si on a des fichiers de taille 500, 550, 150, 100, 100 et $M = 700$ l'algorithme les place sur trois disques : $\{500, 150\}$, $\{550, 100\}$ et $\{100\}$. (Notez que l'on peut les sauvegarder sur deux disques plutôt.)

Votre implantation utilisera une file de priorité. Vous pouvez utiliser la code de Mark Allen Weiss (http ://www.cs.fiu.edu/~weiss/dsaajava/code/DataStructures/BinaryHeap.java).

Implantez la classe `Disque` qui représente un disque de taille $M$ : $M$ est un argument du constructuer ou $M = 700000$ par défaut.

```
public Disque(int M) { /* votre code ... */};
public Disque(){ this(700000);}
```

La classe `Disque` mantient la liste des tailles des fichiers y sauvegardés. Elle implémente l'interface `Comparable` pour qu'on puisse la placer sur le monceau : c'est par cette fonctionnalité que vous implantez l'heuristique *worst fit*.

Votre logiciel doit lire une liste de nombre entiers positifs de `stdin` (c'est la liste de tailles), et afficher les disques ($M = 700000$) utilisés pour sauvegarder les fichiers. La sortie donne l'information suivante :

- taille totale de fichiers, divisée par $M$ — c'est une borne inférieure sur le nombre de disques nécesssaires
- nombre de disques utilisés
- s'il y a moins de 100 fichiers, la liste de disques dans l'ordre décroissant d'espace libre : pour chaque disque, on doit afficher le nombre de fichiers, l'espace libre, et la taille des fichiers dans l'ordre d'insertion.

Exemple :

```
% java archiveDisques < fichiers20.txt
taille totale : 7.8
nombre de disques : 9
2 102000 : 98000 500000
1 12000 : 588000
...
```

Détails sur des tests à performer seront affichés sur le site du cours.