TD – 02

Exercices d'applications

Savoirs et compétences :

- ❒ *Alg – C17 : tris d'un tableau à une dimension de valeurs numériques (tri par insertion, tri rapide, tri fusion).*

Exercice 1 – Représentation du coût temporel des tris

Objectif Représenter pour chacun des tris les courbes indiquant le temps d'exécution en fonction du nombre d'éléments à trier.

On donne la bibliothèque de tri `tris.py` dans laquelle différents tris ont été implémentés. On dispose ainsi des fonctions :

- `tri_insertion`;
- `tri_rapide`;
- `tri_fusion`.

On dispose aussi de la méthode `sort` disponible en Python.

On utilisera de plus le module `time()` de la bibliothèque `time` pour créer un chronomètre et le module `randint` de la bibliothèque `random`.

Question 1 *Tracer, dans chacun des 4 cas, le temps de tri d'une liste en fonction du nombre d'éléments de la liste. Le nombre d'éléments variera de 0 à 2 000 000. Une liste de n éléments sera composée de nombres choisis aléatoirement entre 0 et n. Ce réseau de courbes représentera le cas moyen.*

Question 2 *Conclure sur l'efficacité algorithme de chacun des tris dans le cas moyen.*

Exercice 2 – Classement de l'étape Tarbes – La Pierre-Saint-Martin – 167 km

Les coureurs du tour de France sont en train de terminer la seizième étape du Tour de France qui sépare Tarbes et La Pierre-Saint-Martin.

Le fichier `classement_général` rassemble le classement général à l'issue de l'étape 9. Le fichier `etape_16` contient le classement de l'étape 10 uniquement.

Objectif L'objectif est de réaliser le classement général après la seizième étape.

Lecture des fichiers de résultat

Question 1 *Réaliser la fonction* `charge_classement` *permettant de lire un fichier de classement et de retourner une liste de la forme* `[[Nom_1, Dossard_1, Temps_1], [Nom_2, Dossard_2, Temps_2], ...]`. *Le temps devra être exprimé en secondes.*

Classement en fin d'étape

Dans une première approche, on souhaite réaliser le classement général après la fin de l'étape.

Question 2 *Réaliser la fonction permettant d'ajouter les temps de l'étape 10 aux temps du classement général.*

Question 3 *Quel méthode de tri vous semble la mieux adaptée au tri du classement général ?*

Question 4 *Modifier les algorithmes de tris pour pouvoir trier la liste donnée suivant le temps de course d'un coureur. Le classement général a-t-il changé à l'issue de la seizième étape ?*

Classement en cours d'étape – Implémentation d'une file

On cherche à reconstituer le classement général au fur et à mesure que les coureurs arrivent. Dans cette partie le classement de l'étape (liste de liste) sera vu comme une **file** FIFO (First In First Out) ou le premier élément est le premier coureur arrivé et le dernier élément est le dernier coureur à avoir passé la ligne d'arrivée.

Question 5 *Implémenter les fonctions élémentaires liées à la gestion des files :* `enfiler`, `defiler`, `est_vide`. *À l'intérieur de ces fonctions, on s'autorise les méthodes liées au listes (*`append`, `pop`, *...).*

Question 6 *Implémenter la fonction* `ajout` *ayant pour but d'ajouter le temps de l'étape d'un coureur dans le classement général et de mettre à jour ce classement. La gestion du classement de l'étape devra être réalisé grâce à une liste.*

Classement en cours d'étape – arbre binaire

Dans cette partie, on se propose d'étudier une nouvelle structure de donnée : les arbres binaires. L'objectif sera de représenter le classement non plus sous forme d'une liste ordonnée, mais sous la forme d'un arbre.

Présentation des arbres

Un tas binaire est une structure de données informatiques qui permet d'accéder au maximum (respectivement minimum) d'un ensemble de données en temps constant. On peut la représenter par un arbre binaire vérifiant deux contraintes :

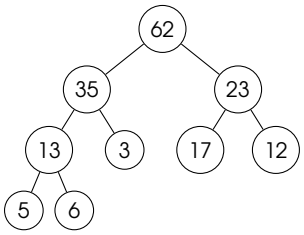
- c'est un arbre binaire parfait : tous les niveaux de l'arbre (excepté le niveau le plus bas) sont totalement remplis. Si le dernier n'est pas totalement rempli alors il doit l'être de gauche à droite ;
- c'est un tas : une clé est associée à chaque nœud de l'arbre. Cette dernière doit être supérieure ou égale aux clés de chacun de ses fils.

Ainsi, lorsque les clés sont des nombres (valeurs) et quand la relation d'ordre choisie est l'ordre naturel, on parle alors de tas-max (ou max-heap).

Un tas binaire étant un arbre binaire complet, on peut l'implémenter à l'aide d'un tableau tel que :

- la racine de l'arbre (niveau le plus haut) se trouve à l'index absolu 1 ;
- en considérant un nœud à l'index absolu i :
  - son fils gauche se trouve à l'index absolu $2i$;
  - son fils droit se trouve à l'index absolu $2i+1$;
- en considérant un nœud à l'index absolu $i > 1$:
  - son père se trouve à l'index absolu $i/2$, le symbole / désignant ici la division entière.

La figure suivante illustre un arbre binaire (trié) et le tableau qu'on peut lui associer avec les index associés (index absolu, index en langage Python).



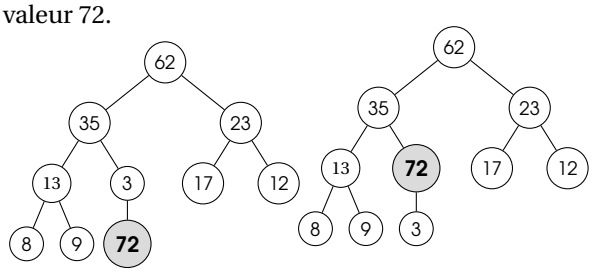
Valeur dans l'arbre (clé)	62	35	23	13	3	17	12	5	6
Index absolu dans l'arbre	1	2	3	4	5	6	7	8	9

Le nœud ayant la valeur 62 est la racine de l'arbre (niveau le plus haut). Les nœuds ayant les clefs 5 et 6 sont au niveau le plus bas.

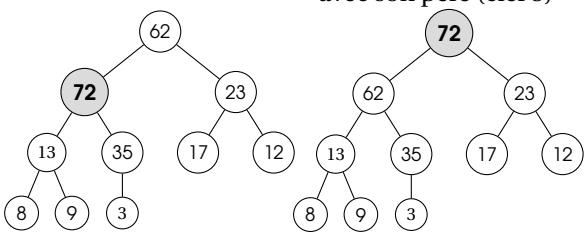
Construction de l'arbre

On souhaite construire l'arbre représentatif du classement des coureurs. Les valeurs de l'arbre seront le temps de course.

Dans un tas binaire, on insère l'élément à la prochaine position libre (la position libre la plus à gauche possible sur le dernier niveau) puis on effectue l'opération suivante (nommée `percolateUp`) pour rétablir si nécessaire la propriété d'ordre du tas binaire : tant que l'élément n'est pas la racine de l'arbre et que la clé de l'élément est strictement supérieure à son père on échange les positions entre l'élément et son père. L'exemple de la figure suivante montre l'insertion d'un élément ayant une clé de valeur 72.



Insertion de la clef 72 — Échange de la clef 72 avec son père (clef 3)



Échange de la clef 72 avec son père (clef 35) — Échange de la clef 72 avec son père (clef 62)

Question 7 *En utilisant la fonction* `percolateUp` *construire l'arbre correspondant à l'arbre présenté dans l'exemple.*

Question 8 *Modifier la fonction* `percolateUp` *pour pouvoir réorganiser le classement général.*

Extraction du premier du classement

On souhaite extraire le premier du classement de l'arbre. Pour cela, il faut retirer la racine de notre tas binaire (c'est-à-dire le maximum de notre tas). Cependant il faut pouvoir conserver la structure de tas binaire après cette suppression. Pour cela, on procède donc de la manière suivante :

- on supprime la racine et on met à sa place l'élément qui était en dernière position de l'arbre (c'est-à-dire le plus à droite sur le dernier niveau) ;
- puis on exécute l'opération nommée `percolateDown` dont l'implémentation est donnée dans le fichier ressource.

Question 9 *On cherche à analyser les évolutions de la fonction* `percolateDown`*. Appliquer l'opération d'extraction sur l'exemple du tas binaire modélisé dans le paragraphe précédent. On donnera la liste de départ, un schéma du tas binaire à chaque étape ainsi que la liste finale.*

Question 10 *Modifier la fonction* `percolateDown` *pour qu'elle puisse s'appliquer au classement général.*