# TP2 – IFT1025

**TP à faire en groupe de 2 personnes**

*TP à faire en groupe de deux.*
*Ce devoir doit être rendu sous la forme d'un projet java zippé (obtenu grâce à export dans Eclipse), nommé de la façon suivante :*
*Nom1I1_Nom2I_TP1.zip*
*Avec Nom1 le nom du premier étudiant, I1 la première lettre du prénom du premier étudiant, Nom2 le nom du second étudiant et I2 la première lettre du prénom du second étudiant.*

*A rendre sur Studium pour le : 7/12/2020, 23h59*

*Une remise en retard sera pénalisée de 10% par jour.*

## Buts

Ce TP est un exercice pour pratiquer les notions et les techniques enseignées dans le cours jusqu'à présent, à savoir : l'organisation des classes, l'utilisation de listes, le tri, le fichier, et l'interface graphique.

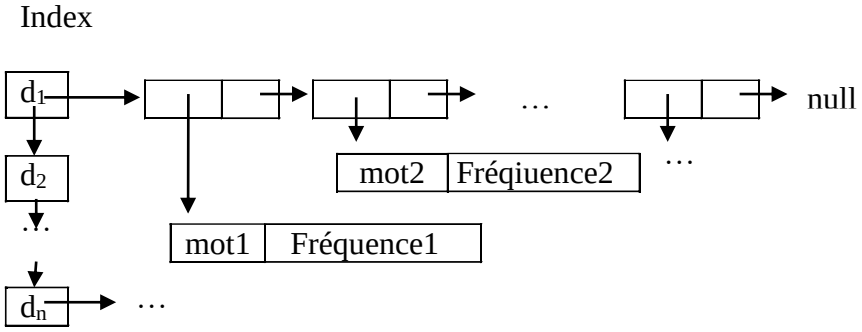
## Problème

On veut construire un système de recherche d'information (engin de recherche) simplifié. Ce système doit indexer un ensemble de textes (documents), construire une structure d'index et l'utiliser pour retrouver les textes correspondant à une requête de l'utilisateur. Ce sont les fonctions de base typiquement utilisées dans un engin de recherche.
L'engin de recherche doit faire deux traitements : Indexation des documents en mode hors-ligne (offline) et traiter une requête de l'utilisateur en ligne. Ces traitements sont expliqués en bas.

## 1. Fonction d'indexation

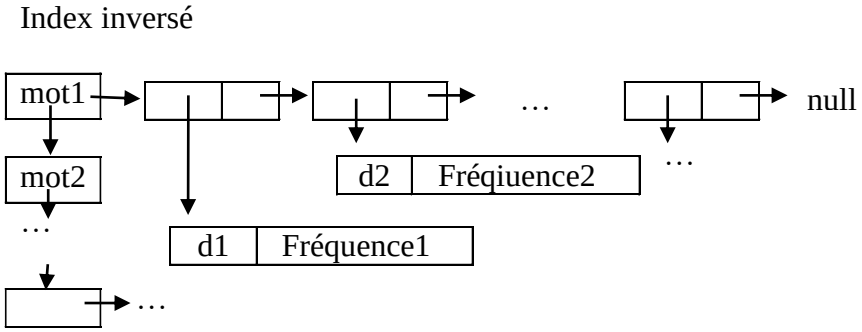
Structure d'index de documents : Pour chaque texte (document sans structure), on doit créer une structure pour stocker tous les mots et leurs fréquences. Pour ce TP, on utilise la structure de liste pour stocker les mots et leurs fréquences. Cette partie de structure est illustrée dans la figure en bas par la liste horizontale, où pour chaque document $d_i$, il y a une liste de nœuds enchaînés, et chaque nœud contient un objet qui stocke un mot et sa fréquence dans le document. Vous pouvez choisir à utiliser une classe existante en Java comme ArrayList ou LinkedList.

L'index doit stocker ces informations pour tous les documents. Ainsi, pour un ensemble de documents, on utilise une autre liste pour les enchaîner, illustrée comme la liste verticale dans la figure : Chaque document est identifié (ID) par son nom de fichier (d1, d2, …, dans l'illustration).




Avec cette structure construite, on peut connaître les mots qui apparaissent dans chaque document, ainsi que leurs fréquences respectives.

Structure d'index inversé : En plus de la structure d'index des documents en haut, on a aussi besoin d'une structure d'index inversé pour accélérer la recherche. Cette structure d'index inversé a une structure similaire. Dans cette structure d'index inversé, on stocke, pour chaque mot, la liste de documents contenant ce mot, avec sa fréquence respective dans chaque document. Ainsi, chaque nœud contient le document et sa fréquence. Dans le sens vertical, nous avons une liste de mots, et chaque mot est lié à une liste de documents et fréquences.
Cette structure est illustrée comme suit :




Processus d'indexation : On montre ici comment les structures d'index sont créées.
Pour indexer un document, on doit spécifier un fichier de texte à indexer (ou un ensemble de fichiers dans un répertoire). On ne considère pas la structure de document dans ce TP, un document est simplement un texte : une suite de mots, séparés par des espaces, des ponctuations habituelles, et des changements de ligne. Un programme d'indexation doit faire les tâches suivantes pour chaque document :

- Créer un nouveau nœud dans la liste de documents dans la première structure d'index (liste verticale) pour stocker l'identité du document;
- Parcourir le texte, et en même temps, faire les test étapes suivantes :
    - reconnaître les mots dans le texte (en faisant une *tokenisation*);

- Insérer chaque mot reconnu dans la liste de nœuds du document (liste horizontale dans la première structure), avec la fréquence correspondante.
  - Si c'est la première fois que le mot est rencontré dans le document, alors on crée un nœud contenant le mot et la fréquence 1;
  - Si le mot a été déjà inséré, alors on augmente la fréquence de 1.

Note : Pour la tokenisation, vous pouvez utiliser split ou regex sur un String. StringTokenizer étant obsolète, il est déconseillé de l'utiliser.

À la fin de cette étape d'indexation, la structure d'index (première structure) sera complétée. On crée ensuite la deuxième structure d'index à partir de la première. Il s'agit de parcourir la première structure, et crée la deuxième structure au fur et à mesure. Le processus de cette création se fera de la manière similaire, au fur et à mesure du parcours de la première structure d'index.

Dans la première structure, une fois complétée, la liste de mots (horizontale) doit être triée. Ceci facilite la conversion en la deuxième structure (inversée). Vous devez réfléchir comment réaliser ceci de façon efficace. Vous avez notamment le choix entre : garder la liste triée tout au long, ou trier la liste à la fin.
(Note : il serait possible qu'on implémente ceci avec la structure de Map, mais on n'utilise pas cette structure. Vous devez utiliser seulement une liste dans ce TP.)

Dans la deuxième structure d'index (inversé), chaque liste de documents (horizontale) et la liste de mots (verticale) doivent rester triées afin de faciliter la recherche pour une requête (voir la suite). Pensez à une façon d'implémenter votre conversion pour respecter ceci.

# 2. La fonction d'affichage

Une fois les structures d'index construites, on peut demander à afficher le contenu de chaque structure.
L'affichage de l'index de documents doit afficher les mots et leurs fréquences pour chaque document indexé. L'affichage de l'index inversé affiche la liste de documents et la fréquence qu'ils contiennent pour chaque mot.
Vous pouvez choisir un format d'affichage raisonnable, mais clair. Cet affichage vise essentiellement à vous aider à régler vos programmes (voir ce que vos structures d'index contiennent). Testez cette fonction avec quelques documents courts d'abord.
(Il est aussi possible que le texte original soit aussi affiché en même temps. Pour cela, il faut stocker le texte (dans une autre structure), et l'afficher. Vous n'êtes pas demandés à faire cela.)

# 3. La fonction de recherche

L'utilisateur peut entrer une requête qui contient un ou plusieurs mots dans une fenêtre de requête (comme sur Google). Le système doit alors déterminer tous les documents qui

contiennent tous les mots de la requête, et les afficher dans l'ordre inversé de leurs scores, c'est-à-dire le plus grand score en premier.

Il y a de multiples façons de calculer ce score. Pour ce TP, le score d'un document pour une requête est l'accumulation de toutes les fréquences des mots de la requête. Par exemple, si une requête contient deux mots A et B, alors :

- il faut que tous les deux mots apparaissent dans un document pourque celui-ci soit pris en compte
- le score sera l'addition des fréquences de ces 2 mots dans le document

Voici quelques détails sur les étapes pour constituer cette liste de réponses finale. Pour une requête contenant des mots :

- Trouver la liste de documents correspondant au premier mot de la requête dans l'index inversé, et initialiser la liste de réponse à cette liste;
- Si cette liste est vide, alors on retourne une liste vide comme réponse finale.
- Sinon, pour chacun des autres mots (s'il y en a) de la requête (qu'on appelle nouveau mot), trouver la liste de documents correspondant dans l'index inversé, et combiner cette liste avec la liste de réponses déjà constituée comme suit :
    - si un document existe à la fois dans la liste de réponses déjà constituée et dans la liste de documents correspondant au nouveau mot, alors ajouter la fréquence du document dans la liste de réponses (accumulation).
    - sinon, si le document est seulement dans la liste de réponses, enlever ce document de la liste de réponse. Si le document est seulement dans la liste du nouveau mot, ne rien faire (ne pas l'ajouter dans la liste de réponses).

  Cette façon de combiner les listes de documents de différents mots exige qu'un document dans la liste de réponse contienne tous les mots de la requête. (Note : Cette étape ressemble à la fusion de deux listes, mais différente.)
  À la fin de cette étape pour un nouveau mot, on peut vérifier si la liste de réponse est vide. Si oui, on retourne tout de suite la liste vide comme réponse sans poursuivre sur les autres mots de la requête.
- Si la liste de réponse finale n'est pas vide, trier cette liste dans l'ordre décroissant de la fréquence globale, de sorte que le document contenant plus d'occurrences des mots de la requête soit classé au début.

Une fois l'opération de recherche terminée, vous affichez la liste de réponses sur l'écran (l'ID des documents et le score correspondant).

# 4. L'interface graphique

Programmer une interface graphique simple pour permettre à lancer les opérations décrites avant. Il vous serait possible de construire 2 interfaces, l'une pour l'indexation (pour l'administrateur de l'engin de recherche), et l'autre pour la recherche (pour l'utilisateur).
Dans la première interface, l'administrateur doit pouvoir lancer l'indexation sur un ou plusieurs documents stockés dans un répertoire. Dans la deuxième interface, l'utilisateur peut entrer une requête et voir la liste de documents trouvés affichée.

Une liste de documents sera affichée sur Studium pour vous servir des exemples.

# Notation

Ce TP correspond à 15 points dans l'évaluation globale. Ces 15 points sont attribués comme suit :

- Fonction de l'indexation (création des 2 structures)
- Fonction de recherche
- Interface graphique
- Style de programme (organisation des méthodes et de classes, lisibilité du code) et commentaires dans le programme

Mettez des commentaires nécessaires dans votre code pour faciliter la lecture, sans toutefois surcharger. Soignez le formatage de votre code pour que ce soit facile à lire. Mettez votre nom dans l'entête de chaque code.