Projet "Quel bazar !"

Vous avez enfin pu mettre la main sur ce livre d'Algorithmique (eh, oui !) que vous cherchiez depuis longtemps. Pas de chance, l'éditeur a raté son coup et toutes les pages sont dans le désordre le plus complet. De plus (décidément, l'éditeur n'est pas doué !), les pages ne sont pas numérotées. Vous n'avez pas le temps, ni l'envie d'attendre un nouvel envoi, alors vous décidez d'agir pour ré-ordonner les pages.

Vous avez besoin d'un coup de pouce ... que votre professeur est prêt à vous donner, mais à sa manière. A partir de sa copie du livre, il a extrait un ensemble de mots clés distincts, qu'il appelle *dictionnaire* et qu'il vous donne avec la précision suivante : si deux pages ont au moins k mots du dictionnaire en commun, alors c'est sûr, les deux pages font partie du même chapitre [1]. Ce critère permet de reconstituer l'ensemble de pages de chaque chapitre, par transitivité (si les pages a et b font partie du même chapitre, et b et c font également partie du même chapitre, alors a et c font partie du même chapitre).

Il ne vous reste plus qu'à utiliser le dictionnaire pour grouper les pages par chapitre, puis d'ordonner les pages de chaque chapitre. Vous décidez d'écrire un programme pour la première tâche, mais de faire la deuxième à la main. Vous scannez donc toutes les pages, puis sauvegardez chaque page sous la forme d'un fichier texte. Avec le dictionnaire donné également sous la forme d'un fichier texte, vous êtes prêt à écrire le programme.

Travail demandé

Vous devez réaliser un programme *aussi efficace que possible* qui prendra en entrée (en tant qu'arguments fournis en ligne de commande) (1) les n fichiers contenant une page chacun [2] ; nous supposerons que chaque page est identifiée par un identifiant entier qui se trouve seul sur la première ligne du fichier contenant la page ; (2) le fichier avec le dictionnaire [3] et (3) la valeur de k ~~(au clavier)~~, et qui fournira en sortie :

- les identifiants des pages, groupés chapitre par chapitre
- pour chaque chapitre, la liste de mots clés propres au chapitre, c'est-à-dire qui se trouvent dans ce chapitre mais ne se trouvent dans aucun autre chapitre.

Votre programme sera écrit en Java, mais n'est autorisé à utiliser que les collections linéaires de type tableau ou liste. Vous créerez vous-même les structures de données arborescentes dont vous aurez besoin pour satisfaire la demande d'efficacité. Par ailleurs, si vous utilisez des méthodes prévues par Java (ex. tri d'un tableau), assurez-vous de bien connaître leur complexité.

Rendu de TP

Les programmes doivent fonctionner parfaitement sur les machines du CIE, sous Linux. Un rapport d'au maximum 10 pages sera fourni, comprennant (entre autres) :

- les commandes de compilation et exécution en mode console
- une vue globale de votre approche, sous la forme d'un algorithme (c'est-à-dire en pseudo-code) dont les instructions sont des appels à des procédures/fonctions ; la spécification et les paramètres de chaque procédure/fonction sont précisément décrits
- pour chaque procédure/fonction :
  * le détail de la procédure/fonction, sous forme algorithmique (c'est-à-dire en pseudo-code) ;
  * l'explication de son fonctionnement ;

1. La valeur du k est connue à l'avance, il vous la donne donc, mais elle n'a pas beaucoup d'importance en soi
2. pour tous ces fichiers, supposer que les mots sont séparés par des blancs, des fins de ligne ou les signes de ponctuation usuels (. , ; : ! ?) mais pas de parenthèse ou autre chose
3. mots séparés par des blancs

    * les raisons pour lesquelles l'algorithme proposé est correct ;
    * sa complexité ;
    * les raisons pour lesquelles vous considérez cet algorithme aussi efficace que possible
- des jeux de données commentés

Important

1. Les fichiers du programme, ainsi que les jeux de données, seront mis dans un répertoire Nom1Nom2 (où Nom1 et Nom2 sont les noms des deux étudiants du binôme). Ce répertoire sera ensuite archivé sous le nom Nom1Nom2.zip. L'archive sera déposée sur Madoc, au plus tard le **vendredi 5 décembre 2014 à minuit** (Madoc n'acceptera pas de retard).
2. La dernière séance est consacrée à une démo de 10 minutes par projet, pour laquelle les fichiers en entrée vous seront fournis (respecter donc scrupuleusement le format indiqué). Cette dernière séance n'est donc pas une séance de travail sur le projet.
3. *Tout* est important pour la notation. En particulier, il sera accordé beaucoup d'attention au respect des consignes et à la recherche d'une complexité minimum - garantie d'une efficacité maximum - pour votre méthode, via la mise en place des structures de données les plus adaptées.