Projet d'algorithmique

Cours d'Informatique de Deuxième Année

—Deug Sciences S3-MI—

Index d'un document

Le but du projet est d'écrire un programme qui établit l'index d'un document et permet la consultation de l'index construit. Une application immédiate serait celle de pouvoir corriger tous les mots mal orthographiés d'un document.

- ce travail doit être rendu au plus tard le mardi 19 mai 1998 aux chargés de TP, la soutenance aura lieu lors de la dernière séance de Travaux Pratiques (le mardi 26 mai 1998) ;
- la clarté et le soin apportés à la rédaction de votre travail, ainsi que la lisibilité de votre *listing* seront pris en compte ;
- ce travail peut être réalisé en binôme, chaque personne devant être présente à la soutenance et pouvant être interrogée individuellement ;
- la partie programmation devra être réalisée en *C ANSI*, compilable directement sous *HP–UX* (l'*UNIX* de *HP*).

---

1 Définition d'un index

Par définition l'index d'un document est formé d'entrées composées d'un mot et des numéros de lignes où ce mot apparaît dans le document (pour les gros document, il s'agit plutôt des numéros de pages). Les entrées sont ordonnées selon l'ordre alphabétique des mots et, pour chaque mot, les numéros de lignes sont en ordre croissant.

2 Construction de l'index

L'objet de cette première partie est la création d'un index à partir d'un document. Pour cela, plusieurs méthodes sont proposées : à vous d'implémenter une de ces méthodes, au choix.

2.1 Utilisation d'une liste chaînée

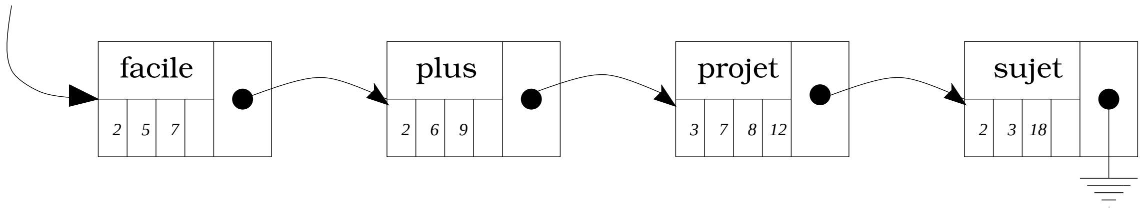
L'index peut être construit en utilisant une structure de liste ordonnée chaînée par pointeurs, où chaque cellule de la liste contient les informations suivantes :

1. un mot du document ;

2. la liste des numéros de lignes où ce mot apparaît ;

3. un pointeur sur la cellule suivante.

Les mots apparaissant dans le document doivent apparaître une et une seule fois dans l'index, indépendamment du fait qu'ils soient en minuscule ou majuscule dans le document. La liste des numéros de lignes peut être implémentée par une simple liste chaînée d'entiers ordonnés par ordre croissant, ou par un tableau ordonné d'entiers, comme dans le dessin suivant :



Dans le cas où la liste d'entiers est implémentée par tableau, il faudra prévoir une réallocation dynamique éventuelle, afin de ne pas limiter le nombre d'entrées dans le tableau. L'implémentation de la liste d'entiers, comme liste chaînée par pointeurs est toutefois conseillée.

2.2 Par "hachage ouvert"

On remarque qu'il serait toutefois plus avantageux, et pas plus difficile, de considérer 26 listes chaînées au lieu d'une seule. Chacune des listes contiendrait les mots du document qui commencent par la même lettre : une liste pour tous les mots du document qui commencent par la lettre `a`, une liste pour tous les mots du document qui commencent par la lettre `b`, etc. Dans ce cas, l'index peut être implémenté comme un tableau de 26 éléments (pour les 26 lettres de l'alphabet). Chaque élément du tableau est une liste chaînée définie comme ci-dessus mais ayant la propriété que tous les mots qu'elle contient commencent par la même lettre. On parle alors d'une implémentation de l'index par "hachage ouvert" (voir le livre "Structures de données et algorithmes", Aho, Hopcroft, Ullman, page 123).

3 Consultation de l'index

Une fois l'index construit, on doit, bien entendu, pouvoir le consulter. Consulter l'index signifie, au minimum :

- pouvoir rechercher, autant de fois qu'on le souhaite un mot quelconque et avoir en sortie tous les numéros de lignes où ce mot apparaît dans le document ; un message opportun signalera le cas où le mot n'apparaît pas dans le document ;
- pouvoir lister l'ensemble des mots de l'index ;
- pouvoir afficher le nombre de mots de l'index ;
- afficher tous les mots apparaissant à une ligne `l` donnée.

4 Description type d'une session

Le programme que vous devez faire doit être interactif, au sens où l'utilisateur doit pouvoir faire des requêtes et obtenir des réponses. On pourra donc prévoir un petit langage de commandes permettant à l'utilisateur de passer ses demandes au programme. Le langage utilisé devra être très simple et pourra s'inspirer de l'exemple suivant.

Au lancement de `a.out` (par exemple), le programme affiche un *prompt*—nous avons choisi le *prompt* `INDEX>`—et se met en attente d'un ordre de la part de l'utilisateur. La première commande utile sera la commande `h` fournissant de l'aide sur les commandes disponibles. L'utilisateur peut désormais demander la création d'un index, par la commande `c` à partir d'un document de nom `fichier.txt`, puis par exemple demander le nombre de mots de cet index par la commande `n`. L'utilisateur pourra alors par exemple demander les lignes où apparaît le mot `facile` en utilisant la commande `r` suivie du mot `facile`.

L'exemple suivant détaille cette session.

```
$ a.out
INDEX> h
```

Commande	*Effet*
`q`	`Pour quitter le programme`
`h`	`Pour obtenir cette aide`
`c nom-de-fichier`	`Création d'un index`
`m`	`Liste tous les mots de l'index`
`n`	`Affiche le nombre de mots de l'index`
`r` mot	`Affiche toutes les lignes où apparait un mot donné`
`l numéro-de-ligne`	`Liste tous les mots apparaissant sur une ligne donnée`

```
INDEX> c fichier.txt
INDEX> n
342647
INDEX> r facile
facile : 2, 5, 7,
INDEX> q
```

5 Travail à rendre

Vous devez rendre :

1. un listing de votre programme (qui devra être judicieusement commenté). Vous n'êtes pas obligés d'implémenter l'amélioration suggérée utilisant un “hachage ouvert” ; toutefois, cela est vivement conseillé. Avec un peu de réflexion, vous pourrez vous rendre compte que cette méthode n'est ni plus longue, ni plus difficile à écrire. Par contre, elle présente l'avantage d'améliorer quelque peu l'exécution de votre programme (est-ce toujours vrai ?) ;

2. un compte rendu du travail effectué, dans lequel vous expliquerez brièvement les algorithmes utilisés ainsi que leur complexité et une courte justification du choix de l'algorithme.

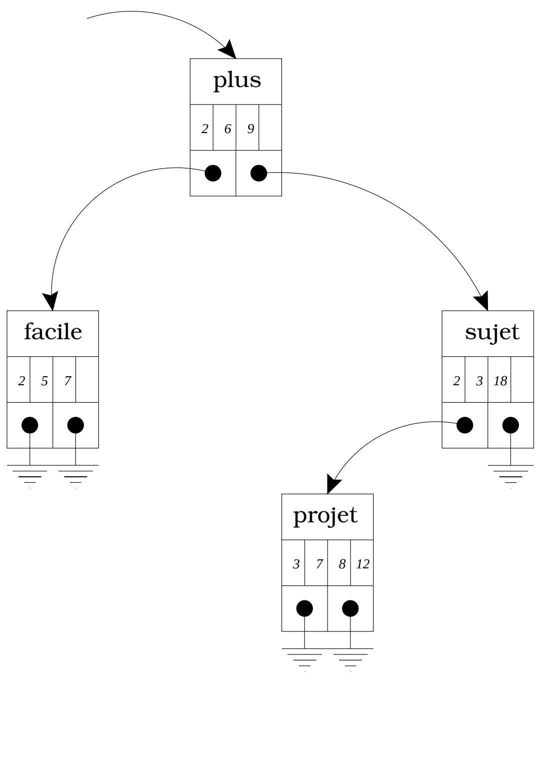
6 Améliorations et options

6.1 Sur la construction

L'index peut être construit en utilisant, au lieu d'une liste chaînée, un arbre binaire de recherche, ou encore une table de hachage avec une fonction de hachage plus fine que celle suggérée précédemment.

6.1.1 Par arbre binaire de recherche

Plutôt que de manipuler une liste chaînée, on peut se servir d'une structure d'*arbre binaire de recherche*. Un arbre binaire est un arbre dont chaque nœud possède au plus deux fils. Un arbre binaire de recherche est un arbre binaire pour lequel étant donné un nœud indexé par un mot w, tous les mots du sous-arbre gauche sont plus petits que w pour l'ordre alphabétique, et tous les mots du sous-arbre droit sont plus grands que w pour ce même ordre. La figure ci-dessous illustre cette structure sur l'exemple de la section 2.1. A noter que le chaînage est encore une fois assuré par des pointeurs (un pointeur sur le fils gauche et un pointeur sur le fils droit). Par ailleurs, nous avons à nouveau choisi de représenter la liste des numéros de lignes par un tableau d'entiers, mais cette solution peut tout aussi bien être remplacée par une représentation par liste chaînée.



6.1.2 Par une fonction de hachage plus fine

Une autre idée consiste à modifier la fonction de hachage dont nous avons parlé précédemment. Cette fonction de hachage prenait en argument un mot, et en regardant la première lettre de ce mot, on pouvait savoir dans quelle liste chaînée notre mot devait être inséré, ou alors, était susceptible de se trouver—dans le cas d'une recherche.

Il y a bien entendu beaucoup d'autres méthodes de hachage qui permettent de réduire les coûts d'accès à un élément de l'index et nous vous renvoyons à nouveau au livre d'Aho, Hopcroft, Ullman, page 122, ou alors à l'ouvrage "Concepts fondamentaux de l'Informatique", Aho, Ullman.

L'implémentation de plusieurs méthodes pourra s'inscrire dans le cadre de la comparaison des performances de ces différentes méthodes. On pourra donc rajouter des commandes supplémentaires au langage afin de faire appel à telle ou telle technique de construction de l'index.

6.2 Sur la consultation

La consultation de l'index peut être enrichie afin de connaître, par exemple, le mot qui apparaît le plus souvent ou le moins souvent dans le document, ou avoir aussi la liste des mots triés selon leur fréquence d'apparition (prévoir peut-être dans ce cas, un champ supplémentaire dans chaque cellule pour mémoriser le nombre d'occurences d'un mot dans le document).

Vous pouvez ajouter toute amélioration ou extension de votre choix, digne d'intérêt dans le cadre de l'index d'un document.

En particulier, nous avons parlé de l'utilisation de votre programme afin de corriger tous les mots mal orthographiés d'un document. Pour cela, on fait d'abord appel à la commande "*ispell*" d'*Unix* qui extraît d'un document tous les mots ne figurant pas dans le dictionnaire (de *ispell*). Ensuite, on consulte l'index du document en question (construit par votre programme) pour connaître les numéros de lignes où ces mots apparaissent.