Bases de données documentaires et distribuées
Cours NFE04
Introduction a la recherche d'information

Auteur : Philippe Rigaux

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

Information Retrieval, définition

Une définition

La **Recherche d'Information** (*Information Retrieval*, IR) consiste à trouver des **documents** peu ou faiblement structurés, dans une grande **collection**, en fonction d'un **besoin d'information**.

Prend peu à peu la prédominance sur les techniques de bases de données.

- Recherche sur le Web. Utilisée quotidiennement par des milliards d'utilisateurs.
- Recherche dans votre boîte mail.
- Recherche sur votre ordinateur (*Spotlight*).
- Recherche dans une base documentaire, publique ou privée.

À bien distinguer d'un recherche type "base de données" : requête structurée, données structurées, réponse « exacte ».

Spécificités de l'évaluation des moteurs de recherche

Contrairement à une base de données (SQL), le résultat dépend de **l'interprétation** d'un besoin.

On ne peut jamais dire qu'un résultat est totalement exact (ou totalement faux).

Deux notions importantes :

- **Faux positifs** : ce sont les documents **non pertinents** inclus dans le résultat ; ils ont été sélectionnés à tort.
- **Faux négatifs** : ce sont les documents **pertinents** qui **ne sont pas** inclus dans le résultat.

En général, chercher à réduire les faux positifs entraîne l'augmentation des faux négatifs, et réciproquement.

Evaluation de la pertinence : précision et rappel

La **précision** mesure la fraction des vrais positifs dans le résultat r.

Si on note $t_p(r)$ et $f_p(r)$ le nombre de vrais et de faux positifs dans r, alors

$$\text{précision} = \frac{t_p(r)}{t_p(r) + f_p(r)} = \frac{t_p(r)}{|r|}$$

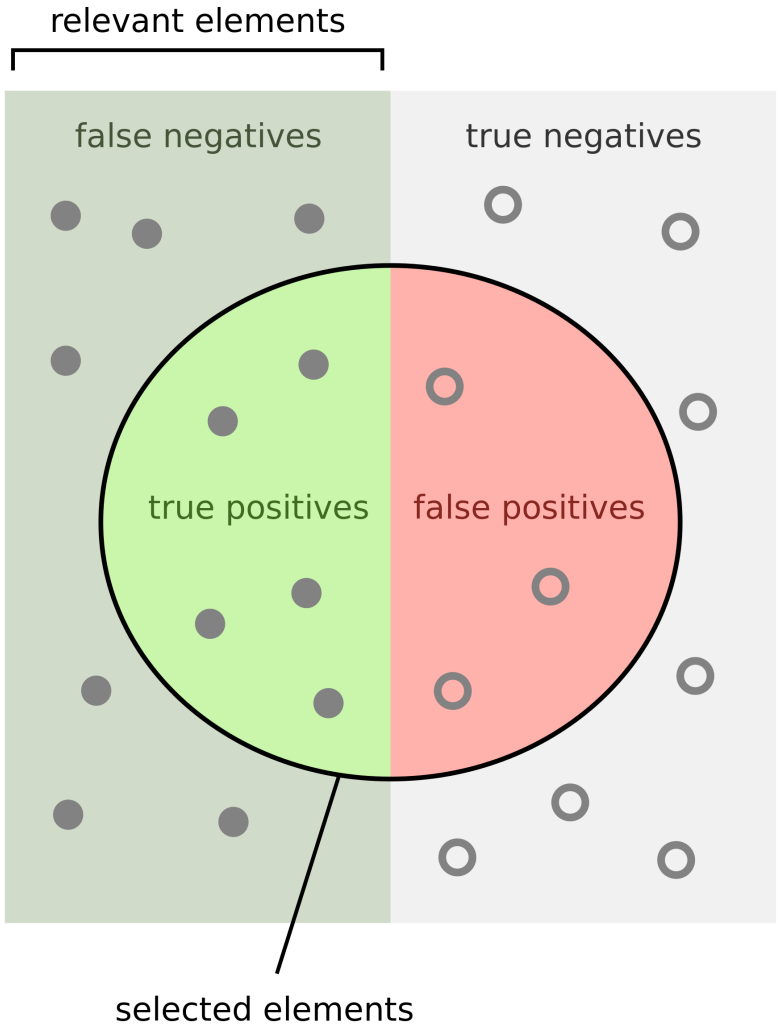
Le **rappel** mesure la fraction de faux négatifs.

Si on note t_p le nombre de vrais positifs **dans la collection** (supposé connu), alors le rappel est :

$$\frac{t_p(r)}{t_p}$$

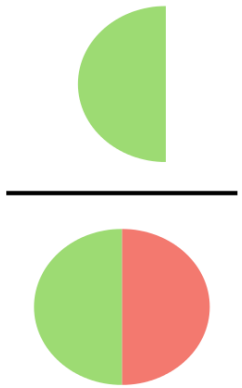
L'évaluation d'un système de RI est difficile : implique des tests rigoureux avec des utilisateurs sur un échantillon.

Evaluation en images

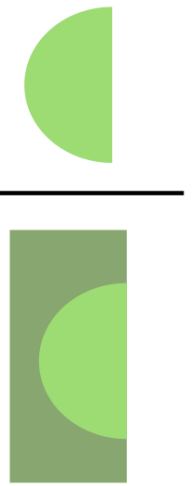


Evaluation en images

How many selected items are relevant?

Precision = 

How many relevant items are selected?

Recall = 

Notre exemple de base

La RI permet d'effectuer des recherches **booléennes** et des recherches **plein texte**. Dans ce dernier cas, il faut **analyser** des documents textuels.

Un ensemble (modeste) nous servira de guide.

d_1	Le loup est dans la bergerie.
d_2	Le loup et le trois petits cochons.
d_3	Les moutons sont dans la bergerie.
d_4	Spider Cochon, Spider Cochon, il peut marcher au plafond.
d_5	Un loup a mangé un mouton, les autres loups sont restés dans la bergerie.
d_6	Il y a trois moutons dans le pré, et un mouton dans la gueule du loup.
d_7	Le cochon est à 12 le Kg, le mouton à 10 E/Kg.
d_8	Les trois petits loups et le grand méchant cochon.

Le besoin et la solution

On veut chercher tous les documents parlant de loups, de moutons mais pas de bergerie (c'est le besoin).

Parcourir tous les documents ?

- potentiellement long ;
- critère "pas de bergerie" n'est pas facile à traiter ;
- autres types de recherche ("le mot 'loup' doit être **près** du mot 'mouton'") sont difficiles ;
- comment **classer** par pertinence les documents trouvés ?

Structure spécialisée : la **matrice d'incidence** et surtout son inversion.

Matrice avec documents en ligne

On sélectionne un ensemble de mots (ou **termes**), constituant notre **vocabulaire** (ou **dictionnaire**).

Documents en ligne, termes en colonnes. Dans chaque cellule : 1 si le terme est dans le document, 0 sinon.

La matrice d'incidence

	loup	mouton	cochon	bergerie	pré	gueule
d_1	1	0	0	1	0	0
d_2	1	0	1	0	0	0
d_3	0	1	0	1	0	0
d_4	0	0	1	0	0	0
d_5	1	1	0	1	0	0
d_6	1	1	0	0	1	1
d_7	0	1	1	0	0	0
d_8	1	0	1	0	0	0

Pour effectuer la recherche

On prend les vecteurs binaires des **termes** (les colonnes).

- Loup : 11001101
- Mouton : 00101110
- Bergerie : 01010011

Puis :

- ET logique sur les vecteurs de Loup et Mouton, on obtient 00001100.
- ET logique avec le **complément** du vecteur de Bergerie (01010111)

On obtient 00000100, d'où on déduit que la réponse est limitée au document d_6.

Opération binaires **très efficace**, mais...

Passons à grande échelle

Quelques hypothèses :

- Un million de documents, mille mots chacun en moyenne.
- Disons 6 octets par mot, soit 6 GO (pas très gros en fait !)
- Disons 500 000 termes **distincts**

$\Rightarrow$ la matrice a 500×10^9 bits, soit 62 GO.

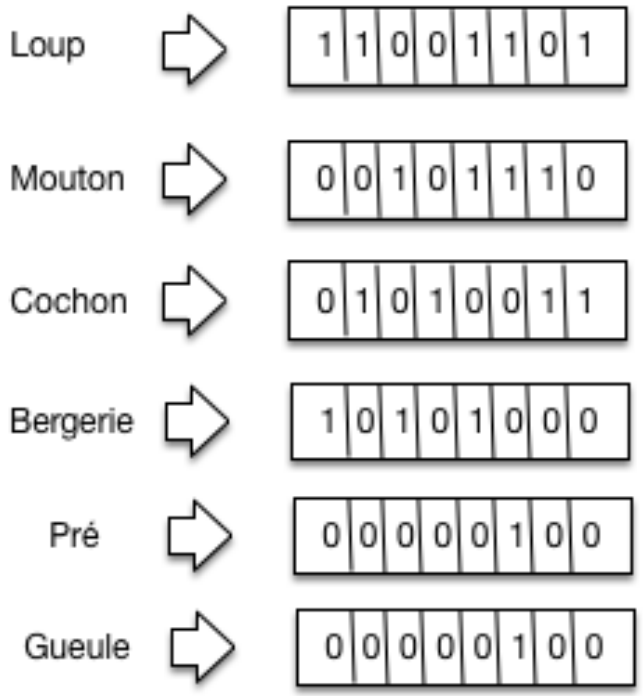
Ne tient pas en mémoire, ce qui va beaucoup compliquer les choses....

Comment faire mieux ?

On peut faire mieux

Il vaut mieux avoir les termes en ligne pour disposer des vecteurs dans une zone mémoire contigue.

On parle de matrice inversée, et de **liste inversée**. Une liste par terme ; dans chaque liste, 1 pour les documents contenant le terme.

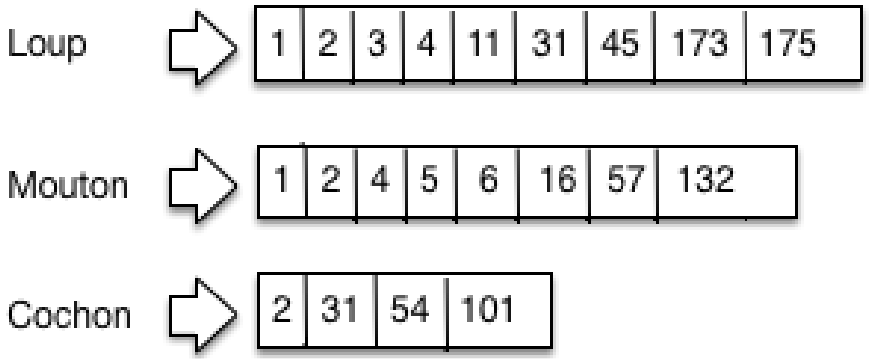


Important

Les mises à jour beaucoup plus difficiles car elles impliquent la réorganisation d'une liste compacte. Prix à payer pour l'efficacité en **lecture** (recherche).

On peut encore faire mieux

La matrice est **creuse** : il n'y a que 10^9 positions avec des 1, soit un sur 500.



Essentiel

On place dans les cellules **l'identifiant** du document. De plus chaque liste est **triée** sur l'identifiant du document.

Index inversé

La structure utilisée dans **tous** les moteurs de recherche.

- Un **répertoire** contient tous les **termes**.
- Une **liste** (inversée) est associée à chaque **terme**, **triée** par docId.
- Chaque élément de la liste est appelé une **entrée**.

Concept : la notion de **terme** (**token** en anglais) est différente de celle de "mot". À revoir.

Vocabulaire : le répertoire est parfois appelé *dictionnaire* ; les listes sont des *posting list* en anglais ; les entrées sont des *postings*.

Efficacité : le répertoire devrait toujours être en mémoire ; les listes, autant que possible en mémoire, sinon fichiers contigus sur le disque.

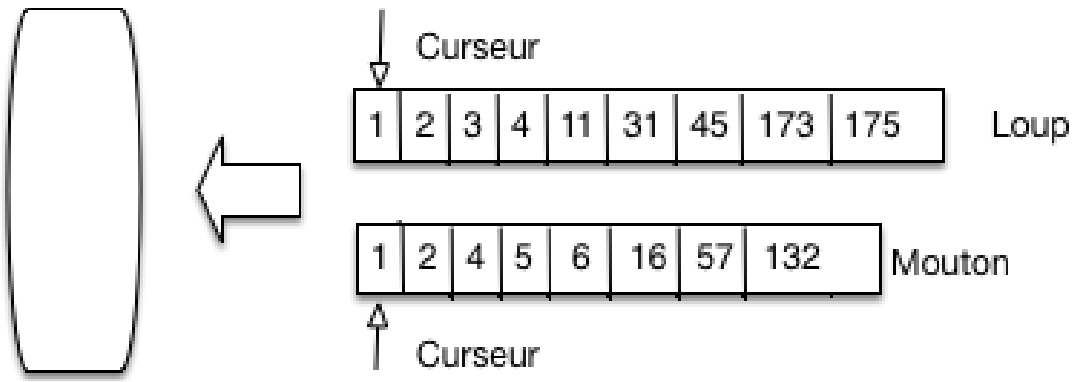
Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.

On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !

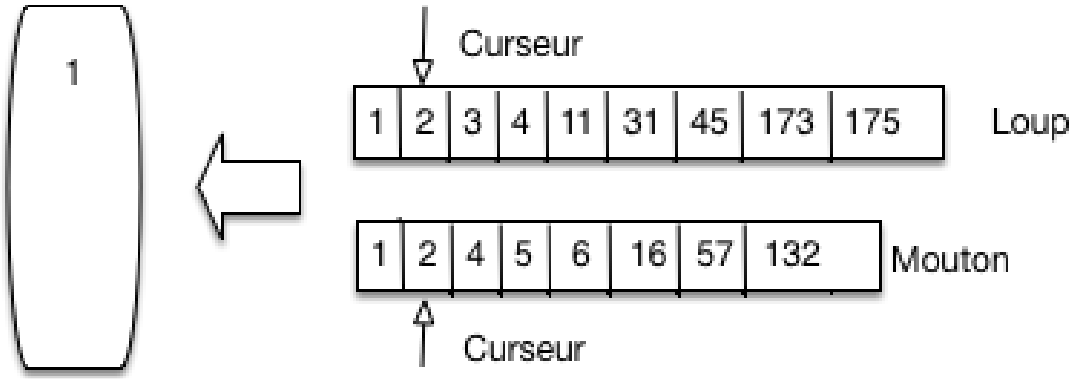
On avance sur la liste du plus petit docId.



Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



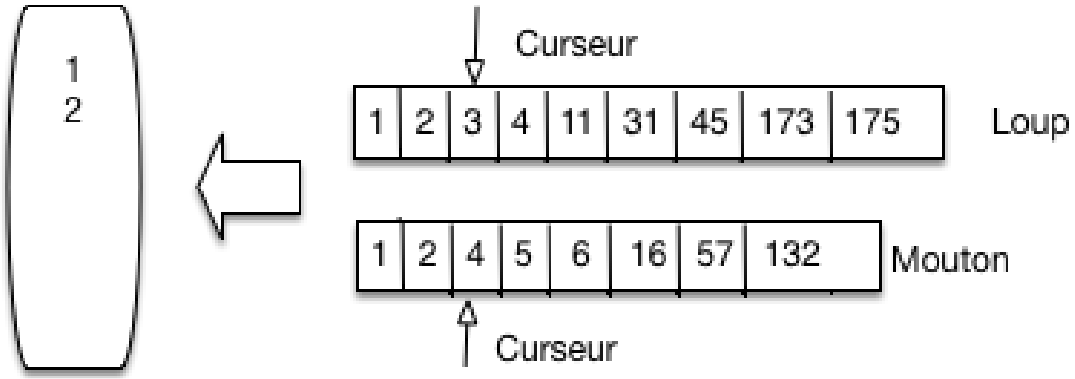
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



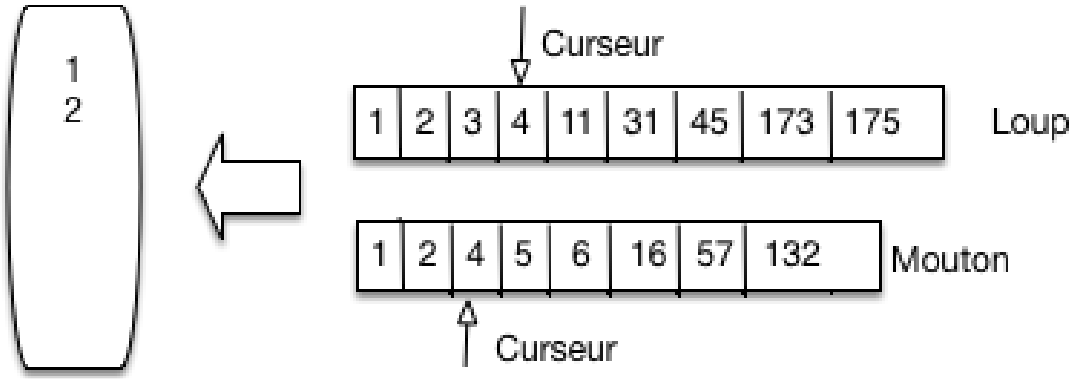
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



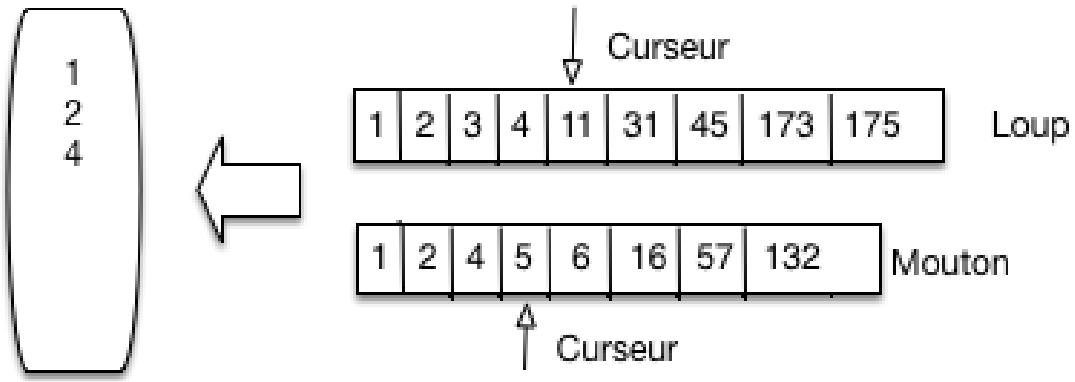
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



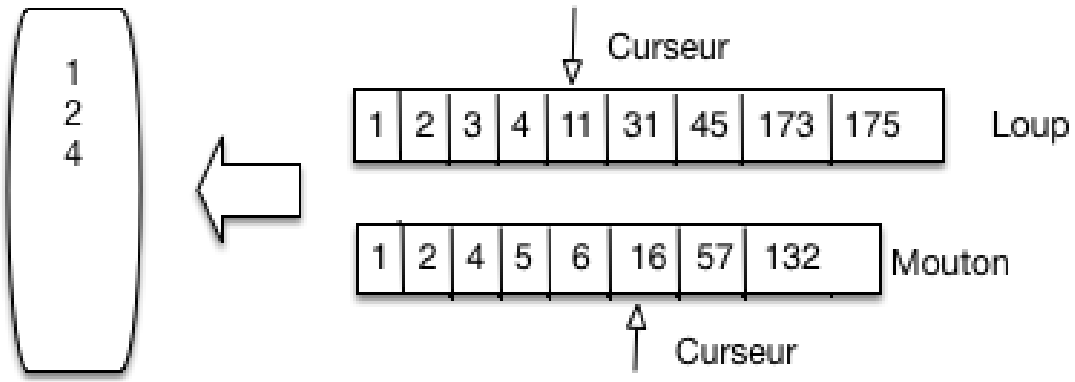
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



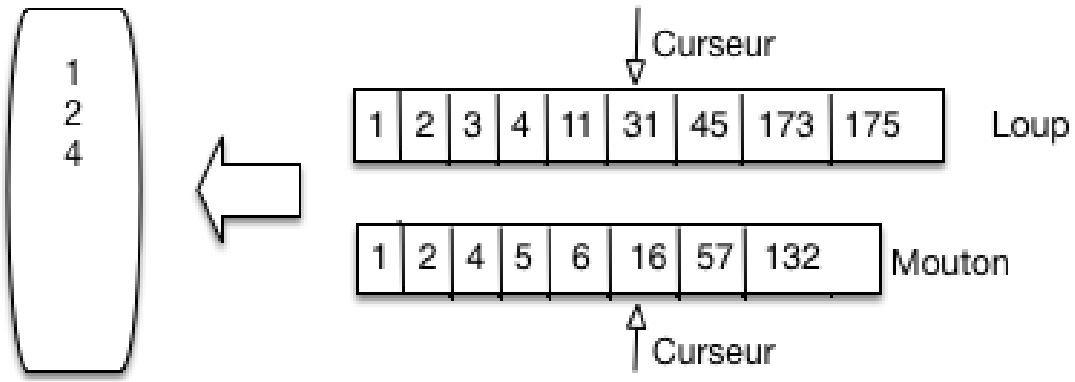
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



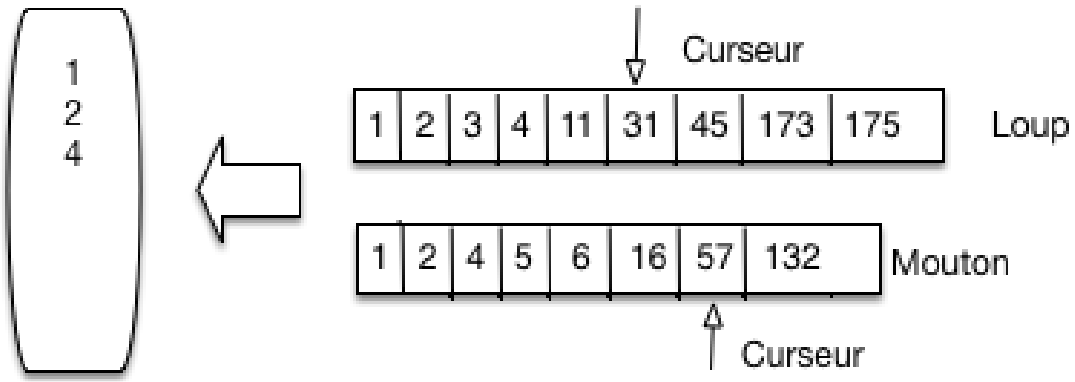
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



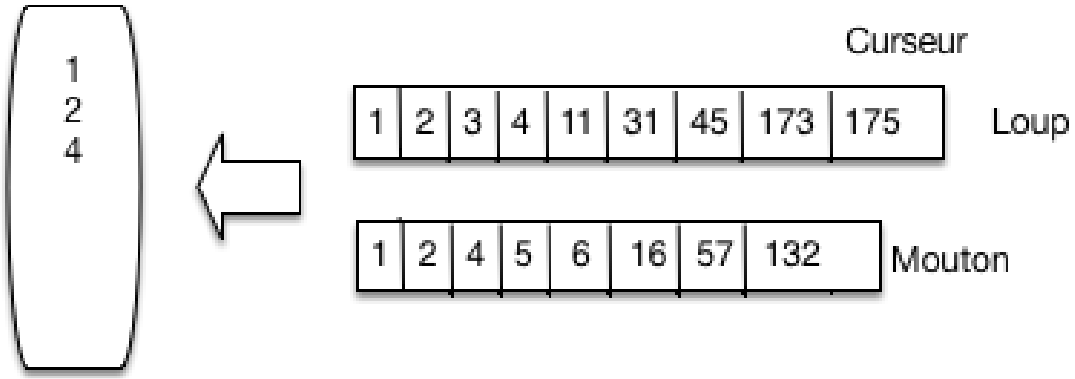
Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Traitement d'une recherche par *fusion* de listes triées

Recherchons les documents parlant de loup et de mouton.

Algorithme par **fusion** : on parcourt **séquentiellement** les deux listes.
On compare à chaque étape les docId ; s'ils sont égaux : **trouvé** !
On avance sur la liste du plus petit docId.



Un seul parcours suffit : recherche linéaire, parcours séquentiel. Pas mieux.

C'est une recherche dite **Booléenne** : pas de classement, résultat exact.

Premier bilan

En résumé : index inversé, tri et compaction, parcours linéaire très rapide sont les fondements techniques de la RI.

- Permet d'effectuer des recherches **puissantes** (combinaison de critères) et **flexibles**.
- **Garantissent une très grande efficacité.**

Que reste-t-il à faire ?

Quels termes ? Quels termes indexe-t-on ? Beaucoup moins facile que ça n'en a l'air...

Quelles requêtes ? De la plus simple ("sac de mots") à plus structurée (index struturé, requêtes Boolénnes, recherche de phrases).

Performance. Construction, compression, distribution, optimisation des accès, mises à jour, etc.

Classement ? Si j'ai des millions de documents dans le résultat, je veux les classer. Comment ?