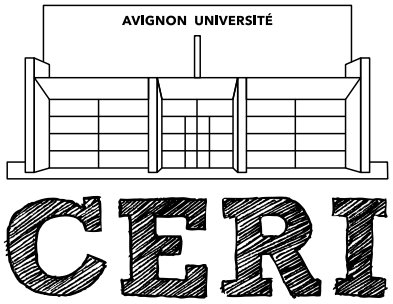


Centre d'Enseignement et de Recherche en Informatique

Partie 5
Construction d'index

Vincent Labatut

Laboratoire Informatique d'Avignon – LIA EA 4128

vincent.labatut@univ-avignon.fr

2019/20

M2 ILSEN

UE Ingénierie du document et de l'information

UCE3 Indexation & Recherche d'information

AVIGNON UNIVERSITÉ

Plan de la séance

1. Contraintes sur la construction
2. Méthode du tri par bloc
3. Méthode simple passe en mémoire
4. Autres méthodes d'indexation

Section 1

Contraintes sur la construction

AVIGNON
UNIVERSITÉ

Contraintes sur la construction
Contraintes principales

- Indexation : construction d'un index à partir d'une collection
- Tâche contrainte par :
  - Matériel :
    - Vitesse
    - Espace de stockage (mémoires primaire et secondaire)
    - Nombre de machines
  - Évolution de la collection : statique vs. dynamique
- Caractéristiques matérielles :
  - Accès :
    - Disque : $5 \times 10^{-3} + 2 \times 10^{-8}$ s (positionnement + transfert)
    - Mémoire : 5×10^{-9} s (transfert)

  → Utiliser mémoire en priorité (cache)
  → Minimiser positionnements (données contigües)
  → Transferts gérés par le bus → Processeur libre → utiliser compression

Contraintes sur la construction
Approche basique

1. Parcourir chaque fichier : mettre à jour une liste de paires (terme,docID) (gauche)
2. Trier la liste par terme puis docID (centre)
3. Fusionner les occurrences multiples dans le même document
4. Regrouper les occurrences multiples dans des documents différents pour obtenir l'index (droite)
5. Calculer les statistiques nécessaires (ex. fréquences) (droite)

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

⟹

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

⟹

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Contraintes sur la construction
Taille d'un corpus typique

- Exemple de collection réelle : Reuters RCV1
  - Dépêches de presse collectées sur une année (1996–97)
  - 800 000 documents = 1 Go de texte
  - 100 000 000 tokens et 400 000 termes
  - Nombre moyen de tokens/document : 200
  - Taille moyenne d'un token : 4,5 octets
  - Taille moyenne d'un terme : 7,5 octets
- Liste de (terme,docID) : 1,15 Go
  - (si 4 octets utilisés pour docID)

Contraintes sur la construction
Problème de mémoire

- Problème : traitement de grandes collections
  - → construction d'index impossible *en mémoire*
    - Ex. : NYT = 150 ans de dépêches
    - Principal problème : étape de tri (point 2, p.5)
- Solution : utiliser la mémoire de masse (DD)
    - Exactement le même algorithme de tri ?
      - → Lent, car beaucoup de positionnements
    - → besoin d'un algorithme spécifique, dit externe

Section 2

Méthode du tri par bloc

AVIGNON
UNIVERSITÉ

Méthode du tri par bloc
Description du principe

- eng : *Blocked sort-based indexing*
- Représentation des termes dans le lexique :
  - termID au lieu de chaîne de caractères
  - Map supplémentaire : termID → terme
  - (but : réduire la taille des fichiers)
- Principe :
  - Partition régulière des données, en blocs
  - Taille d'un bloc : déterminée pour qu'il tienne en mémoire et puisse y être trié
  - Chaque bloc est trié/inversé séparément
  - Résultats intermédiaires stockés sur DD
  - Fusion de ces résultats pour obtenir l'index (≈ étape de fusion dans un tri fusion)

Méthode du tri par bloc
Algorithme principal

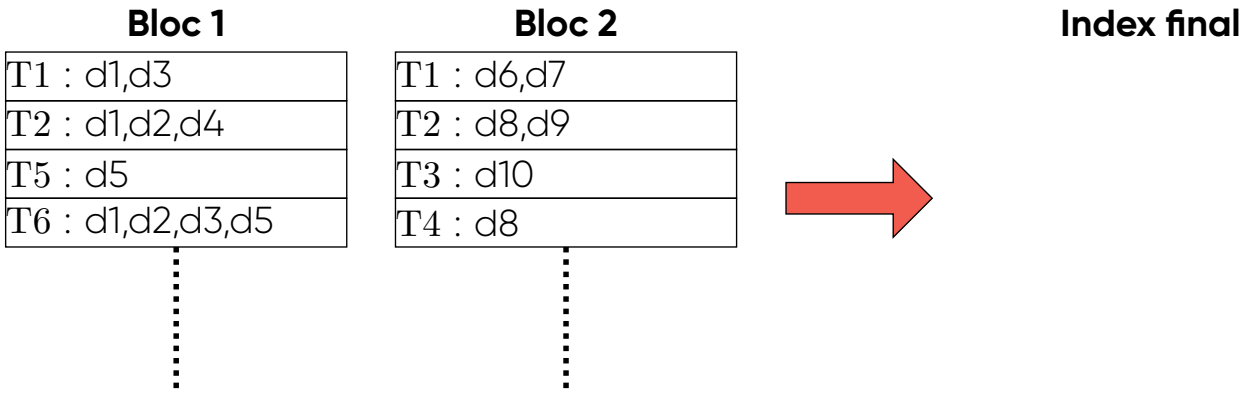
Algorithme principal :

1. Tant qu'il reste du texte :
   1. Lister les paires (termID,docID) jusqu'à atteindre une certaine taille prédéfinie
      → On obtient un nouveau bloc (≈ étape 1 p.5)
   2. Bloc inversé puis enregistré (=fichier inverse partiel) :
      1. Tri des paires (termID,docID) (≈ étape 2 p.5)
      2. Index partiel : fusion et regroupement (≈ étapes 3 & 4 p.5)
      3. Enregistrement du bloc
2. Fusion simultanée de tous les blocs

Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

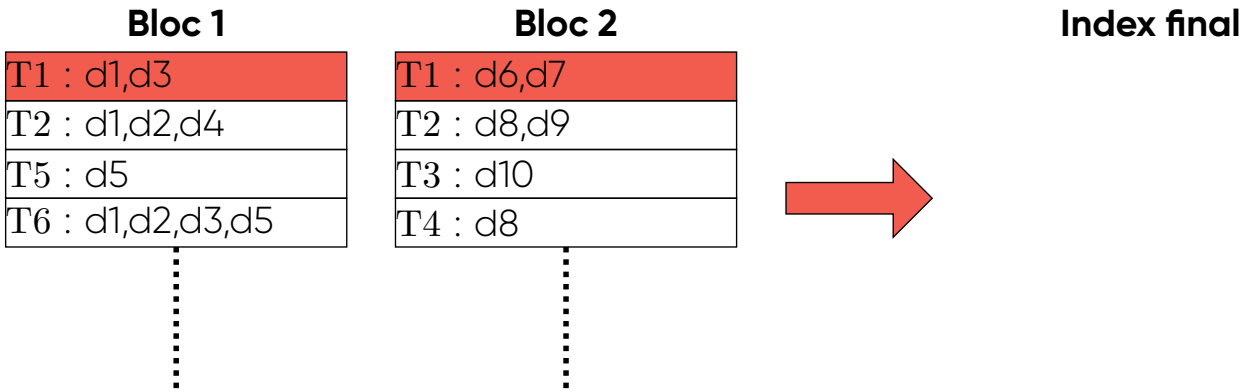
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

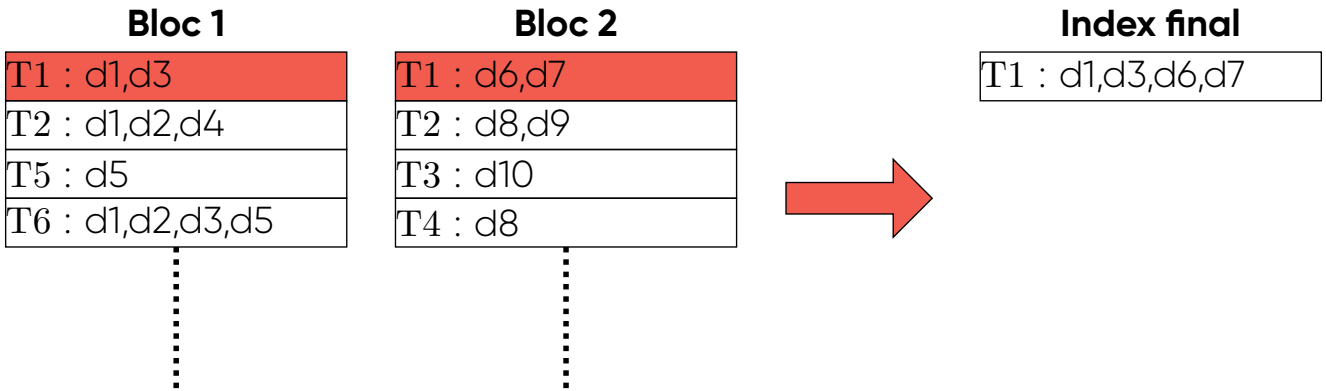
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

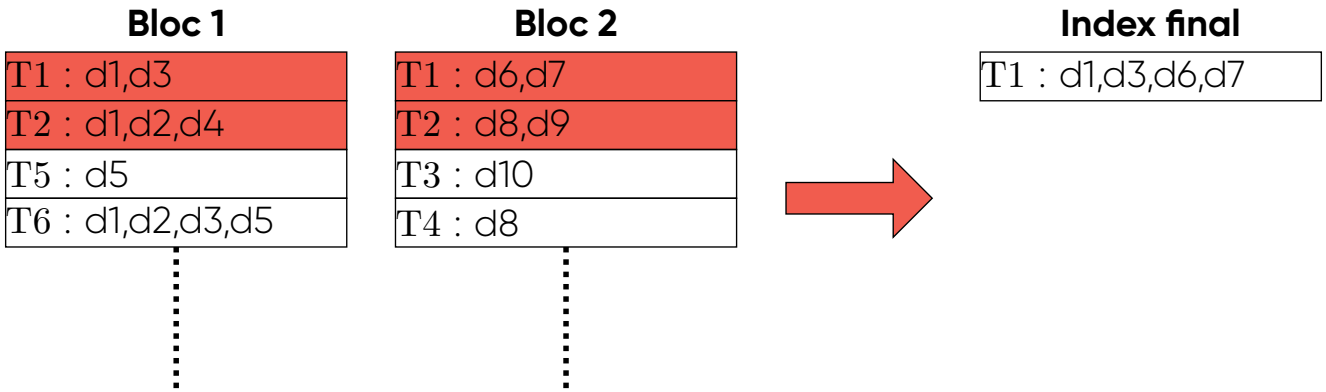
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

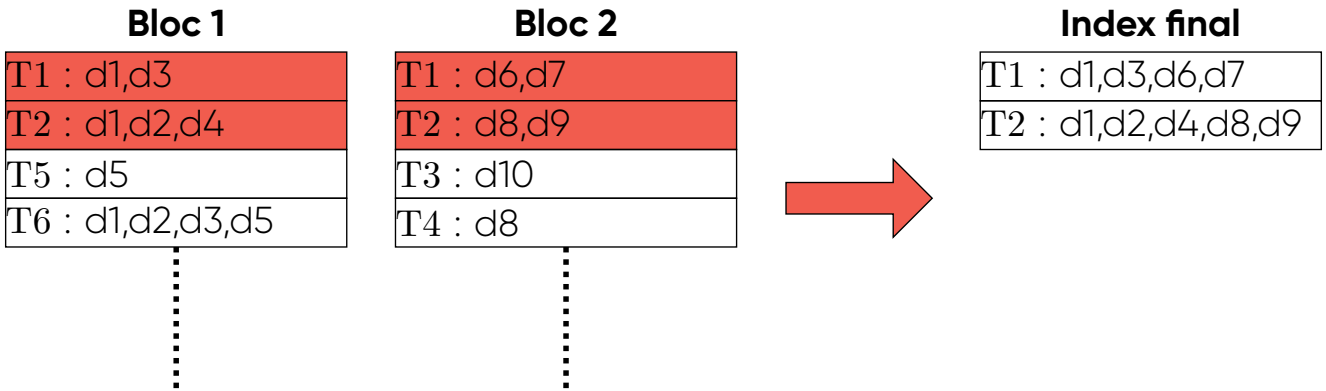
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

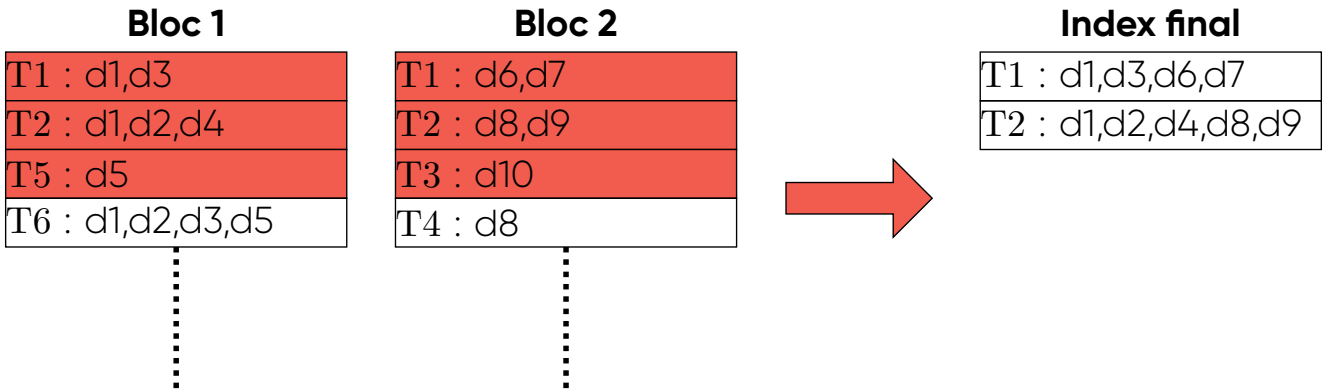
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

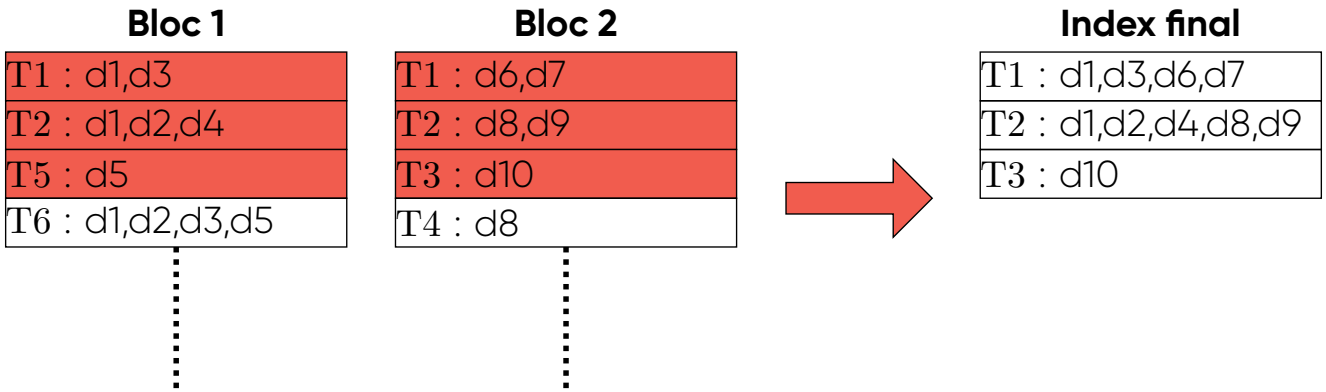
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

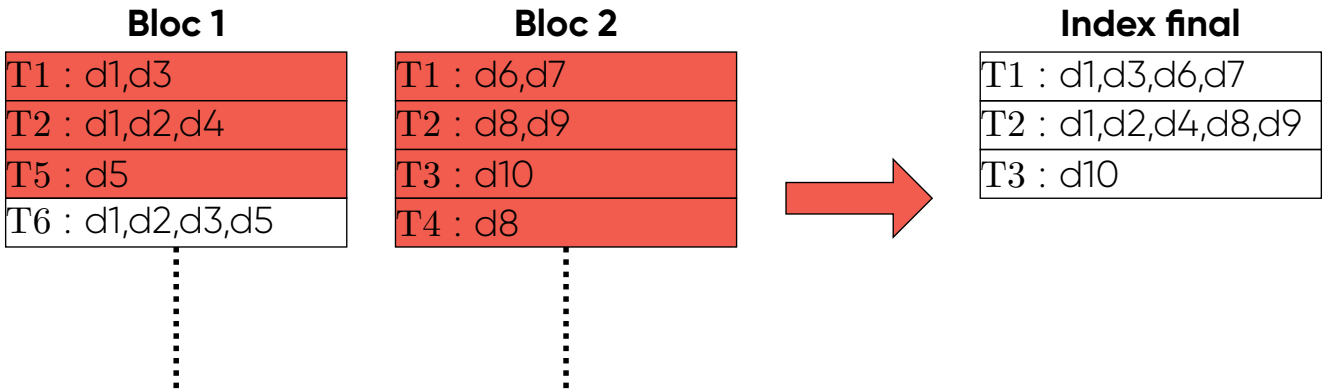
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

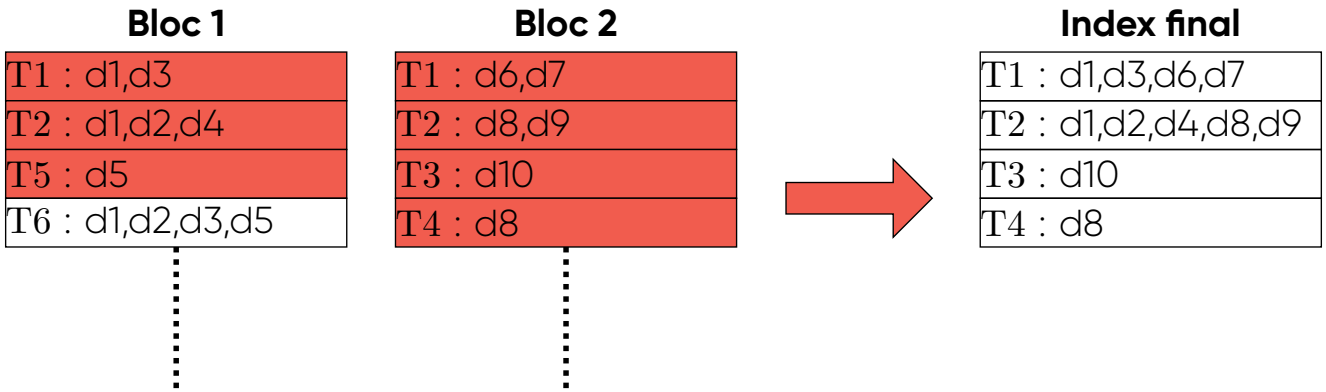
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

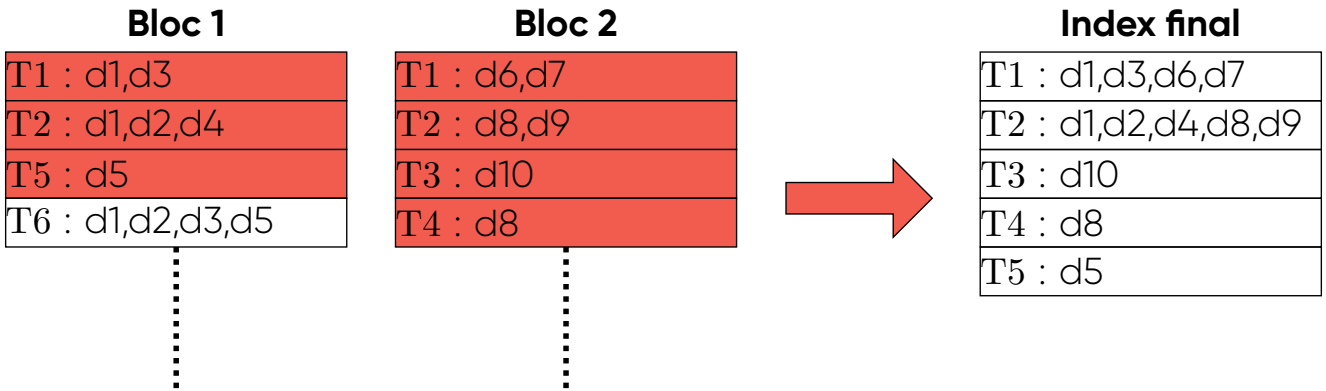
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

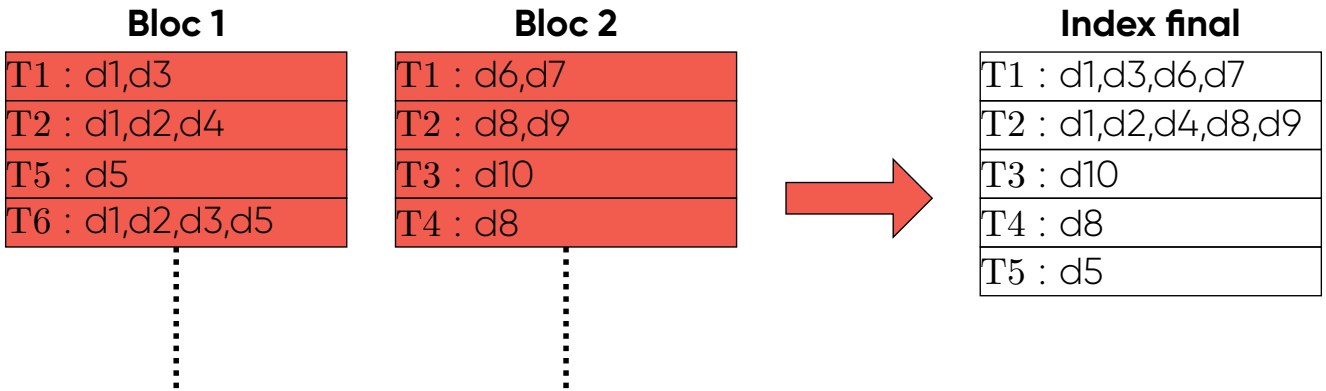
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

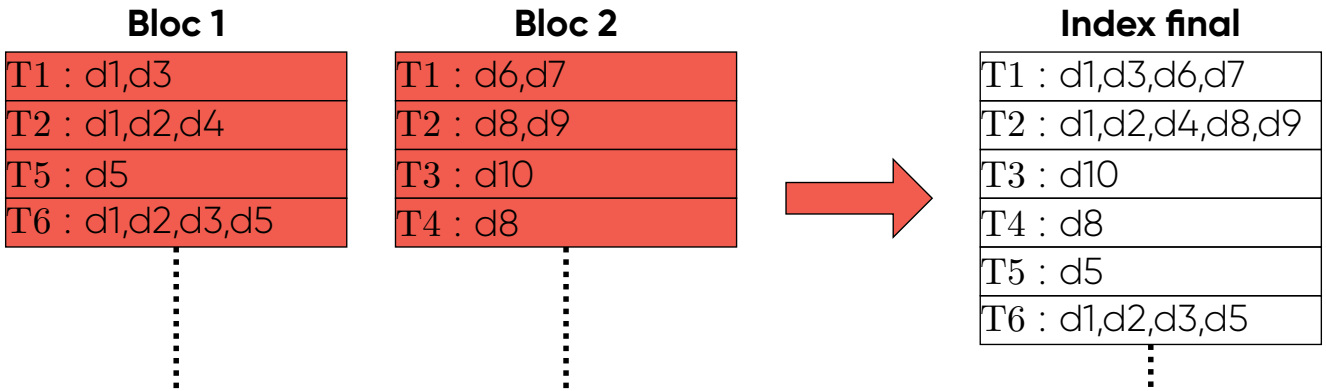
1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Étape de fusion des blocs

Fusion simultanée de tous les blocs :

1. Ouverture de tous les fichiers (y compris résultat)
2. Lire la première entrée de chacun
3. Traitement itératif :
   1. Fusionner les entrées de termID minimal
   2. Écrire l'entrée obtenue dans le fichier résultat
   3. Lire l'entrée suivante dans les blocs concernés



Méthode du tri par bloc
Propriétés

- Complexité : $\Theta(S\log(S))$
  - S : nombre de paires (termID,docID), i.e. de tokens, dans le corpus
  - Correspond à la complexité du tri (étape 1.2.1 p.10)
- Mais d'autres étapes sont plus longues en temps effectif, à cause de leurs accès disque :
  - Parsing du texte et construction des blocs (étape 1.1 p.10)
  - Fusion des blocs (étape 2 p.10)
- Limitation : la map termID $\rightarrow$ terme doit tenir entièrement en mémoire

  $\rightarrow$ Pas toujours possible, sur de grands corpus

Section 3

Méthode simple passe en mémoire

AVIGNON
UNIVERSITÉ

Méthode simple-passe en mémoire
Description du principe

- eng : *Single-pass in-memory indexing*
- Représentation des termes dans le lexique :
  - Termes (et non pas termID comme pour le tri par bloc)
  - Pas besoin de map termID $\rightarrow$ term (moins de mémoire vive)
- Principe :
  - Partition des données en blocs (comme tri par bloc)
  - Stocke (DD) un index différent pour chaque bloc (comme tri par bloc)
  - Différence avec tri par bloc : tri à l'insertion (et non plus *a posteriori*)
    - Index *partiel* du bloc construit au fur et à mesure, en insérant les tokens un par un

Méthode simple-passe en mémoire
Algorithme principal

Algorithme principal :

1. Tant qu'il reste des paires (term,docId) à traiter
   1. On initialise un nouvel index partiel
   2. Tant qu'il reste de la place en mémoire
      - Si le lexique contient déjà le terme, on rajoute le docId dans ses postings
      - Sinon, on insère le terme dans l'index partiel courant
   3. Quand plus de place en mémoire :
      1. On enregistre l'index partiel courant
      2. On commence un nouvel index partiel
2. On fusionne tous les index partiels pour obtenir l'index final

Méthode simple-passe en mémoire
Propriétés

- Étape de fusion : comme pour tri par bloc
- Tailles de listes de postings et lexique inconnues à l'avance :
  - → Besoin d'utiliser des structures dynamiques
    - Ex. : lexique=hashmap, arbre de recherche ; postings=liste chaînée...
- Complexité : $\Theta(S)$
  - Sans tri séparé, toutes les étapes dépendent linéairement du nombre total de tokens (dans le pire des cas)
- Encore plus efficace avec compression (plus gros blocs)

Section 4

Autres méthodes d'indexation

AVIGNON
UNIVERSITÉ

Autres méthodes
Index positionnel, MapReduce

- Index positionnel
  - Adaptation des deux méthodes présentées :
    - Utiliser des triplets (terme,docID,(positions)) au lieu des paires (terme,docID)
- Construction distribuée
  - Index distribué sur un cluster de machines
  - Ex. : utilisation de MapReduce (cf. [MRS08] p.75)
    - Préparation : $\mathcal{T}$ partitionné arbitrairement en k parties $\mathcal{T}_i$.
    - Split : corpus découpé en h blocs de texte.
    - Map : chaque bloc est traité par un noeud du système, qui le découpe en une séquence de paires (terme,docID), enregistrées dans k fichiers correspondant aux k parties de $\mathcal{T}$.
    - Reduce : chaque noeud traite les h fichiers associés à une $\mathcal{T}_i$, produisant un fichier unique trié correspondant à une partie de l'index global.

Autres méthodes
Indexation dynamique

- Indexation dynamique
  - Contexte : évolution fréquente des documents
  - Système de double index :
    1. Index historique (statique) : instantané du corpus à un instant donné
    2. Index auxiliaire (dynamique) : modifications survenues depuis l'instantané
  - Fonctionnement :
    1. Construction habituelle de l'index historique initial
    2. Enregistrement des modifications dans l'index auxiliaire
    3. Quand il dépasse une certaine taille : intégration ou reconstruction totale
  - Résolution des requêtes
    1. Évaluées sur index historique
    2. Filtrées/complétées en fonction d'index auxiliaire

Section 5

Conclusion

AVIGNON
UNIVERSITÉ

Concepts abordés dans cette partie

- Indexation externe
- Méthode simple-passe en mémoire
- Méthode du tri par bloc
- Indexation dynamique

Lectures recommandées

[MRS08] *Introduction to Information Retrieval*, chapitre 4.

[BCC10] *Information Retrieval : Implementing and Evaluating Search Engines*, chapitres 4 & 14.

[BR11] *Modern Information Retrieval : The Concepts and Technology behind Search*, chapitre 10.

[AG13] *Recherche d'information - Applications, modèles et algorithmes*, chapitre 2.

[CMS15] *Search Engines : Information Retrieval in Practice*, chapitre 5.

Références bibliographiques I

[AG13] M.-R. Amini et É. Gaussier. *Recherche d'information - Applications, modèles et algorithmes.* Paris, FR : Eyrolles, 2013. url : https://www.eyrolles.com/Informatique/Livre/recherche-d-information-9782212673760/.

[BR11] R. Baeza-Yates et B. Ribeiro-Neto. *Modern Information Retrieval : The Concepts and Technology behind Search.* 2nd Edition. Boston, USA : Addison Wesley Longman, 2011. url : http://people.ischool.berkeley.edu/~hearst/irbook/.

[BCC10] S. Büttcher, C. L. A. Clarke et G. V. Cormack. *Information Retrieval : Implementing and Evaluating Search Engines.* Cambridge, USA : MIT Press, 2010. url : http://www.ir.uwaterloo.ca/book/.

[CMS15] W. B. Croft, D. Metzler et T. Strohman. *Search Engines : Information Retrieval in Practice.* Pearson, 2015. url : http://www.search-engines-book.com/.

Références bibliographiques II

[MRS08] C. D. Manning, P. Raghavan et H. Schütze. *Introduction to Information Retrieval*. New York, USA : Cambridge University Press, 2008. url : http://www-nlp.stanford.edu/IR-book/.