ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

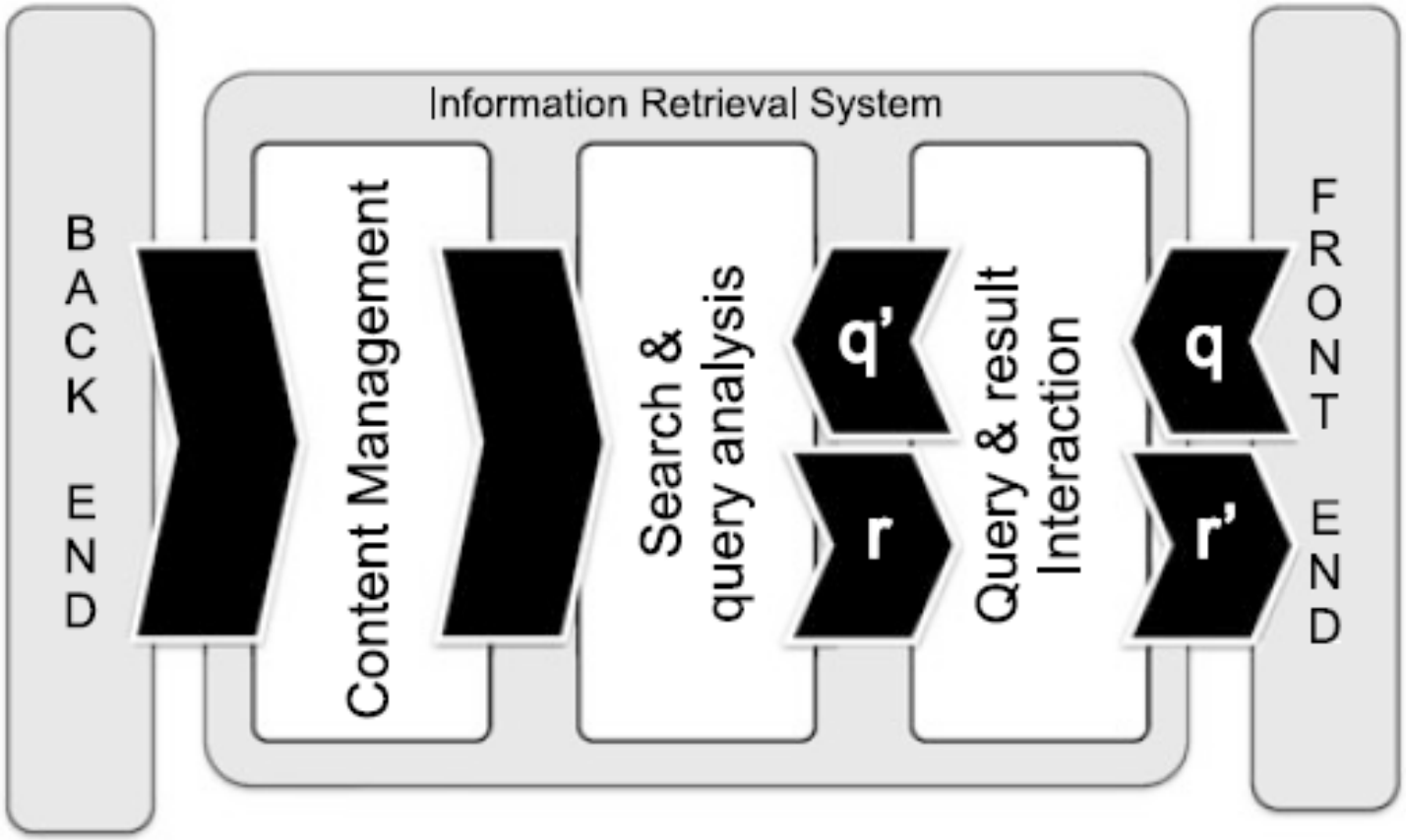
ΜΗΧΑΝΕΣ ΑΝΑΖΗΤΗΣΗΣ
(SEARCH ENGINES)

ΔΙΑΛΕΞΗ 2

ΔΙΔΑΣΚΩΝ
ΚΩΣΤΑΣ ΚΟΛΟΜΒΑΤΣΟΣ

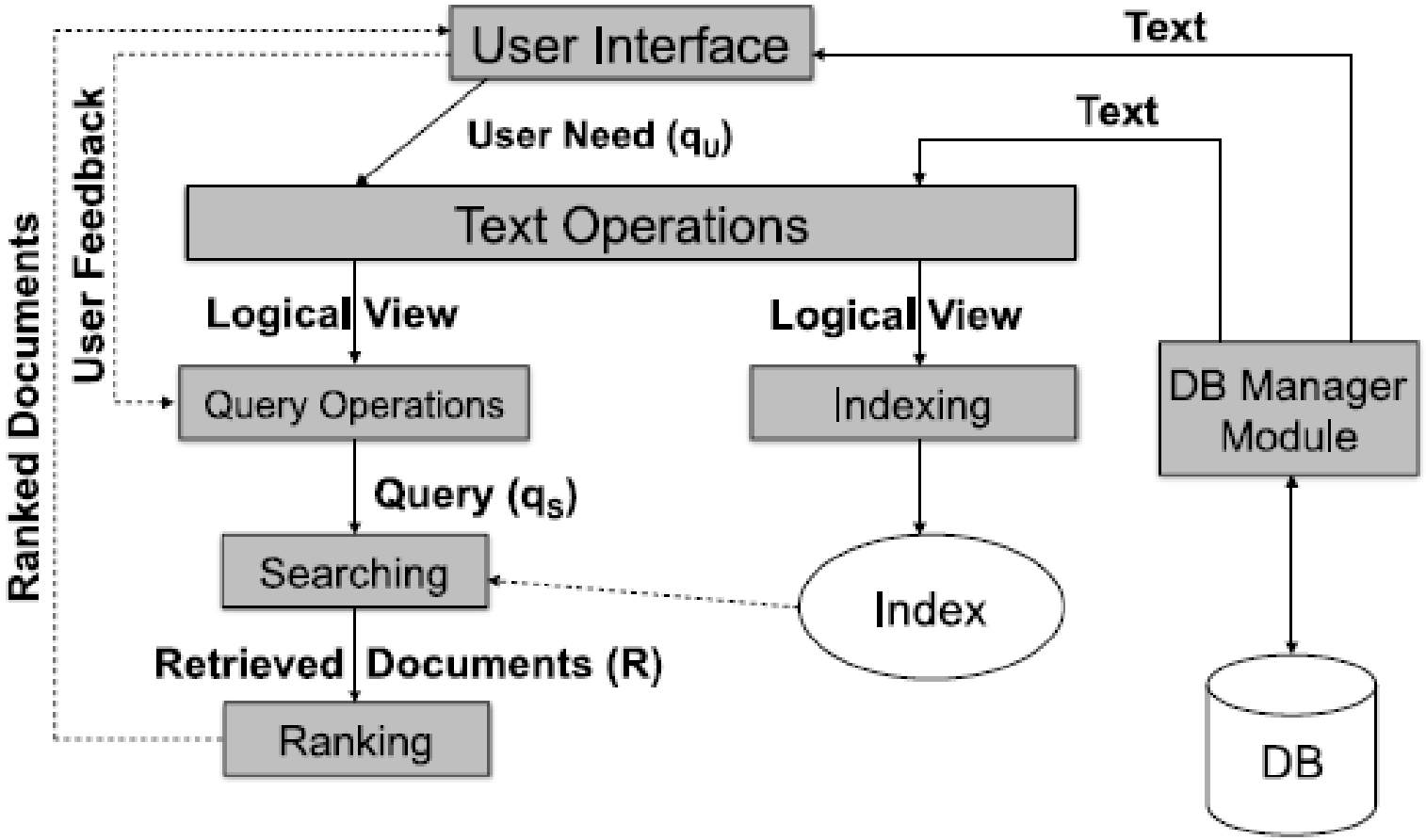
IR Process (1/3)

- Η διαδικασία απεικονίζεται στο ακόλουθο σχήμα:



IR Process (2/3)

- Αρχιτεκτονική ενός textual IR system



IR Process (3/3)

- Η εξέταση ενός μεγάλου συνόλου εγγράφων είναι αδύνατη
- Η γραμμική εξέταση δεν είναι αποδοτική
- Χρειαζόμαστε την τεχνική του **ευρετηρίου (indexing)**

- Το **ευρετήριο (index)** είναι μια λογική αναπαράσταση όπου τα έγγραφα αναπαριστώνται με ένα σύνολο index terms ή keywords
- Η λογική μας είναι να ταιριάξουμε την αναπαράσταση των ερωτημάτων με τα ευρετήρια των εγγράφων
- Οι **λέξεις κλειδιά (keywords)** πρέπει να εξαχθούν από το κάθε ένα έγγραφο ή να καθοριστούν από κάποιον expert

Indexing Process

- Τα βήματα που ακολουθούμε έχουν ως εξής:
  - Καθορισμός των πηγών των δεδομένων
  - Μετατροπή των δεδομένων στα έγγραφα ώστε να αναπαρασταθούν σε μια λογική αναπαράσταση
  - Δημιουργία του ευρετηρίου

- Ο καθορισμός των πηγών γίνεται από ένα σύστημα διαχείρισης βάσεων δεδομένων
  - Καθορίζονται τα έγγραφα, οι λειτουργίες πάνω σε αυτά, η δομή τους, ποια στοιχεία μπορούν να ανακτηθούν

Κείμενα (1/5)

- Στα κείμενα, δεν λαμβάνουμε υπόψιν μας όλες τις πιθανές λέξεις για την αναπαράσταση των εγγράφων
- Οι πιο αντιπροσωπευτικές είναι τα ουσιαστικά (nouns) ή ομάδες λέξεων που περιλαμβάνουν ουσιαστικά

- Ένα IR σύστημα προ-επεξεργάζεται τα έγγραφα για να καθορίσει τις πιο σημαντικές λέξεις για να χρησιμοποιηθούν στα ευρετήρια
- Ένα ευρετήριο πρέπει να ικανοποιεί δύο διαφορετικά και συχνά αντικρουόμενους στόχους:
  - **Exhaustiveness**. Ανάθεση ενός επαρκώς μεγάλου αριθμού στοιχείων σε ένα έγγραφο
  - **Specificity**. Αποκλεισμός γενικών στοιχείων που 'κουβαλούν' μικρή σημασιολογία (π.χ. προθέσεις, σύνδεσμοι)

Κείμενα (2/5)

- Οι γενικοί όροι / στοιχεία είναι αυτοί που έχουν υψηλή συχνότητα εμφάνισης σε ένα έγγραφο

- Οι ειδικοί όροι / στοιχεία έχουν μικρή συχνότητα εμφάνισης σε μια συλλογή εγγράφων (πλήθος εγγράφων στα οποία απαντώνται)

Κείμενα (3/5)

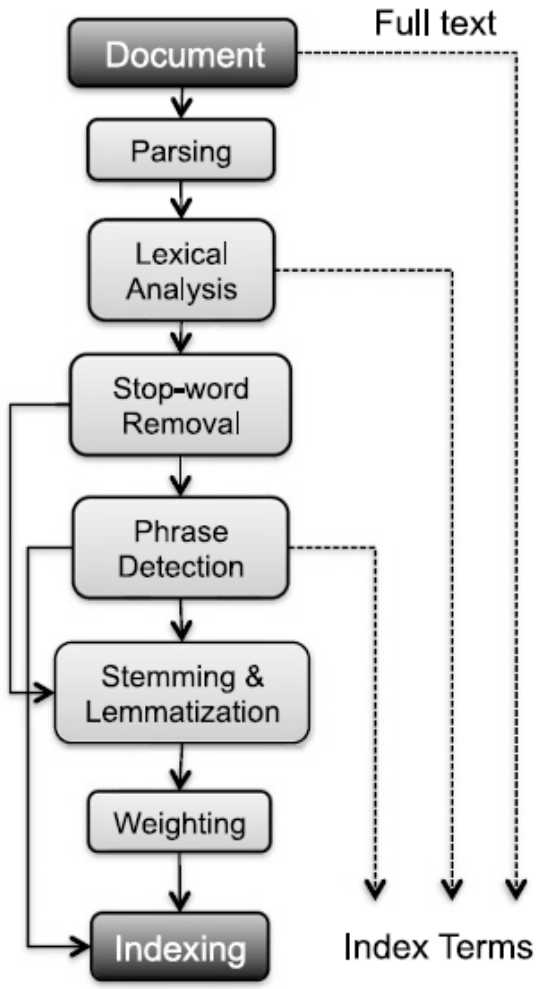
- Λειτουργίες σε κείμενα:
  - **Document parsing.** Αναγνώριση και διάσπαση των κειμένων σε τμήματα – δημιουργία των **document units**
  - **Lexical analysis.** Δημιουργούνται τα **tokens** του εγγράφου ως μια ροή λέξεων – σωστή αναγνώριση των ημερομηνιών, ακρωνύμια, κ.λπ.
  - **Stop-Word removal.** Αφαιρούμε τις πιο συχνές λέξεις.

    Παράδειγμα:

    Η φράση 'search engines are the most visible information retrieval applications'

    θα γίνει

    'search engines most visible information retrieval applications'*



* κάποιες μηχανές αναζήτησης δεν την εφαρμόζουν διότι μειώνεται το recall

Κείμενα (4/5)

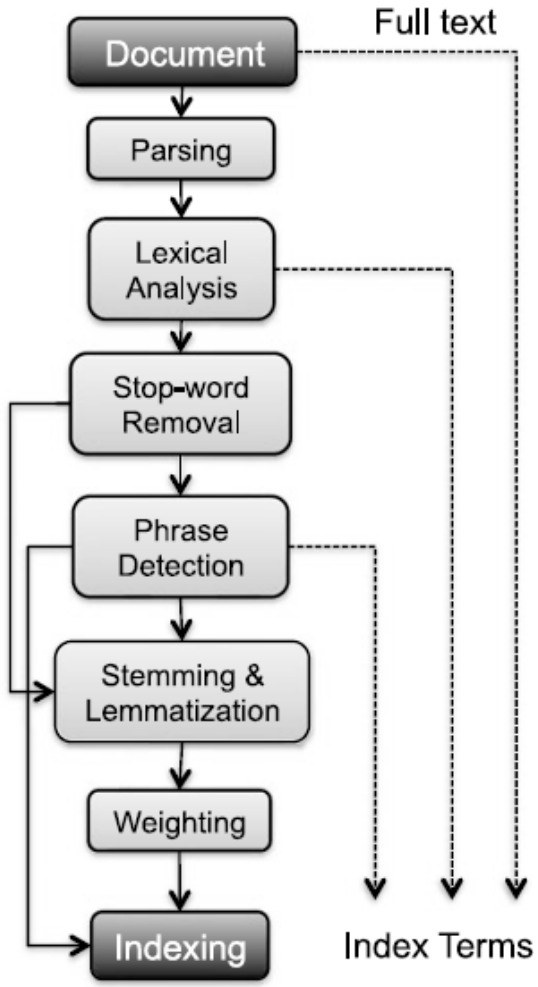
- Λειτουργίες σε κείμενα (συνέχεια):
  - **Phrase detection.** Αναγνωρίζει ουσιαστικά και άλλες εκφράσεις –υιοθετούνται κανόνες ή μορφολογική ανάλυση ή συντακτική ανάλυση ή συνδυασμός τους
  - **Stemming and Lemmatization.** Στοχεύει στην αφαίρεση των επιθημάτων από τις λέξεις

    Παράδειγμα:

    Η φράση 'search engines are the most visible information retrieval applications'

    θα γίνει

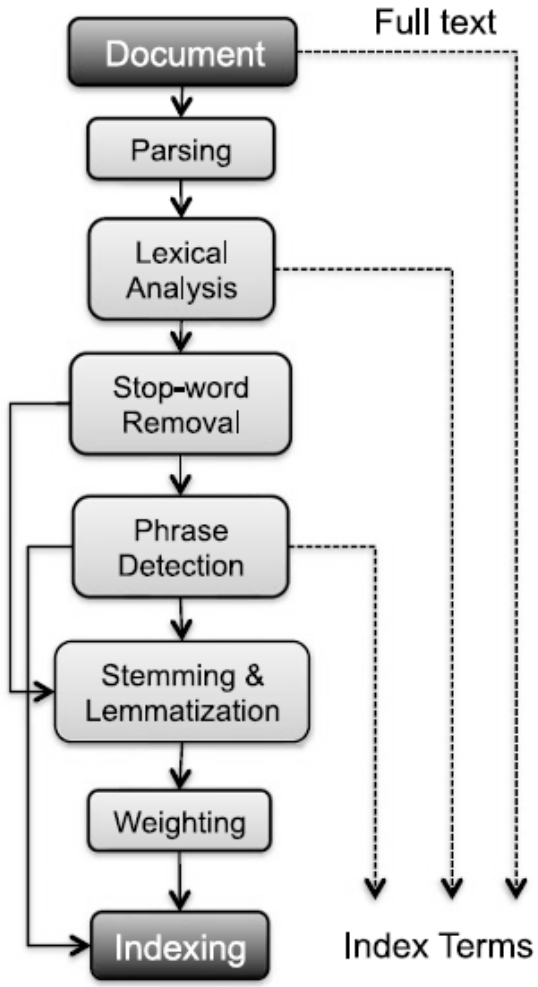
    'search engine most visibl inform retriev applic'*



* στο lemmatization υιοθετούνται λεξικά (π.χ. Wordnet) ώστε να αποτυπωθεί η βάση μιας λέξης

Κείμενα (5/5)

- Λειτουργίες σε κείμενα (συνέχεια):
  - **Weighting.** Κάθε λέξη που έχει αναγνωριστεί παίρνει ένα βάρος ανάλογα με τη σημασία της μέσα στο έγγραφο
    - Μπορούμε να αναθέσουμε το 0 αν δεν υπάρχει η λέξη ή το 1 αν υπάρχει.



Λειτουργίες σε Κείμενα

Major Steps

- Τα βήματα που ακολουθούμε για την κατασκευή του ευρετηρίου είναι:
  - Συλλογή των εγγράφων
  - Εφαρμογή του tokenization
  - Προεξεργαζόμαστε το κείμενο
  - Δημιουργούμε το ευρετήριο με τα στοιχεία και τα έγγραφα στα οποία παρουσιάζονται

Ανάκτηση της Ακολουθίας Χαρακτήρων (1/2)

- Το κάθε κείμενο είναι μια ακολουθία από bytes που πρέπει να μετατρέψουμε σε ακολουθία χαρακτήρων
- Παράδειγμα: για τα Αγγλικά είναι εύκολο αφού βασιζόμαστε στον ASCII
- Όμως μπορεί να έχουμε κωδικοποίηση για άλλες γλώσσες ή σε ειδική κωδικοποίηση ή σε UTF-8
- Πρέπει να καθορίσουμε το σωστό encoding

- Υιοθετούνται τεχνικές μηχανικής μάθησης ή ευρετικές μεθοδολογίες
- Μετά την επιλογή της κωδικοποίησης πραγματοποιείται η μετατροπή σε χαρακτήρες

Ανάκτηση της Ακολουθίας Χαρακτήρων (2/2)

- Επίσης, οι χαρακτήρες ίσως θα πρέπει να ανακτηθούν από κάποια ιδιαίτερη κωδικοποίηση όπως έγγραφα doc, zip
- Πάλι θα πρέπει να καθορίσουμε τη μορφή του εγγράφου
- Για τα plain text έγγραφα μπορεί να απαιτηθεί αποκωδικοποίηση

- Παράδειγμα:
  - Στα XML έγγραφα, χαρακτήρες όπως & πρέπει να αναπαρασταθούν στη σωστή μορφή &

- Ειδική αντιμετώπιση μπορεί να απαιτηθεί για τα pdf έγγραφα

Επιλογή των Document Units (1/2)

- Για μεγάλα έγγραφα, προκύπτει η ανάγκη για το ευρετήριο του λεγόμενου granularity
- Για μια συλλογή βιβλίων είναι κακή ιδέα να δούμε το κάθε βιβλίο σαν ένα έγγραφο
- Μια αναζήτηση για Chinese toys μπορεί να φέρει ένα βιβλίο που αναφέρει το στοιχείο China στο 1ο κεφάλαιο και το toys στο τελευταίο κεφάλαιο
- Αντίθετα, μπορούμε να φτιάξουμε ευρετήριο για κάθε κεφάλαιο ή κάθε παράγραφο σαν ένα μικρό έγγραφο
- Αφού τα έγγραφα θα είναι μικρότερα οι χρήστες ίσως μπορέσουν πιο καλά να βρουν τα σχετικά αποσπάσματα στο κείμενο

Επιλογή των Document Units (2/2)

- Αν τα τμήματα είναι πολύ μικρά, είναι πιθανό να χάσουμε σημαντικά αποσπάσματα
- Τα στοιχεία κατανέμονται σε πολλά μικρά τμήματα
- Αν τα τμήματα είναι πολύ μεγάλα, μπορεί να πάρουμε πλαστά ταιριάσματα και η σχετική πληροφορία να είναι δύσκολο να βρεθεί
- Υπάρχει ένα trade off ανάμεσα στο precision και στο recall

Tokenization (1/7)

- Δοσμένης μιας ακολουθίας χαρακτήρων και ένα τμήμα του εγγράφου, το tokenization έγκειται στην τμηματοποίηση σε κομμάτια που καλούνται tokens
- Την ίδια στιγμή μπορεί να απορρίψουμε διάφορους χαρακτήρες – π.χ. σημεία στίξης

- Παράδειγμα:

Input: Friends, Romans, Countrymen, lend me your ears;
Output: | Friends | Romans | Countrymen | lend | me | your | ears |

Tokenization (2/7)

- Το token είναι ένα στιγμιότυπο μιας ακολουθίας χαρακτήρων σε ένα συγκεκριμένο έγγραφο που ομαδοποιούνται ώστε να προκύψει μια σημασιολογική μονάδα

- Ένας **τύπος (type)** είναι μια κλάση όλων των tokens που περιέχουν την ίδια ακολουθία χαρακτήρων

- Ένα **στοιχείο (term)** είναι ένας τύπος που περιλαμβάνεται σε ένα λεξικό

- Το σύνολο των στοιχείων του ευρετηρίου μπορεί να είναι άλλα στοιχεία πέρα από τα tokens*

* στα σύγχρονα IR συστήματα αυτό δεν συμβαίνει

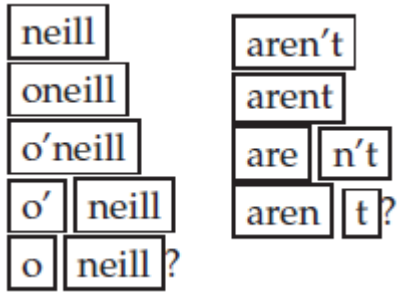
Tokenization (3/7)

- Τα tokens προκύπτουν από διάφορες επεξεργασίες που θα δούμε στη συνέχεια του μαθήματος

- Παράδειγμα:
  - Φράση: to sleep perchance to dream
  - Tokens: 5
  - Types: 4
  - Το to μπορεί να παραληφθεί ως ένα stop word
  - Προκύπτουν 3 στοιχεία: sleep, perchance, dream

Tokenization (4/7)

- Το κύριο ερώτημα στη διαδικασία είναι: **ποια είναι τα σωστά tokens;**
- Στο προηγούμενο παράδειγμα τα πράγματα είναι απλά: αφαιρούμε κενά και τους άσχετους χαρακτήρες
- Παράδειγμα:
  - Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.
  - Για το O'Neil και το aren't ποια είναι τα σωστά tokens?



  - Μια απλή στρατηγική είναι να διαχωρίσουμε σε όλους τους μη αλφαβητικούς χαρακτήρες

Tokenization (5/7)

- Τα προηγούμενα ζητήματα δυστυχώς εξαρτώνται από τη γλώσσα (**language specific**)
- Θα πρέπει να αναγνωριστεί η γλώσσα του κάθε εγγράφου
- Υπάρχουν **κατηγοριοποιητές (classifiers)** που κατατάσσουν τα έγγραφα σε μια γλώσσα
- Υιοθετούν υποακολουθίες χαρακτήρων σαν παραμέτρους
- Οι πιο πολλές γλώσσες έχουν κάποια πρότυπα - υπογραφές

Tokenization (6/7)

- Ειδικές περιπτώσεις αποτελούν οι γλώσσες προγραμματισμού, ονόματα σειρών στην τηλεόραση, κ.λπ.
- Οι νέες τεχνικές περιλαμβάνουν νέους τύπους ακολουθιών χαρακτήρων όπως τα e-mails, IP διευθύνσεις, κ.λπ.

- Γενικά όμως μπορεί να αποφεύγονται αριθμοί, χρηματικές αναπαραστάσεις, URLs, αφού η χρήση τους αυξάνει το μέγεθος των λεξικών

- Βέβαια επωμιζόμαστε το κόστος να αναζητήσουν οι χρήστες αντίστοιχες συμβολοσειρές
- Παράδειγμα: αναζήτηση του αριθμού της γραμμής όπου υπάρχει ένα bug

Tokenization (7/7)

- Hyphenation
  - Λέξεις που συνδέονται με παύλα
  - Υιοθετούνται είτε για διαχωρισμό ή για σύνδεση λέξεων
  - Παραδείγματα:
    - Co-education
    - Hewlett-Packard
  - Το πρώτο παράδειγμα θα μπορούσε να είναι ένα token
  - Το δεύτερο παράδειγμα θα μπορούσε να χωριστεί σε δύο tokens

  - Η διαχείρισή τους είναι περίπλοκη
  - Μπορούμε να το δούμε σαν πρόβλημα κατηγοριοποίησης (classification)
  - Μπορούμε να υιοθετήσουμε κάποιες ευρετικές τεχνικές

Dropping Stop Words (1/2)

- Κάποιες κοινές λέξεις που έχουν μικρή αξία για το indexing των εγγράφων καλούνται stop words
- Η γενική στρατηγική είναι να απεικονιστεί η συχνότητα των λέξεων
- Αυτές που εμφανίζονται πιο συχνά εισάγονται στη λίστα των stop words
- Παράδειγμα (Reuters-RCV1)

a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

- Αν αφαιρεθούν αυτές οι λέξεις τότε μειώνεται το πλήθος των λιστών (postings) που πρέπει να αποθηκευθούν

Dropping Stop Words (2/2)

- Οι μηχανές αναζήτησης συνήθως δεν χρησιμοποιούν λίστες με stop words
- Κάποια συστήματα εστιάζουν στα στατιστικά μιας γλώσσας ώστε να απεικονίσουμε τις πιο κοινές λέξεις
- Σε αναζήτηση προτάσεων, η χρήση stop words μπορεί να οδηγήσει σε πιο αποδοτικά ερωτήματα

- Παράδειγμα:
  - President of the Unites States
  - President AND “United States”
  - Flights to Athens

Εμπειρικοί Νόμοι

- Εμπειρικοί νόμοι που ισχύουν για κείμενα:
  - **Zipf's Law.** Η συχνότητα κάθε λέξης είναι αντιστρόφως ανάλογη της κατάταξης της στον πίνακα συχνοτήτων
  - Αν οι λέξεις έχουν ταξινομηθεί με βάση μια συνάρτηση r(w) σε φθίνουσα σειρά συχνότητας f(w) ισχύει ότι

    $r(w) \times f(w) = c$

    όπου c είναι σταθερά που εξαρτάται από τη γλώσσα (Αγγλικά: c=10)

  - **Luhn's Analysis.** Η συχνότητα μιας λέξης απεικονίζει τη σημασία της λέξης μέσα σε ένα έγγραφο
  - Η σχετική θέση σε μια πρόταση των λέξεων που έχουν μεγάλο βαθμό σημασίας είναι μια μετρική που καθορίζει τη σημασία της πρότασης

Εμπειρικοί Νόμοι

- Εμπειρικοί νόμοι που ισχύουν για κείμενα:
  - **Heap's Law.** Το μέγεθος του λεξικού V μπορεί να υπολογιστεί ως*

    $V=K\,N^{\beta}$
  - Αυτό σημαίνει πως για ένα έγγραφο το μέγεθος του λεξικού και συνεπώς το μέγεθος του index μεγαλώνει λιγότερο από γραμμικά

* N είναι το μέγεθος σε λέξεις του εγγράφου, $K \in [10,100]$ και $0<\beta<1$ (τυπικά μεταξύ 0.4 και 0.6)

Δομές Δεδομένων

Δομές Δεδομένων για Κείμενα (1/2)

- Όπως έχουμε δει το πρώτο βήμα να υιοθετήσουμε ένα διαχειριστή βάσεων δεδομένων
- Μειώνουμε το χρόνο αναζήτησης με τη χρήση των ευρετηρίων
- Μια κρίσιμη ερώτηση είναι το ποια δομή δεδομένων θα υιοθετήσουμε για την αναπαράσταση των ευρετηρίων

- Μια εύκολη λύση είναι να υιοθετήσουμε τον **term-document incidence matrix**
- Οι γραμμές τους αντιστοιχούν στα στοιχεία και οι στήλες στα έγγραφα μιας συλλογής C

Δομές Δεδομένων για Κείμενα (2/2)

- Κάθε κελί παίρνει την τιμή 1 αν το στοιχείο υπάρχει στο αντίστοιχο έγγραφο διαφορετικά παίρνει την τιμή 0
- Σε μεγάλες συλλογές εγγράφων και στοιχείων ο πίνακας είναι μεγάλος ενώ είναι επίσης αραιός

- Η πιθανότητα να συμβεί η κάθε λέξη σε κάθε έγγραφο μειώνεται όσο αυξάνεται το πλήθος των εγγράφων
- Μια βελτίωση αποτελεί η τεχνική του **inverted index**

Inverted Indexes (1/3)

- Η βασική αρχή της τεχνικής είναι πολύ απλή:
  - Αρχικά ένα λεξικό V δημιουργείται για να απεικονίσει τις εμφανίσεις των στοιχείων σε μια συλλογή εγγράφων C
  - Προαιρετικά, η συχνότητα εμφάνισης κάθε στοιχείου t_i αποθηκεύεται στη δομή
  - Έπειτα, για κάθε στοιχείο (καλείται posting) δημιουργείται μια λίστα L_i ή posting list ή inverted list και περιλαμβάνει μια αναφορά σε κάθε έγγραφο όπου το στοιχείο συναντάται
  - Το σύνολο των postings μαζί με τις λίστες καλούνται inverted index ή inverted file ή postings file

Dictionary entry	Posting list for entry
⋮	…
princess	1 3 8 22 41 55 67 68 78 120
⋮	…
witch	1 2 8 30 …
⋮	…
dragon	2 3 4 122 …
⋮	…

Inverted Indexes (2/3)

Document ID	sentence ID	text
1	1	Once upon a time there lived a beautiful princess
		⋮
1	19	The witch cast a terrible spell on the princess
2	34	The witch hunted the dragon down
		⋮
2	39	The dragon fought back but the witch was stronger

Word	Document ID
once	1
upon	1
a	1
time	1
there	1
lived	1
a	1
beautiful	1
princess	1
the	1
witch	1
cast	1
a	1
terrible	1
spell	1
on	1
the	1
princess	1
the	2
witch	2
hunted	2
the	2
dragon	2
down	2
tThe	2
dragon	2
fought	2
back	2
but	2
the	2
witch	2
was	2
stronger	2

(a) map

Word	Document ID
a	1
a	1
a	1
back	2
beautiful	1
but	2
cast	1
down	2
dragon	2
dragon	2
fought	2
hunted	2
lived	1
on	1
once	1
princess	1
princess	1
spell	1
stronger	2
terrible	1
the	1
the	1
the	2
the	2
the	2
the	2
the	2
time	1
there	1
upon	1
was	2
witch	1
witch	2
witch	2

(b) sort

Word	Document ID	Frequency
a	1	3
back	2	1
beautiful	1	1
but	2	1
cast	1	1
down	2	1
dragon	2	2
fought	2	1
hunted	2	1
lived	1	1
on	1	1
once	1	1
princess	1	2
spell	1	1
stronger	2	1
terrible	1	1
the	1	2
the	2	4
time	1	1
there	1	1
upon	1	1
was	2	1
witch	1	1
witch	2	2

(c) merge

Inverted Indexes (3/3)

Word	# Documents	Total Frequency
a	1	3
back	1	1
beautiful	1	1
but	1	1
cast	1	1
down	1	1
dragon	1	2
fought	1	1
hunted	1	1
lived	1	1
on	1	1
once	1	1
princess	1	2
spell	1	1
stronger	1	1
terrible	1	1
the	2	6
time	1	1
there	1	1
upon	1	1
was	1	1
witch	2	3

Document id	Frequency
1	3
2	2
⋮	
2	2
⋮	
1	2
⋮	
1	2
2	4
⋮	
1	1
2	2

Dictionary entry	Posting list for entry
⋮	…
princess	1 3 8 22 41 55 67 68 78 120
⋮	…
witch	1 2 8 30 …
⋮	…
dragon	2 3 4 122 …
⋮	…

Παράδειγμα

Doc 1
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

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

Συμπίεση του Λεξικού (1/2)

- Ο Heap law μας λέει πως η αύξηση του μεγέθους του λεξικού είναι της τάξης $O(N^{\beta})$
- Παράδειγμα: ένα σύνολο εγγράφων μεγέθους 1 Gb θα έχει ένα λεξικό περίπου 5 Mb
- Με αυτό τον τρόπο το λεξικό μπορεί να αποθηκευτεί στη μνήμη
- Αντίθετα, οι posting lists αποθηκεύονται στο δίσκο
- Μεγάλου μεγέθους λεξικά θα πρέπει να συμπιεστούν

Συμπίεση του Λεξικού (2/2)

- String storage
  - Το ευρετήριο μπορεί να αναπαρασταθεί είτε ως ένας πίνακας σταθερού μεγέθους στοιχείων ή
  - ως μεγάλες συμβολοσειρές που συνοδεύονται από δείκτες προς τα στοιχεία μέσα στη συμβολοσειρά
  - Με αυτό τον τρόπο το μέγεθος του λεξικού μπορεί να μειωθεί στο μισό
- Block storage
  - Τα στοιχεία ομαδοποιούνται σε ομάδες σταθερού μεγέθους k και ένας δείκτης διατηρείται ώστε να απεικονίζει το πρώτο στοιχείο σε κάθε ομάδα
  - ένα επιπλέον byte χρησιμοποιείται για να αποθηκεύσει το μέγεθος των στοιχείων
  - Αυτή η λύση 'εξαλείφει' τους k-1 δείκτες αλλά απαιτεί k επιπλέον bytes για να απεικονίσει το μέγεθος κάθε στοιχείου

B και B^+ Δένδρα (1/13)

- Η αναζήτηση σε ένα μια inverted index δομή γίνεται ως ακολούθως:
  - Αρχικά, προσπελαύνουμε το λεξικό για την αναγνώριση των στοιχείων του ερωτήματος
  - Για κάθε στοιχείο του ερωτήματος ανακτούμε τα posting files
  - Τα αποτελέσματα φιλτράρονται
    - αν το ερώτημα απαρτίζεται από πολλαπλά στοιχεία (συνδεόμενα με λογικούς τελεστές), τότε συγχωνεύουμε τις partial λίστες που εξάγονται ως αποτελέσματα
  - Στο τέλος, η τελική λίστα εξάγεται ως αποτέλεσμα

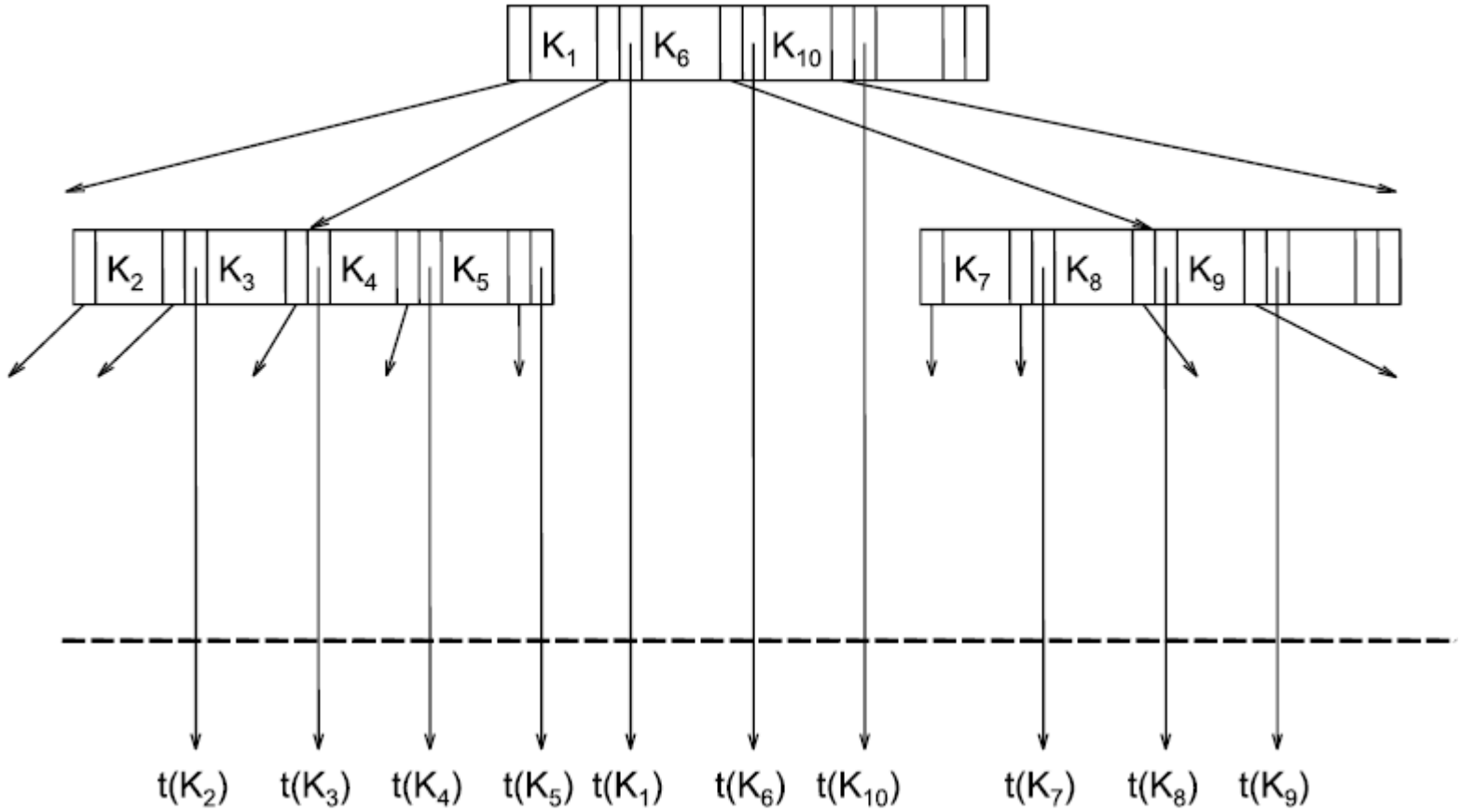
B και B^+ Δένδρα (2/13)

- Μια αποδοτική τεχνική για την αναζήτηση είναι να απεικονίσουμε τα ευρετήρια ως δένδρα αναζήτησης
- Δυο εναλλακτικές προσεγγίσεις είναι τα B και B^+ δένδρα
- Πρόκειται για γενικεύσεις των δυαδικών δένδρων αναζήτησης

B και B^+ Δένδρα (3/13)

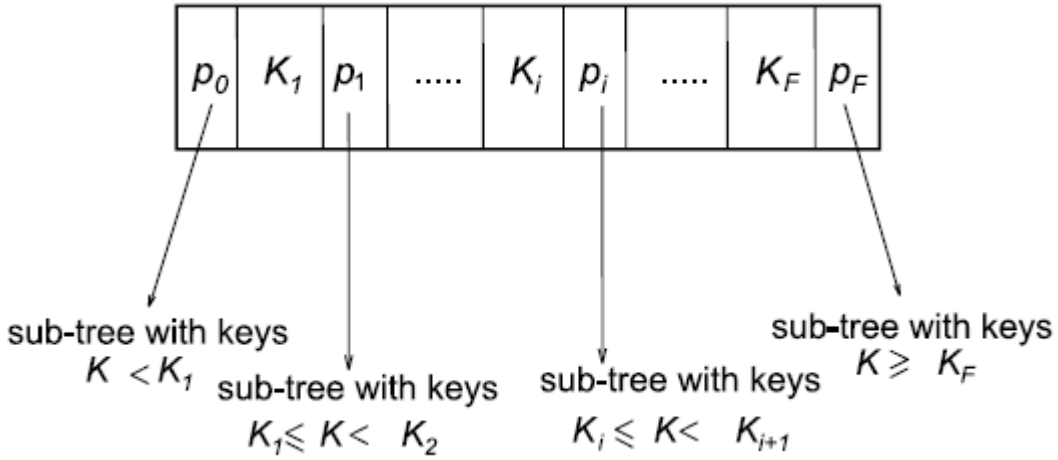
- Κάθε εσωτερικός κόμβος περιλαμβάνει ένα πλήθος κλειδιών από d μέχρι 2d

- d είναι το βάθος του δένδρου



B και B^+ Δένδρα (4/13)

- Κάθε κλειδί K_i σχετίζεται με δύο δείκτες
  - ένα προς το υπο-δένδρο με στοιχεία μικρότερα από το κλειδί
  - ένα προς το υπο-δένδρο με στοιχεία μεγαλύτερα από το κλειδί

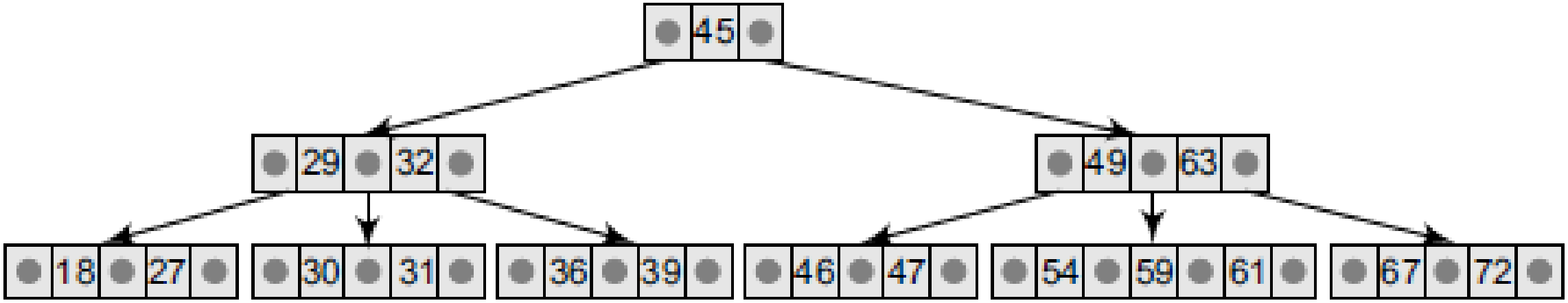


B και B^+ Δένδρα (5/13)

- Τα B-δένδρα είναι ειδική περίπτωση των M-δένδρων και υιοθετούνται για την προσπέλαση σε δίσκους
- Τα B-δένδρα έχουν όλα τα χαρακτηριστικά των M-δένδρων και επιπλέον:
  - Κάθε κόμβος έχει **το πολύ 2d παιδιά** (το δένδρο είναι τάξης 2d)
  - Κάθε κόμβος εκτός από τη ρίζα και τα φύλλα έχει **τουλάχιστον d παιδιά***
  - Η ρίζα έχει τουλάχιστον **δύο παιδιά** αν δεν είναι φύλλα
  - Όλα τα φύλλα βρίσκονται στο ίδιο επίπεδο.

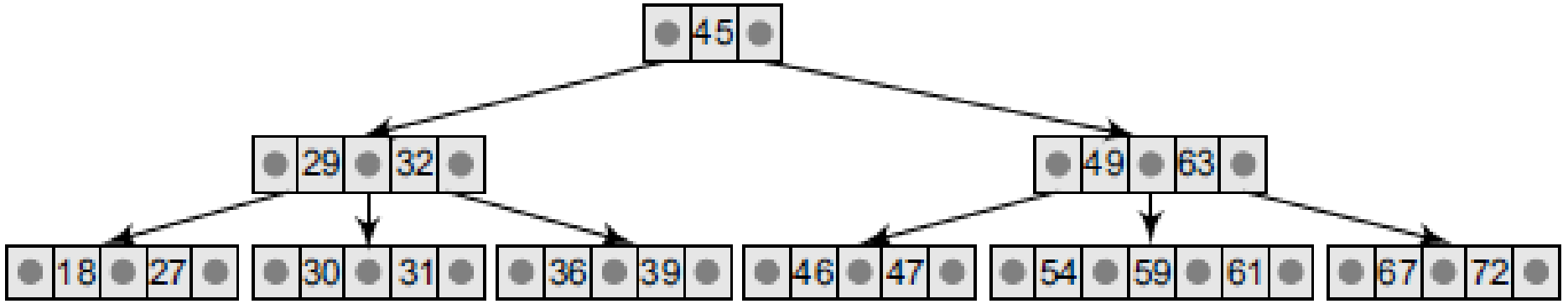
* Αυτό το χαρακτηριστικό κάνει τα B-δένδρα να χαρακτηρίζονται από μικρού μήκους μονοπάτια από τη ρίζα μέχρι και τα φύλλα.

B και B^+ Δένδρα (6/13)



B και B^+ Δένδρα (7/13)

- Η αναζήτηση στοιχείου είναι παραπλήσια με την αναζήτηση σε ένα δυαδικό δένδρο
- Για την αναζήτηση του 59 θα πρέπει να ξεκινήσουμε από τη ρίζα
- Κατευθυνόμαστε προς το δεξιό υπο-δένδρο αφού το 59 είναι μεγαλύτερο από τη ρίζα
- Στη συνέχεια το 59 είναι μεγαλύτερο από το 49 και μικρότερο από το 63, συνεπώς κατευθυνόμαστε προς το δεξιό υπο-δένδρο του 49



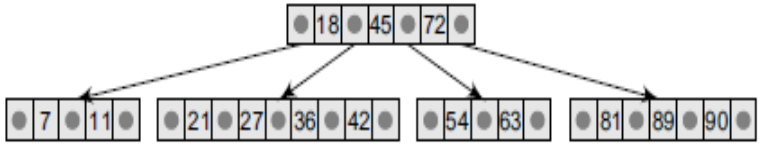
B και B^+ Δένδρα (8/13)

- Όλες οι εισαγωγές γίνονται στο επίπεδο των φύλλων
- Αρχικά, αναζητούμε τον κόμβο στον οποίο θα μπει το νέο στοιχείο και στη συνέχεια εξετάζουμε αν το φύλλο είναι γεμάτο ή όχι

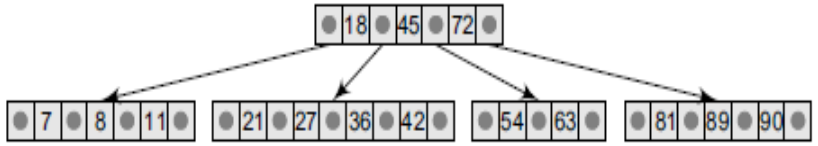
- Αν δεν είναι γεμάτο, απλά εισάγουμε το στοιχείο στη σωστή σειρά των κλειδιών
- Αν είναι γεμάτο, τότε εισάγουμε το στοιχείο στη σωστή σειρά και διασπάμε σε δύο κόμβους το συγκεκριμένο φύλλο

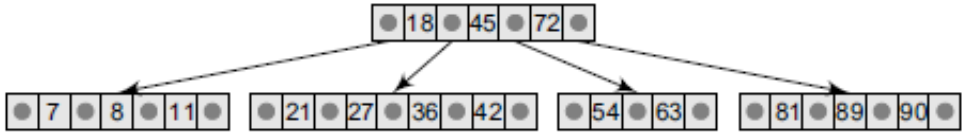
- Το κλειδί διάμεσος μετακινείται στον πατρικό κόμβο και εργαζόμαστε όπως και στην εισαγωγή ενός στοιχείου σε ένα φύλλο

B και B^+ Δένδρα (9/13)

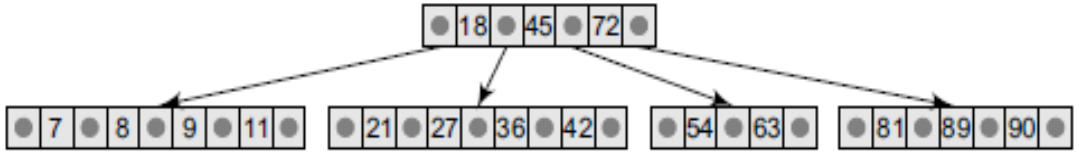


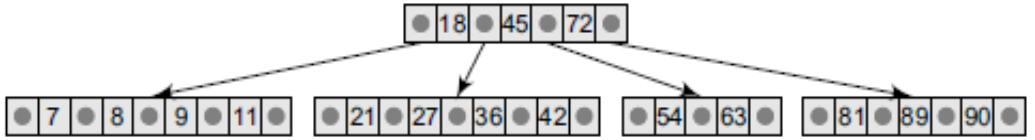
Step 1: Insert 8



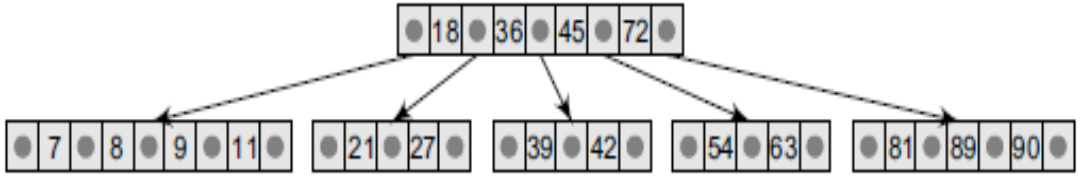


Step 2: Insert 9





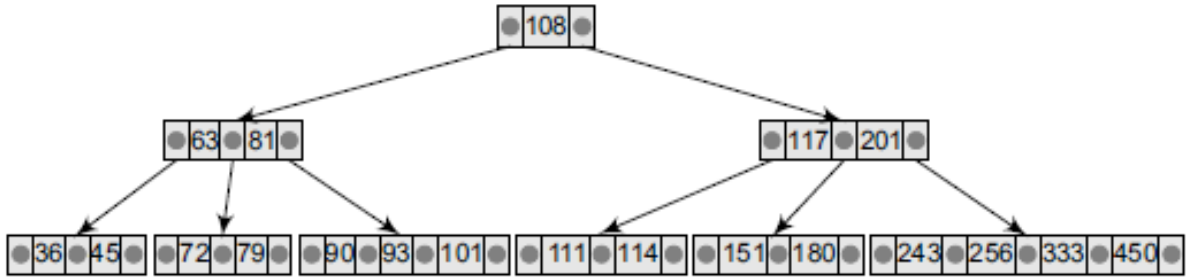
Step 3: Insert 39



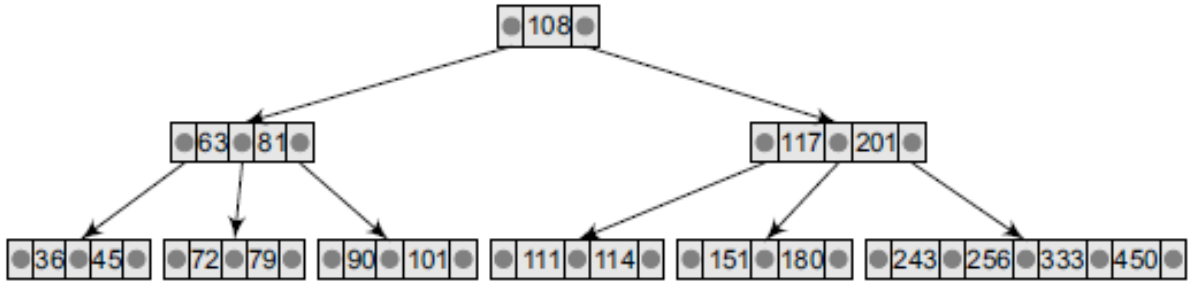
B και B^+ Δένδρα (10/13)

- Υπάρχουν δύο περιπτώσεις διαγραφής: διαγραφή σε ένα κόμβο φύλλο και διαγραφή σε ένα εσωτερικό κόμβο
- Αν ο κόμβος που περιέχει το στοιχείο προς διαγραφή περιλαμβάνει πλήθος στοιχείων πάνω από το όριο του d τότε απλά διαγράφουμε το στοιχείο
- Αν τα εναπομείναντα στοιχεία δεν είναι πάνω από το όριο του d τότε παίρνουμε στοιχείο είτε από τον αριστερό ή το δεξιό κόμβο
- Αν ο αριστερός κόμβος έχει στοιχεία πάνω από το όριο, τότε παίρνουμε το μεγαλύτερο κλειδί στον πατρικό κόμβο και κατεβάζουμε το ενδιάμεσο κλειδί του πατρικού κόμβου
- Αν ο δεξιός κόμβος έχει στοιχεία πάνω από το όριο, τοποθετούμε το μικρότερο κλειδί στον πατρικό κόμβο και κατεβάζουμε από τον πατρικό το ενδιάμεσο κλειδί
- Στην περίπτωση που και οι δύο κόμβοι-αδέρφια έχουν το μικρότερο δυνατό πλήθος στοιχείων, τότε τα συγχωνεύουμε σε νέο κόμβο στον οποίο τοποθετείται το ενδιάμεσο κλειδί του πατρικού κόμβου
- Αν ο πατρικός κόμβος μείνει με λιγότερα στοιχεία από το όριο του δένδρου, τότε η προηγούμενη διαδικασία 'ανεβαίνει' προς τα πάνω και, έτσι, μειώνεται το ύψος του δένδρου

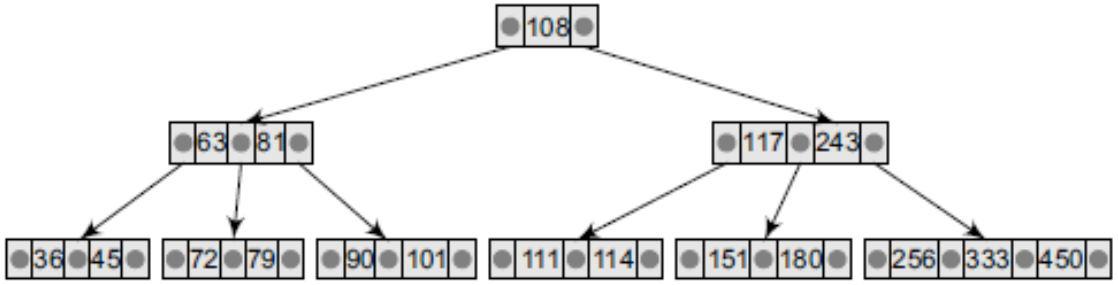
B και B^+ Δένδρα (11/13)



Step 1: Delete 93



Step 2: Delete 201



B και B^+ Δένδρα (12/13)

- Τα B^+ δένδρα είναι ειδική περίπτωση των B δένδρων
- Ενώ ένα B-δένδρο μπορεί να αποθηκεύσει κλειδιά και εγγραφές σε εσωτερικούς κόμβους, τα B^+ δένδρα αποθηκεύουν όλες τις εγγραφές στα φύλλα
- Στους εσωτερικούς κόμβους αποθηκεύονται μόνο τα κλειδιά
- Συχνά, τα φύλλα συνδέονται μεταξύ τους μέσω μιας λίστας

B και B^+ Δένδρα (13/13)

- Αποτίμηση των B δένδρων
  - Υιοθετούνται πολύ συχνά σε σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων
  - Έχουν πολύ μικρό χρόνο προσπέλασης
  - Ο μέγιστος χρόνος προσπέλασης ενός δένδρου τάξης d είναι $O(\log_d n)$ όπου n είναι το βάθος του δένδρου
  - Καταλαμβάνουν μικρό σχετικά χώρο
  - Το μειονέκτημα είναι η φτωχή επίδοση σε μια σειριακή αναζήτηση
  - Το μειονέκτημα αυτό καλύπτεται από τις συνδεδεμένες λίστες των B^+ δένδρων
  - Επίσης, μετά από πολλές εισαγωγές μπορεί να μην ισορροπημένα
  - Μπορούμε να υιοθετήσουμε τεχνικές επανεξισορρόπησης