Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
ΗΥ463 - Συστήματα Ανάκτησης Πληροφοριών
2007-2008 Εαρινό Εξάμηνο

Φροντιστήριο 4

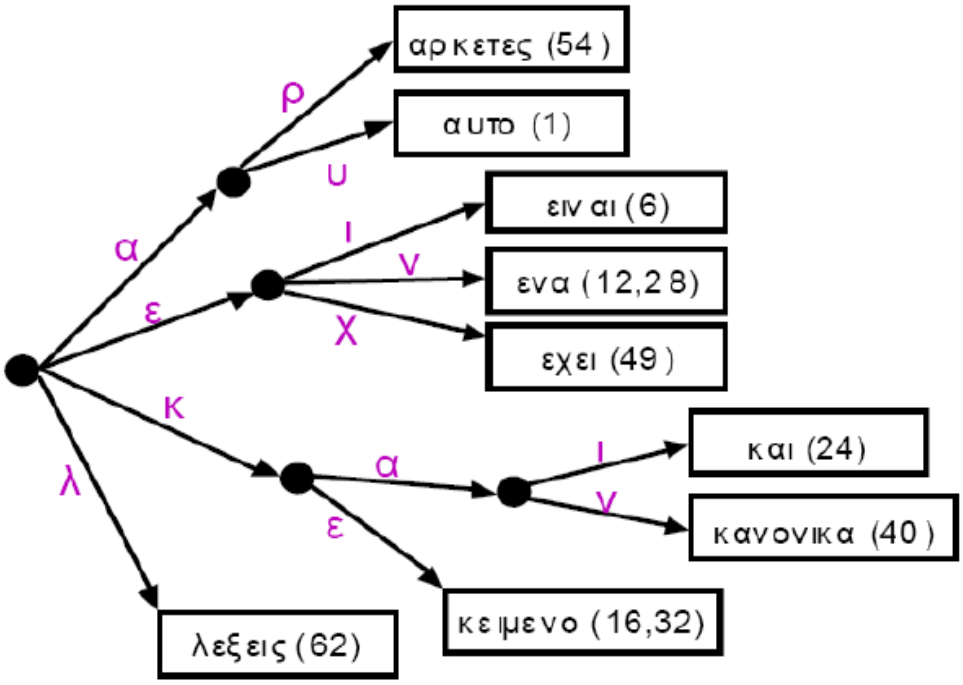
Άσκηση 1

Θεωρείστε ένα έγγραφο με περιεχόμενο «αυτό είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις». Αγνοώντας τους τόνους, σχεδιάστε:
(α) το trie του λεξιλογίου του παραπάνω εγγράφου,
(β) το δένδρο καταλήξεων του παραπάνω εγγράφου θεωρώντας ως σημεία ευρετηρίου (index points) τις αρχές των λέξεων, και
(γ) συμπτύξτε το παραπάνω δένδρο καταλήξεων στη μορφή ενός Patricia tree.

Λύση

```
    5    10   15   20   25   30   35   40   45   50   55   60   65
αυτό είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
```

(α) Το trie του λεξιλογίου είναι



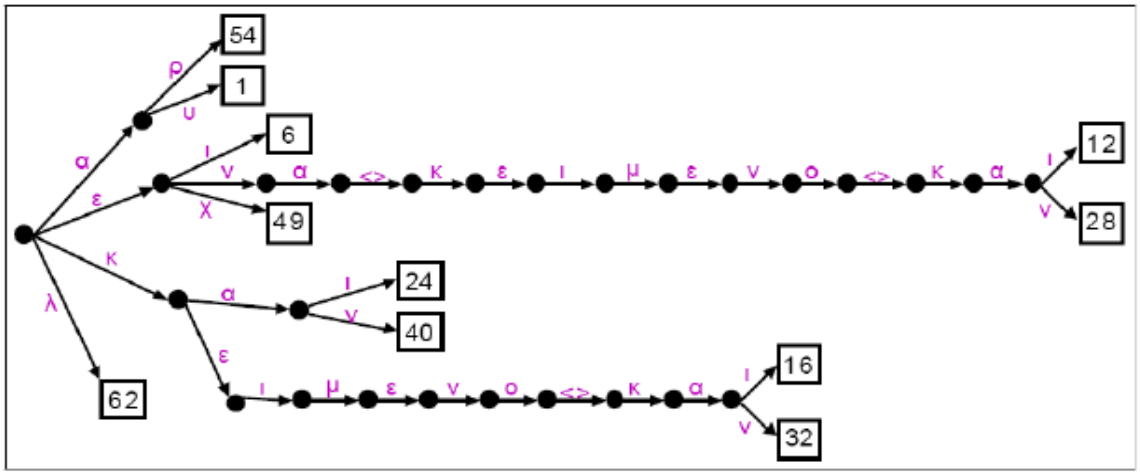
(β) Οι καταλήξεις του κειμένου είναι

αυτό είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
και ένα κείμενο κανονικά έχει αρκετές λέξεις
ένα κείμενο κανονικά έχει αρκετές λέξεις
κείμενο κανονικά έχει αρκετές λέξεις
κανονικά έχει αρκετές λέξεις
έχει αρκετές λέξεις
αρκετές λέξεις
λέξεις

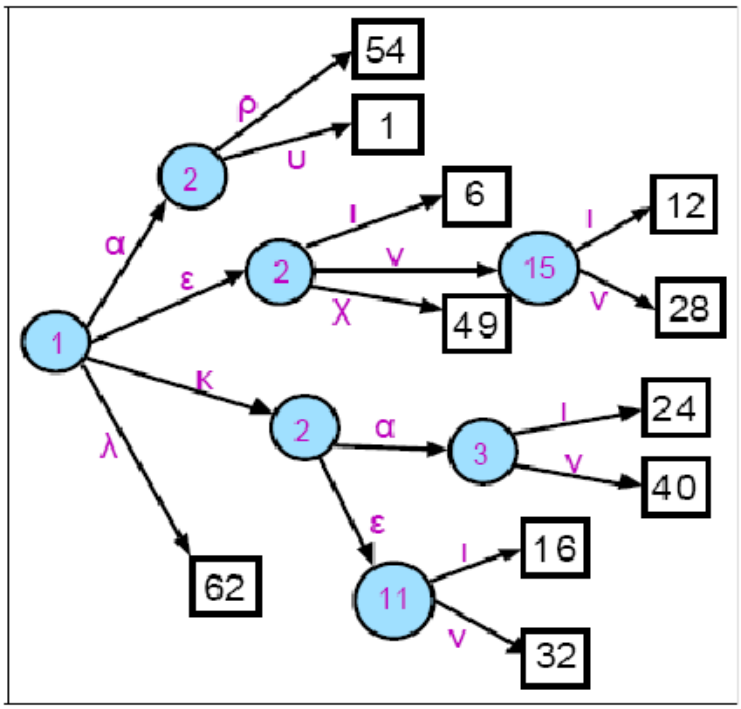
Ταξινομώντας τις παραπάνω καταλήξεις λεξικογραφικά έχουμε:

αρκετές λέξεις
αυτό είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
είναι ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
ένα κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
ένα κείμενο κανονικά έχει αρκετές λέξεις
έχει αρκετές λέξεις
και ένα κείμενο κανονικά έχει αρκετές λέξεις
κανονικά έχει αρκετές λέξεις
κείμενο και ένα κείμενο κανονικά έχει αρκετές λέξεις
κείμενο κανονικά έχει αρκετές λέξεις
λέξεις

Το δένδρο καταλήξεων που προκύπτει από τις ταξινομημένες καταλήξεις θεωρώντας ως σημεία ευρετηρίου (index points) τις αρχές των λέξεων είναι:



(γ) Το Patricia tree που προκύπτει από το παραπάνω δένδρο καταλήξεων είναι,



Άσκηση 2

Προσπαθήστε μέσω διαδικτύου να βρείτε όσο το δυνατόν περισσότερα στοιχεία σχετικά με το μέγεθος του ελληνικού παγκόσμιου ιστού (αριθμός σελίδων, μέσο μέγεθος σελίδων, πλήθος συνδέσμων, συνολικός όγκος, και όποια άλλη χρήσιμη και αξιόπιστη μέτρηση βρείτε). Σκοπός μας είναι η σχεδίαση του ευρετηρίου μιας μηχανής αναζήτησης για τον ελληνικό ιστό, το οποίο να μπορεί να φιλοξενηθεί στο μηχάνημα που έχουμε στη διάθεση μας αυτή τη στιγμή (κύρια μνήμη: 2GBytes, σκληρός δίσκος: 150GB). Βάσει αυτών που έχουμε δει στο μάθημα, εκτιμήστε το μέγεθος που θα έχει το λεξιλόγιο και οι λίστες εμφάνισης αν αποφασίζαμε να χρησιμοποιήσουμε μια δομή ανεστραμμένου αρχείου. Περιγράψτε όποια άλλη εναλλακτική ή συμπληρωματική δομή ευρετηρίου κρίνετε ότι μπορεί να επιταχύνει τη λειτουργία της μηχανής αναζήτησης.

Λύση

Στο [Crawling a Country: Better Strategies than BreadthFirst for Web Page Ordering] αναφέρεται ότι το μέγεθος των σελίδων στο .gr domain το 2004 ήταν περίπου 3.5 εκατομμύρια σελίδες. Επίσης, στο [http://www.optimizationweek.com/reviews/average-web-page/] αναφέρουν ότι το μέσο μέγεθος μίας ιστοσελίδας στο web (μόνο κείμενο, χωρίς εικόνες, scripts κλπ) είναι 25 KB. Τέλος, υποθέτουμε ότι ο μέσος αριθμός εξερχόμενων συνδέσμων που υπάρχουν ανά site είναι 20, το μέσο μέγεθος μιας λέξης είναι 10 χαρακτήρες και ότι υπάρχουν περίπου 150000 λέξεις στο ελληνικό λεξιλόγιο.

Ένα ανεστραμμένο ευρετήριο αποτελείται από το λεξιλόγιο και τις λίστες εμφάνισης κάθε λέξης που ανήκει στο λεξιλόγιο. Εφόσον πρόκειται να ευρετηριάσουμε όλο το ελληνικό domain, το λεξιλόγιο θα περιέχει όλες τις ελληνικές λέξεις και αρκετές αγγλικές, επομένως το μέγεθός του θα είναι περίπου $(150000 + 40000) \cdot 10\text{bytes} = 1.81\text{MB}$, έχοντας υποθέσει ότι θα υπάρχουν 40000 αγγλικές λέξεις επιπρόσθετα. Λόγω του μικρού του μεγέθους, το λεξιλόγιο θα μπορούσε για λόγους απόδοσης να βρίσκεται μόνιμα φορτωμένο στην κύρια μνήμη.

Γνωρίζοντας ότι κάθε σελίδα έχει μέγεθος 25 KB κατά μέσον όρο, θα περιέχει περίπου 25KB 10 B = 2560 λέξεις, όπου 10B είναι το μέσο μέγεθος μιας λέξης. Συνολικά, όλη η συλλογή των εγγράφων θα περιέχει περίπου $3500000 \cdot 2560 = 8960000000$ λέξεις. Οπότε το μέγεθος των λιστών εμφάνισης, αν για την αποθήκευση μιας εμφάνισης απαιτούνται 4 bytes, θα είναι $8960000000 \cdot 4\text{bytes} = 33.37\text{GB}$.
Αθροίζοντας το μέγεθος του λεξιλογίου και των λιστών εμφάνισης, το μέγεθος του ανεστραμμένου ευρετηρίου θα είναι περίπου $33.37\text{GB} + 1.81\text{MB} = 33.371\text{GB}$.

Άσκηση 3

Θέλετε να σχεδιάσετε ένα ΣΑΠ που να βασίζεται στο διανυσματικό μοντέλο για μια συλλογή κειμένων συνολικού μεγέθους στο δίσκο 1 Gigabyte. Έστω ότι το μέσο μέγεθος των λέξεων που εμφανίζονται στα κείμενα είναι 10 χαραχτήρες και ότι το πλήθος των διαφορετικών λέξεων της συλλογής είναι 10.000.

(α) Ποιο το αναμενόμενο (μέγιστο) μέγεθος του ανεστραμμένου ευρετηρίου για τη συλλογή αυτή;
(β) Ποιο το αναμενόμενο (μέγιστο) μέγεθος του ανεστραμμένου ευρετηρίου αν χρησιμοποιήσετε block addressing με μέγεθος block ίσο με 200 λέξεις; Τι ποσοστό μείωσης έχουμε, σε σχέση με το (α);
(γ) Αν έπρεπε το ευρετήριο να καταλαμβάνει το πολύ 1 MB (π.χ. για να χωράει στην κύρια μνήμη ενός κινητού τηλεφώνου) πως θα σχεδιάζατε το ανεστραμμένο ευρετήριο;
(δ) Αν έπρεπε να καταλαμβάνει το πολύ 100 K τι θα κάνατε;
(ε) Αν έπρεπε να καταλαμβάνει το πολύ 10 K τι θα κάνατε;

Λύση

Κατ' αρχήν θεωρούμε, είτε ότι στη γλώσσα μας δεν μας ενδιαφέρουν τα stopwords, είτε ότι ήδη στην συλλογή του 1Gbyte έχουμε ήδη αφαιρέσει τα stopwords. Ακόμα θα κάνουμε τις εξής θεωρήσεις : Σαν occurencies στους κόμβους των inverted lists θα κρατάμε σε ποιο document βρίσκεται ο όρος, σε ποιες θέσεις μέσα στο κείμενο εμφανίζεται και το tf-idf βάρος του όρου στο έγγραφο. Έτσι στην παρακάτω ανάλυση μας θεωρούμε ότι η συλλογή μας καθώς και το vocabulary αποτελούνται από μη-stopword λέξεις που γίνονται όλες indexing. Η συλλογή μας αποτελείται από 1 GByte = 1.000.000.000 bytes και με μέσο όρο 10bytes/λέξη η συλλογή μας περιέχει 100.000.000 λέξεις. Γνωρίζουμε ότι η συλλογή μας περιέχει 10.000 διαφορετικές λέξεις και συνεπώς κάθε λέξη εμφανίζεται κατά μέσο όρο 100.000.000 / 10.000 = 10.000 φορές. Τέλος αγνοούμε το μέγεθος των pointers (θα μπορούσαμε να λάβουμε υπ' όψιν 4Bytes επιβάρυνση για κάθε pointer με συνακόλουθες μικρές αλλαγές στα νούμερα παρακάτω).

α)

Για το vocabulary του ευρετηρίου θέλουμε να αποθηκεύσουμε τις 10.000 διαφορετικές λέξεις των 10byte και άρα θέλουμε χώρο 100.000 bytes = 100 KB. Αν για κάθε vocabulary entry κρατήσουμε και 4 bytes για το document frequency, το vocabulary θα καταλαμβάνει μόνο του χώρο 140K. Για κάθε μια entry από τις 10.000 θα αποθηκεύσουμε τόσα occurencies σε όλα τα κείμενα όσες φορές εμφανίζεται η λέξη και ήδη αναφέραμε ότι κάθε λέξη εμφανίζεται κατά μέσο όρο 10.000 φορές και άρα θέλει 10.000 integers για να αποθηκευτούν όλα τα occurencies μιας λέξης. Συνεπώς για ένα entry του vocabulary θέλουμε 10.000 * 4 bytes = 40.000 bytes = 40 KB. Αυτό μόνο για τις θέσεις των εμφανίσεων της ίδιας λέξης. Εκτός από αυτό πρέπει να αποθηκεύσουμε σε ποια έγγραφα υπάρχουν αυτές οι εμφανίσεις καθώς και το tf-idf βάρος της (εμφάνισης της) λέξης στο κείμενο. Επειδή ψάχνουμε το μέγιστο μέγεθος ευρετηρίου που μπορεί να έχουμε θα κάνουμε την χειρότερη από άποψη χώρου υπόθεση, ότι κάθε εμφάνιση μιας λέξης βρίσκεται σε διαφορετικό κείμενο και άρα θέλουμε (για κάθε λέξη) 10.000 integers για να κρατάμε τα id του κειμένων που αυτή εμφανίζεται και άλλους 10.000 integers για τα tf-idf βάρη. Συνεπώς για ένα entry του vocabulary θέλουμε (μιλάμε πάντα κατά μέσο όρο) 40K για τις θέσεις των εμφανίσεων στα κείμενα, 40K για τα id των documents και άλλα 40K για τα βάρη. Σύνολο 120K για κάθε μια από τις entries του vocabulary. Συνεπώς για όλες τις inverted lists θέλουμε χώρο 120K * 10.000 = 1.200.000.000 bytes = 1,2 G.

Εν κατακλείδι για το ανεστραμένο αρχείο θέλουμε 1,2G + 140K = 1200,14 Mbytes.

β)

Το να χρησιμοποιήσουμε block addressing με μέγεθος block = 200 λέξεις * 10 byte/λέξη = 2000 bytes δεν έχει σαν αποτέλεσμα καμία βελτίωση στο μέγεθος του ευρετηρίου σε σχέση με το ερώτημα (α). Αυτό συμβαίνει διότι με μέγεθος block 2000 bytes , η 1G συλλογή μας χωρίζεται σε 500.000 blocks (περίπου), τα οποία είναι πάρα πολλά σε σχέση με το πλήθος των διαφορετικών λέξεων της συλλογής μας. Με την υπόθεση ότι οι εμφανίσεις μιας λέξης κατανέμονται ομοιόμορφα μέσα στο κείμενο, μια λέξη εμφανίζεται κάθε 10.000 άλλες άρα μια λέξη εμφανίζεται κάθε 10.000 / 200 = 50blocks. Κατά συνέπεια το πλήθος των occurencies κάθε λέξης θα παραμείνει το ίδιο αφού ναι μεν πλέον ένας integer occurence θα δηλώνει θέση μπλόκ και όχι θέση στο κείμενο αλλά επειδή με μεγάλη πιθανότητα οι εμφανίσεις της ίδιας λέξης θα είναι σε διαφορετικά μπλοκ το πλήθος των μπλοκς που θα εμφανίζεται μια λέξη θα είναι 10.000 και θέλουμε τόσο χώρο όσο και στο (α). Κάνουμε πάλι την χείριστη υπόθεση ότι κάθε μπλοκ (και συνεπώς κάθε εμφάνιση λέξης) είναι σε διαφορετικό κείμενο, και άρα πάλι θέλουμε 3 ακέραιους (αριθμό μπλοκ, id εγγράφου, tf-idf βάρος) ανά κόμβο λίστας, έχοντας 10.000 κόμβους για κάθε μια από τις 10.000 λέξεις.

Συνεπώς και πάλι για το ανεστραμμένο αρχείο θέλουμε 1200,14 Mbytes.

γ)

Σε αυτό το ερώτημα θα χωρίσουμε τη συλλογή των εγγράφων μας σε μεγάλα μπλοκς, ώστε να χωρούν πολλές εμφανίσεις της ίδιας λέξης σε ένα block, και για κάθε λέξη στο ευρετήριο θα κρατάμε μόνο ακεραίους που θα είναι τα μπλοκ που βρέθηκε η λέξη. Το vocabulary του ευρετηρίου από μόνο του (και χωρίς document frequency πληροφορία) καταλαμβάνει 100K και άρα έχουμε ακόμα 900K στην διάθεσή μας. Αυτά τα 900K θα κατανεμηθούν σε 10000 εγγραφές (μια για κάθε λέξη) και έτσι κάθε inverted list θα έχει περίπου 90 bytes (μέσο όρο) χώρο για πληροφορίες σχετικές με την λέξη. Σε αυτά τα 90 bytes πρέπει να περιγραφούν και οι (κατά μέσο όρο) 10.000 εμφανίσεις της λέξης. Αν θεωρήσουμε, όπως παραπάνω, ότι για κάθε μπλοκ θα κρατούμε 4 bytes, τότε θέλουμε 22,5 μπλοκς (90=22,5*4) που θα έχουν τις 10000 εμφανίσεις της λέξης. Άρα σε ένα μπλοκ θέλουμε να εμφανίζεται η λέξη 10.000/ 22,5 = 444,44 φορές κατά μέσο όρο. Όπως αναφέραμε και στο ερώτημα (β) η ιδια λέξη εμφανίζεται κάθε 10.000 άλλες λέξεις και άρα για να περιέχει ένα μπλοκ 444,44 φορές την ίδια λέξη πρέπει να περιέχει συνολικά 10.000 * 444,44 λέξεις = 4.444.400 λέξεις. Η κάθε λέξη είναι 10 bytes και άρα ένα μπλοκ πρέπει να έχει μέγεθος 44.444.000 bytes = 44,44 Mbyte.

Συνεπώς με μπλοκ των 44,44 Mbyte, το ευρετήριο (κρατώντας πληροφορία μόνο 4 byte/block) έχει μέγεθος 90 bytes (ανά εγγραφή) * 10.000 διαφορετικές εγγραφές + 100K (το vocabulary) = 1Mbyte.
Αν θέλαμε να κρατάμε πιο πολύ πληροφορία ανά μπλοκ, π.χ ένα ακόμη byte (έστω έναν pointer σε μια άλλη δομή στο δίσκο η οποία θα κρατά βάρη ή/και άλλη πληροφορία), τότε για παράδειγμα θα έπρεπε να έχουμε τα μισά μπλοκ σε πλήθος (αφού θα κρατούσαμε την διπλάσια πληροφορία) και συνεπώς διπλάσιο μέγεθος μπλοκ.

δ)

Το να καταλαμβάνει το ευρετηρίο 100K σημαίνει ότι το vocabulary ίσα που χωράει στη μνήμη αφού αυτό από μόνο του είναι 100K. Δεν έχουμε πολλές επιλογές παρά να κρατάμε στη μνήμη pointers για θέσεις στο δίσκο όπου βρίσκεται η υπόλοιπη πληροφορία του ευρετηρίου. Όμως το vocabulary καταλαμβάνει και τα 100K και δεν αφήνει χώρο για τίποτα άλλο. Μπορούμε να εφαρμόσουμε

stemming στο vocabulary ώστε το μέγεθος της μέσης λέξης να μετατραπεί από 10 σε 6 bytes. Έτσι θα περισσεύουν 4 bytes ανά λέξη που θα μπορούσε να μπει ο pointer και πάλι να έχουμε 100K χώρο. Ακόμα και εάν έχει εφαρμοστεί stemming ήδη, το μεγεθος της μέσης λέξης (10 bytes) μας αφήνει περιθώρια να την κόψουμε και να εφαρμόσουμε την ιδέα μας. Έχουμε βέβαια το πρόβλημα ότι για μια λέξη που έχει κοινό πρόθεμα με κάποια του vocabulary μας πρέπει να φτάσουμε ώς το δίσκο για να καταλάβουμε ότι αποτύχαμε.

ε)
Όλες οι ιδέες που παρουσιάζονται (blocks,stemming και η παρούσα ιδέα) μπορούν βέβαια να εφαρμοστούν μόνες τους ή συνδυασμένες σε όλα τα ερωτήματα. Αυτό που μπορούμε να κάνουμε όταν έχουμε τόσο μικρή μνήμη όσο 10K είναι να «σπάσουμε» το ευρετήριο του προηγούμενου ερωτήματος σε δέκα κομμάτια και να έχουμε 1 κομμάτι κάθε φορά στη μνήμη. Αν δεν περιέχει τη λέξη που ψάχνουμε πάμε στο δίσκο και φορτώνουμε ένα άλλο κομμάτι. Η παραπάνω διαδικασία είναι πάρα πολύ αργή και μια βελτιστοποίηση είναι να χωρέσουμε κάποιες από τις λέξεις (π.χ. 900) αλφαβητικά επιλέγοντάς τες όσο περισσότερο ομοιόμορφα μπορούμε από όλο το σύνολο του vocabulary. Ανάμεσα σε λέξεις (π.χ κάθε 10 λέξεις) θα κρατούμε έναν pointer σε ένα τμήμα του vocabulary που είναι 10K και αλφαβητικά ανάμεσα στην λέξη πριν από τον pointer και στην επόμενη.

Αν ψάχνουμε μια λέξη λεξικογραφικά ανάμεσα σε «αστο» και «λαλα» ακολουθούμε τον pointer και φορτώνουμε εκείνο το κομμάτι του vocabulary που βρίσκεται λεξικογραφικά ανάμεσα σε «αστο» και «λαλα» και περιέχει ίσως την λέξη που θέλουμε. Αν χρειάζεται μπορούμε να προσθέσουμε και άλλα επίπεδα indexing (δηλαδή αυτό το τμήμα που θα φορτώσουμε να έχει και αυτό pointers σε λεξικογραφικά εμπεριέχοντα τμήματα). Όπως και να έχει το τελευταίο επίπεδο indexing του vocabulary πρεπει να περιέχει pointers στον δίσκο για περαιτέρω πληροφορία (pointers π.χ. σε inverted lists).