Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

GRAPHIT-db:ΠΡΟΤΥΠΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΓΡΑΦΩΝ(II)

Διπλωματικη Εργασια

της

Θεοδώρας Κοντογιάννη

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εργαστηριο Συστηματων Βασεων Γνωσεων και Δεδομενων
Αθήνα, Ιούλιος 2009

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων

GRAPHIT-db:ΠΡΟΤΥΠΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΓΡΑΦΩΝ(II)

Διπλωματικη Εργασια

της

Θεοδώρας Κοντογιάννη

Επιβλέπων: Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Ιουλίου 2009.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2009

(Υπογραφή)

.....

Θεοδωρα Κοντογιαννη

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Σελλή για το ενδιαφέρον που έδειξε στην εκπόνηση αυτής της διπλωματικής εργασίας και για την πολύτιμη βοήθεια που μου έχει προσφέρει. Επίσης θα ήθελα να ευχαριστήσω τους συνεπιβλέποντες της εργασίας Θοδωρή Δαλαμάγκα, Παναγιώτη Μπούρο και Σπύρο Σκιαδόπουλο για τον χρόνο που αφιέρωσαν και τις εποικοδομητικές συμβουλές τους. Ευχαριστώ επίσης τον φίλο μου Ορέστη Βάντζο για την συμπαράσταση και τις παρατηρήσεις του σχετικά με την εργασία αυτή. Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Περίληψη

Πολλές εφαρμογές σε περιοχές όπως η Βιοχημεία και τα Γεωγραφικά Πληροφοριακά Συστήματα (GIS) περιλαμβάνουν την αποθήκευση και αποτίμηση ερωτημάτων σε μεγάλο όγκο δεδομένων που έχουν την μορφή ακολουθιών αντικειμένων. Στην παρούσα διπλωματική εργασία τα δεδομένα αυτά ονομάζονται Συλλογή Μονοπατιών, όπου *μονοπάτι* ορίζεται μία ακολουθία κόμβων. Στόχος της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και υλοποίηση μηχανισμων για αποδοτική κατασκευή και ενημέρωση ευρετηρίων στη Συλλογή Μονοπατιών για μεγάλο όγκο δεδομένων που δεν είναι δυνατόν να αποθηκευθούν εξολοκλήρου στην κύρια μνήμη λόγω του περιορισμένου χώρου της. Για αυτό απαιτείται η τμηματική κατασκευή και αποθήκευση σε δευτερεύον αποθηκευτικό μέσο. Πειραματικά δεδομένα αποδεικνύουν την εφαρμοσιμότητα των μεθόδων μας.

Λέξεις Κλειδιά

μονοπάτι, Συλλογή Μονοπατιών, κόμβος, ευρετήριο, P-Index, L-Index

Abstract

Several applications in areas such as Biochemistry and Geographic Information Systems (GIS) involve storing and querying large volumes of sequential data stored as sequences of objects. In this thesis we refer to such data as Path Collections, where by *path* we refer to a sequence of nodes. The purpose of this thesis is the design and implementation of mechanisms for the efficient storing and updating of index structures on Path Collections which contain large volumes of data that can not be stored in their entirety in the main memory, due to its limited space. Therefore a piecewise construction is required, as well as the transfer to a secondary storage medium. Experimental data prove the applicability of our methods.

Keywords

path, path collections, index, P-Index, L-Index

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

Κεφάλαιο 1

Εισαγωγή

1.1 Αποτίμηση ερωτημάτων σε ακολουθίες αντικειμένων

Η οργάνωση, αποθήκευση και αποτίμηση ερωτημάτων σε δεδομένα μεγάλου όγκου που παρουσιάζονται υπό μορφή ακολουθιών αντικειμένων συναντάται σε πολλές εφαρμογές. Για παράδειγμα τα δίκτυα μεταβολισμού στις βιοχημικές εφαρμογές έχουν να κάνουν με μεγάλες συλλογές από μονοπάτια δηλαδή σειρές από χημικές αντιδράσεις που συμβαίνουν μέσα σε ένα κύτταρο. Άλλο ένα παράδειγμα είναι η συλλογές μονοπατιών που εμφανίζονται σε διαδικτυακές τοποθεσίες όπως το ShareMyRoots.com ή TravelByGPS.com που αρχειοθετούν δημοφιλείς τουριστικές διαδρομές δηλαδή ακολουθίες από σημεία ενδιαφέροντος ή τοποθεσίες που καταχωρούνται από χρήστες.

Κοινό σημείο αυτών των εφαρμογών είναι ότι διαχειρίζονται τα δεδομένα τους σαν ακολουθίες αντικειμένων. Στην παρούσα διπλωματική ονομάζουμε τα αντικείμενα *κόμβους* και μία ακολουθία αντικειμένων *μονοπάτι*. Ένα σύνολο τέτοιων μονοπατιών ορίζει μία *συλλογή μονοπατιών* (ΣΜ). Η διπλωματική εργασία της Λ. Μπούλα [1] και το άρθρο [2] ασχολούνται με την αποτίμηση ερωτημάτων σε μία ΣΜ (ερωτήματα προσβασιμότητας, εύρεσης μονοπατιού). Με σκοπό την αποδοτική αποτίμηση τέτοιων ερωτημάτων ορίστηκαν ευρετήρια που αποθηκεύουν και επιτρέπουν τη γρήγορη πρόσβαση στα μονοπάτια μία συλλογής.

1.2 Αντικείμενο της διπλωματικής

Κατά κανόνα οι ΣΜ είναι πολύ μεγάλες ώστε να αποθηκεύονται στην κύρια μνήμη. Στη παρούσα διπλωματική εργασία θεωρούμε ότι και τα ευρετήρια μίας ΣΜ δεν είναι δυνατόν να κατασκευαστούν εξ ολοκλήρου στην κύρια μνήμη. Κατά συνέπεια, στόχος της εργασίας είναι ο σχεδιασμός μηχανισμών για την αποδοτική κατασκευή και ενημέρωση των ευρετηρίων της συλλογής. Ο διαθέσιμος χώρος της κύριας μνήμης είναι περιορισμένος οπότε οργανώσαμε την κατασκευή των ευρετηρίων ώστε να γίνεται τμηματικά. Κάθε φορά που ο διαθέσιμος χώρος στην κύρια μνήμη γεμίζει, ενώνουμε το κομμάτι του ευρετηρίου στη μνήμη με αυτό στο σκληρό δίσκο και το μεταφέρουμε στο δίσκο, αδειάζοντας το χώρο στη μνήμη. Η διαδικασία

επαναλαμβάνεται μέχρι ολόκληρο το ευρετήριο να αποθηκευτεί στο σκληρό δίσκο. Επίσης προσεγγίζουμε το θέμα της συντήρησης των ευρετηρίων με διαδοχικές ενημερώσεις - προσθήκες νέων μονοπατιών. Για την ενημέρωση των ευρετηρίων υποθέτουμε, επίσης, ότι ο χώρος στην κύρια μνήμη είναι περιορισμένος. Ο κύριος στόχος αυτής της διπλωματικής εργασίας είναι ακριβώς η αποδοτική κατασκευή και ενημέρωση των ευρετηρίων για πολύ μεγάλου όγκου ΣΜ.

1.2.1 Τα ευρετήρια *P-Index* και *L-Index*

Μελετούμε κυρίως δύο είδη ευρετηρίων. Το ευρετήριο *P-Index* (*Path index*) αποθηκεύει σε κάθε εγγραφή του, για ένα συγκεκριμένο κόμβο στη ΣΜ τα μονοπάτια στα οποία ανήκει καθώς και τη θέση του σε αυτά. Στο ευρετήριο *L-Index* (*Link index*) αποθηκεύονται σε κάθε εγγραφή, για ένα συγκεκριμένο μονοπάτι οι κοινοί του κόμβοι με άλλα μονοπάτια (κόμβοι-σύνδεσμοι) και η θέση τους στο μονοπάτι. Δηλαδή για κάθε κόμβο v_i του ευρετηρίου *P-Index* και κάθε μονοπάτι p_j του ευρετηρίου *L-Index* αποθηκεύεται το *ζευγάρι μονοπάτι-θέση* (p_j, pos_{ij}) και το *ζευγάρι κόμβος-θέση* (v_i, pos_{ij}) αντίστοιχα.

1.2.2 Διαχείριση κύριας μνήμης

Υποθέσαμε ότι η διαθέσιμη κύρια μνήμη είναι περιορισμένη. Για την κατασκευή και ενημέρωση των ευρετηρίων οργανώνουμε την κύρια μνήμη σε ένα σύνολο από *τμήματα* (blocks), το κάθε ένα από αυτά με δικό του *αναγνωριστικό* (block_id). Κάθε τμήμα χωράει συγκεκριμένο προκαθορισμένο αριθμό ζευγαριών. Τα ζευγάρια αυτά μπορεί να είναι είτε μονοπάτι-θέση είτε κόμβος-θέση, ανάλογα με το αν αφορούν τον *P-Index* ή τον *L-Index* αντίστοιχα. Κάθε τμήμα συνδέεται, μέσω του αναγνωριστικού του, με την εγγραφή ένος κόμβου του *P-Index* ή ενός μονοπατιού του *L-Index* και αποθηκεύει τα αντίστοιχα ζευγάρια.

1.2.3 Κατασκευή των ευρετηρίων με χρήση περιορισμένης μνήμης

Το ευρετήριο *P-Index* κατασκευάζεται από μία συλλογή μονοπατιών ΣΜ αποθηκευμένη σε αρχείο. Για κάθε κόμβο v_i που βρίσκεται σε μονοπάτι p_j της ΣΜ στην θέση pos_{ij}, προσθέτουμε στην εγγραφή *P-Index*$[v_i]$ του ευρετήριου το ζευγάρι μονοπάτι-θέση (p_j, pos_{ij}). Το ευρετήριο *L-Index* κατασκευάζεται με βάση το *P-Index*. Για κάθε κόμβο v_i του *P-Index* που περιέχεται σε περισσότερα του ενός μονοπάτια $\{p_{j_1}, \ldots\}$, προσθέτουμε στις αντίστοιχες εγγραφές $\{$*L-Index*$[p_{j_1}], \ldots\}$ του ευρετήριου το ζευγάρι κόμβος-θέση (v_i, pos_{ij_1}).

Κατά την προσθήκη ενός ζευγαριού στην εγγραφή ενός ευρετηρίου (*P-Index* ή *L-Index*) ακολουθούμε την εξής διαδικασία:

- Αρχικά ελέγχουμε εαν έχει ανατεθεί κάποιο τμήμα μνήμης στην εγγραφή και κατά πόσο το τμήμα αυτό είναι γεμάτο.

- Εφόσον η εγγραφή δεν διαθέτει κάποιο τμήμα που να μπορεί να αποθηκεύσει το ζευγάρι, επιχειρούμε να δεσμεύσουμε ένα από τα ελεύθερα τμήματα.

- Αν δεν έχουν απομείνει ελεύθερα τμήματα, το τμήμα του ευρετηρίου που έχει ήδη κατασκευαστεί μεταφέρεται σε δευτερεύον αποθηκευτικό μέσο (σκληρός δίσκος) και τα τμήματα μνήμης απελευθερώνονται.
- Σε κάθε περίπτωση, η εγγραφή καταλήγει με ένα τμήμα μνήμης που μπορεί να δεχθεί το ζευγάρι και επομένως η προσθήκη μπορεί να ολοκληρωθεί επιτυχώς.

Με το πέρας της προσθήκης όλων των ζευγαριών της συλλογής μονοπατιών ΣΜ στο ευρετήριο, μεταφέρουμε τα εναπομείναντα δεδομένα από την μνήμη στο δίσκο.

1.2.4 Ενημέρωση των ευρετηρίων με χρήση περιορισμένης μνήμης

Στα πλαίσια αυτής της διπλωματικής εργασίας εξετάζουμε μόνο την προσθήκη νέων μονοπατιών στη συλλογή.Με δεδομένο ότι οι ΣΜ είναι αποθηκευμένες στο σκληρό δίσκο, επικεντρωνόμαστε στην διαδικασία της ενημέρωσης των ευρετηρίων με ομάδες νέων μονοπατιών και όχι για ένα μονοπάτι κάθε φορά. Η διαδικασία για την ενημέρωση των ευρετηρίων περιλάμβανει δύο βασικά βήματα:

1. Κατασκευή ευρετηρίων *P-Index*$_{mem}$ και *L-Index*$_{mem}$ στην μνήμη με βάση την συλλογή νέων μονοπατιών.
2. Συνένωση (merge) των ευρετηρίων στην μνήμη με τα αντιστοιχα *P-Index*$_{disk}$ και *L-Index*$_{disk}$ στον σκληρό δίσκο, τα οποία περιγράφουν την αρχική συλλογή μονοπατιών.

Η ενημέρωση των ευρετηρίων με χρήση περιορισμένης μνήμης ακολουθεί σε γενικά πλαίσια την ίδια λογική με τον αλγόριθμο κατασκευής της προηγούμενης ενότητας. Η ενημέρωση του ευρετηρίου *P-Index* από ένα αρχείο ΣΜ με νέα μονοπάτια γίνεται με την σταδιακή κατασκευή του ευρετηρίου *P-Index* των ενημερώσεων στη μνήμη. Προσθέτουμε τα νέα ζευγάρια μονοπάτι-θέση στις αντίστοιχες εγγραφές του *P-Index*, δεσμεύοντας νέα τμήματα μνήμης και αποθηκεύοντας το ευρετήριο στον δίσκο όταν χρειάζεται.

Η ενημέρωση του *L-Index* χρησιμοποιεί το ήδη ενημερωμένο ευρετήριο *P-Index* που βρίσκεται αποθηκευμένο πια σε δευτερεύον αποθηκευτικό μέσο. Για κάθε νέο μονοπάτι ελέγχουμε αν οι κόμβοι του βρίσκονται σε περισσότερο του ενός μονοπάτια. Αν ναι, τότε ο κόμβος είναι σύνδεσμος και τον προσθέτουμε στον *L-Index* στη μνήμη. Επιπλέον, ελέγχουμε σε πόσα παλιά μονοπάτια (στον *P-Index*) περιλαμβάνεται ο κόμβος. Αν περιλαμβάνεται σε ένα μόνο παλιό μονοπάτι, ο κόμβος έγινε συνδεσμος λόγω των ενημερώσεων και επομένως πρέπει να ενημερώσουμε και την εγγραφή του *L-Index* για το παλιό μονοπάτι.

1.2.5 Συνεισφορά

Η παρούσα διπλωματική εργασία συνεισφέρει, όπως είπαμε, στην αποδοτική κατασκευή και ενημέρωση δύο ευρετηρίων σε συλλογές μονοπατιών. Τα ευρετήρια αυτά έχουν προταθεί σε προηγούμενες ή παράλληλες χρονικά, εργασίες για την αποδοτική αποτίμηση ερωτημάτων προσβασιμότητας και εύρεσης μονοπατιού μεταξύ δύο κόμβων. Πραγματοποιήσαμε επιπλέον, στα πλαίσια της εργασίας, πειράματα κατασκευής και ενημέρωσης με τεχνητές ΣΜ.

1.3 Οργάνωση κειμένου

Το περιεχόμενο της διπλωματικής εργασίας οργανώνεται ως εξής: στο παρών κεφάλαιο παρουσιάζεται μία συνοπτική περιγραφή του θέματος της διπλωματικής εργασίας. Στο κεφάλαιο 2 αναφέρονται οι σχετικές με το αντικείμενο εργασίες που υπάρχουν. Στο κεφάλαιο 3 δίνονται οι ορισμοί εννοιών που θα χρησιμοποιηθούν στη διπλωματική για την περιγραφή του προβλήματος και των αλγόριθμων και η τυπική περιγραφη του προβλήματος. Οι αλγοριθμοι για την λύση του και παραδείγματα τους περιέχονται στο κεφαλαιο 4. Στο κεφαλαιο 5 παρουσιάζονται τα πειράματα για την αξιολόγηση των αλγορίθμων που υλοποιήθηκαν. Στο κεφαλαιο 6 αναφέρονται οι προγραμματιστικές δομές και κλάσεις που χρησιμοποιήθηκαν κατα την υλοποίηση. Στο κεφαλαιο 7 παρουσιάζονται τα συμπεράσματα της διπλωματικής εργασίας και στο κεφ 8 δίνεται η βιβλιογραφια στην οποία ανατρέξαμε για την εκπόνηση της διπλωματικης εργασιας.

Κεφάλαιο 2

Σχετικές Εργασίες

Οι σχετικές με το αντικείμενο της παρούσας διπλωματικής εργασίας εργασίες αφορούν την αποτίμηση ερωτημάτων πάνω σε συλλογές μονοπατιών με τη χρήση ευρετηρίων.

2.1 Αποτίμηση ερωτημάτων πάνω σε δεδομένα γράφων

Διπλωματική Εργασία Λ.Μπούλα [1]

Η διπλωματική εργασία έχει ως αντικείμενο ερωτήματα πάνω σε δεδομένα που είναι οργανωμένα σε γράφους και όχι σε ΣΜ όπως η παρούσα διπλωματική εργασία. Γίνεται χρήση κάποιου ευρετηρίου για την αποτίμηση ερωτημάτων αλλά δεν εμβαθύνει στην κατασκευή ευρετηρίων .

Η διπλωματική αυτή εργασία έχει ως στόχο να μειώσει τα βήματα των αλγόριθμων που χρησιμοποιούνται για την αποτίμηση ερωτημάτων σε γράφους. Για να το κάνει αυτό προτείνει να εξετάζονται σε κάθε βήμα όχι μόνο τα παιδία ενός κόμβου αλλά οι απόγονοι του. Ορίζεται η ΣΜ αλλά μόνο ατο πλαίσιο ενός γράφου και όχι ανεξάρτητα. Ορίζει ένα ευρετήριο PATHINDEX με επιπλέον πληροφορία για τους κόμβους ενός γράφου τα μονοπάτια στα οποία ανήκουν τους απογόνους τους κτλ. Διαφέρει ανάλογα με τις πληροφορίες που χρειάζεται κάθε αλγόριθμος.

Ασχολείται με τα εξής ερωτήματα:

α) Δίνεται ένας κόμβος A και ζητούνται όλα τα παιδιά του.

β) Δίνονται δύο κόμβοι A και B και ζητείται να βρεθεί αν υπάρχει μονοπάτι μεταξύ τους και αν ναι ποιό.

γ) Δίνονται δύο κόμβοι A και B και ζητείται να βρεθεί αν υπάρχει μονοπάτι ανάμεσα σε αυτούς που να ικανοποιεί κάποιο συγκεκριμένο περιορισμό, για παράδειγμα να έχει μήκος το πολύ 100 κόμβους (και αν ναι, ποιό είναι αυτό).

Η γενική ιδέα των αλγόριθμων που χρησιμοποιήθηκαν είναι η εξής:

- Υπάρχει ένα σύνολο κόμβων προς μελέτη, το οποίο ονομάζουμε μέτωπο αναζήτησης. Το μέτωπο αναζήτησης περιέχει αρχικά τον κόμβο A.

- Σε κάθε επανάληψη αφαιρείται ένας κόμβος από το μέτωπο αναζήτησης και ελέγχεται αν πρόκειται για τον κόμβο B.
  - *Αν ναι*, τότε , είτε υπάρχει θετική απάντηση (για το ερώτημα β), είτε ελέγχεται αν ικανοποιείται ο περιορισμός που έχει τεθεί (ερώτημα γ).
  - *Αν όχι*, προστίθενται στο μέτωπο αναζήτησης τα κομμάτια των μονοπατιών που ξεκινούν από τον κόμβο αυτό και μετά.

Υπάρχουν διάφορες εκδοχές αυτής της κεντρικής ιδέας. Για το ερώτημα β) υπάρχουν οι εξής εκδοχές :

- Αλγόριθμοι που τοποθετούν τα κομμάτια των μονοπατιών στο μέτωπο αναζήτησης, με δύο υποεκδοχές:
  - Αλγόριθμοι των οποίων το μέτωπο αναζήτησης είναι μία *ουρά* (queue), και λέγονται bfs-1.
  - Αλγόριθμοι των οποίων το μέτωπο αναζήτησης είναι μία *στοίβα* (stack), και λέγονται dfs-1.
- Αλγόριθμοι που τοποθετούν χωριστά τους κόμβους κάθε μονοπατιού στο μέτωπο αναζήτησης με υποεκδοχές:
  - Αλγόριθμοι των οποίων το μέτωπο αναζήτησης είναι μία *ουρά* (queue), και λέγονται bfs-1.
  - Αλγόριθμοι των οποίων το μέτωπο αναζήτησης είναι μία *στοίβα* (stack), και λέγονται dfs-1.

Για το ερώτημα γ) οι αλγόριθμοι ανήκουν σε μία από τις δύο εκδοχές που αναφέρθηκαν πιο πάνω, αλλά χρησιμοποιούνται δύο δομές, μια στην οποία αποθηκεύεται το μονοπάτι που κατασκευάζεται και μία στοίβα με τους προς εξέταση κόμβους. Οι αλγόριθμοι αυτοί είναι παραλλαγές των dfs-1 αλγορίθμων, εμπλουτισμένες με τους απαραίτητους ελέγχους για το μήκος του ζητούμενου μονοπατιού.

Υλοποιήθηκαν πειράματα για σύγκριση με τους ήδη υπάρχοντες γνωστούς αλγόριθμούς στην βιβλιογραφία.

2.2 Αποτίμηση ερωτημάτων σε συλλογές μονοπατιών

Evaluating reachability queries over path collections [2]

Αυτή η εργασία επικεντρώνεται στην απάντηση ερωτημάτων προσβασιμότητας. Δεδομένης μίας συλλογής μονοπατιών και δύο κόμβων v_s, v_t διαπιστώνεται αν υπάρχει μονοπάτι από το v_s στο v_t και αν ναι, ποιό είναι αυτό. Τα δεδομένα σε αυτή την εργασία είναι σε μορφή συλλογών μονοπατιών, όπως και στην παρούσα διπλωματική εργασία, και γίνεται χρήση του ευρετηρίου *P-Index* και ενός ακόμα του ευρετηρίου *H-Index*, ενός ευρετηρίου πολύ μεγαλύτερο σε όγκο δεδομένων από τον *L-Index* που χρησιμοποιείται στην παρούσα διπλωματική εργασία. Αλλά

όπως και η προηγούμενη εργασία έτσι και αυτό το αρθρο επικεντρώνεται στην αποτίμηση ερωτημάτων με τη βοήθεια ευρετηρίων και όχι στην αποδοτική κατασκευή και ενημέρωση τους.

Για την αποτίμηση ερωτημάτων σε σχέση με την προηγούμενη σχετική εργασία προτείνεται ο αλγόριθμος **pfs** (*path first search*). Η βασική ιδέα για τον αλγόριθμο pfs είναι ότι εξετάζει ολόκληρα μονοπάτια αντί για μεμονωμένους κόμβους. Ο αλγόριθμος παίρνει μια συλλογή μονοπατιών P, την αφετηρία v_s και τον κόμβο στόχο v_t σαν εισόδους και επιστρέφει το μονοπάτι που τους συνδέει, αν φυσικά υπάρχει.

Ο αλγόριθμος χρησιμοποιεί τις εξής δομές δεδομένων:

1. την στοίβα αναζήτησης Q,
2. το σύνολο H που περιλαμβάνει όλους τους κόμβους που έχουν τοποθετηθει στη Q και
3. το σετ προγόνων A που αποθηκεύει τους άμεσους προγόνους κάθε κόμβου στο Q

Ο pfs αλγόριθμος λειτουργεί παρόμοια με την κατά βάθος αναζήτηση.Αρχικά η στοίβα Q και H περιέχει τον κόμβο v_s. Μετά η εγγραφή $(v_s, empty)$ εισάγεται στο A χαρακτηρίζοντας τον κόμβο v_s κόμβο αφετηρία. Ο αλγόριθμος προχωράει εξετάζοντας τα περιεχόμενα της στοίβας . Ο τρέχον κόμβος στην κορυφή v_n ανακτάται από την Q και ελέγχεται αν είναι ο στόχος v_t . Αν ναι η αναζήτηση σταματάει και με τη χρήση της A κατασκευάζεται το μονοπάτι μέχρι την αφετηρία v_s. Αν δεν βρεθεί ο αλγόριθμος ελέγχει όλα τα μονοπάτια που περιέχουν το v_n. Για κάθε τέτοιο μονοπάτι p με κόμβο v_p μετά τον v_n ελέγχεται αν ο v_p δεν έχει τοποθετηθεί ποτέ στην Q. Αν είναι έτσι τοποθετείται στη Q και στην H. Επίσης μία εγγραφή (v_p, άμεσος πρόγονος του v_p στο p) τοποθετείται στο A. Τελικά ο v_p αντικαθίσταται με τον επόμενο κόμβο στο p και η επανάληψη συνεχίζεται.

Προτείνεται και ο αλγόριθμος **pfsP** που είναι μία εξέλιξη του προηγούμενου με βάση τον *P-Index*. Αφού το ευρετήριο του επιτρέπει να αναγνωρίζει αμέσως τα μονοπάτια στα οποία περιέχεται ο κόμβος v_n από τη λίστα $\mu o \nu o \pi \acute{\alpha} \tau \iota \alpha[v_n]$. Επίσης προσφέρει μία γρήγορη συνθήκη τερματισμού. Για παράδειγμα, αν ο κόμβος v_n ανακτάται από την Q, η αναζήτηση μπορεί να τερματιστεί αν υπάρχει μονοπάτι p_c στη συλλογή που περιέχει τους κόμβους v_n και v_t με τον v_t να διαδέχεται τον v_n.

Ακόμα προτείνεται ο αλγόριθμος **pfsH** που κάνει χρήση του ευρετηρίου *H-Index* για τον *H-graph*, που αποθηκεύει για κάθε μονοπάτι τις διασταυρώσεις του με άλλα μονοπάτια και μέσω ποιών κόμβων. Ο αλγόριθμος pfsH είναι παρόμοιος με τον pfs αλλά έχει δύο νέες συνθήκες τερματισμού. Στην αρχή του αλγόριθμου ελέγχεται η ύπαρξη μονοπατιού p που να περιέχει τον κόμβο v_s πριν τον v_t με χρήση του *P-Index*. Όταν εξετάζεται ένα νέο μονοπάτι p_i που περιέχει τον τρέχον κόμβο αναζήτησης v_n στη θέση o_{ni}, ο pfsH ελέγχει αν υπάρχει ακμή (p_i, p_j, u_k) του *H-graph* που να ικανοποιεί τις παραπάνω συνθήκες, εξετάζοντας τις λίστες $\alpha \kappa \mu \acute{\epsilon} \varsigma[p_i]$ και $\mu o \nu o \pi \acute{\alpha} \tau \iota \alpha[v_t]$ του *H-Index*$[P]$ και *P-Index*$[P]$.

Κεφάλαιο 3

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζουμε τους απαραίτητους ορισμούς για την κατανόηση της διπλωματικής εργασίας. Ειναι οι ορισμοι ευρετηρίων η κατασκευή των οποίων εχει ος κύριο στόχο την αποδοτικότερη πρόσβαση σε μεγάλο όγκο δεδομένων.

3.1 Ορισμός Συλλογής Μονοπατιών και των ευρετηρίων της

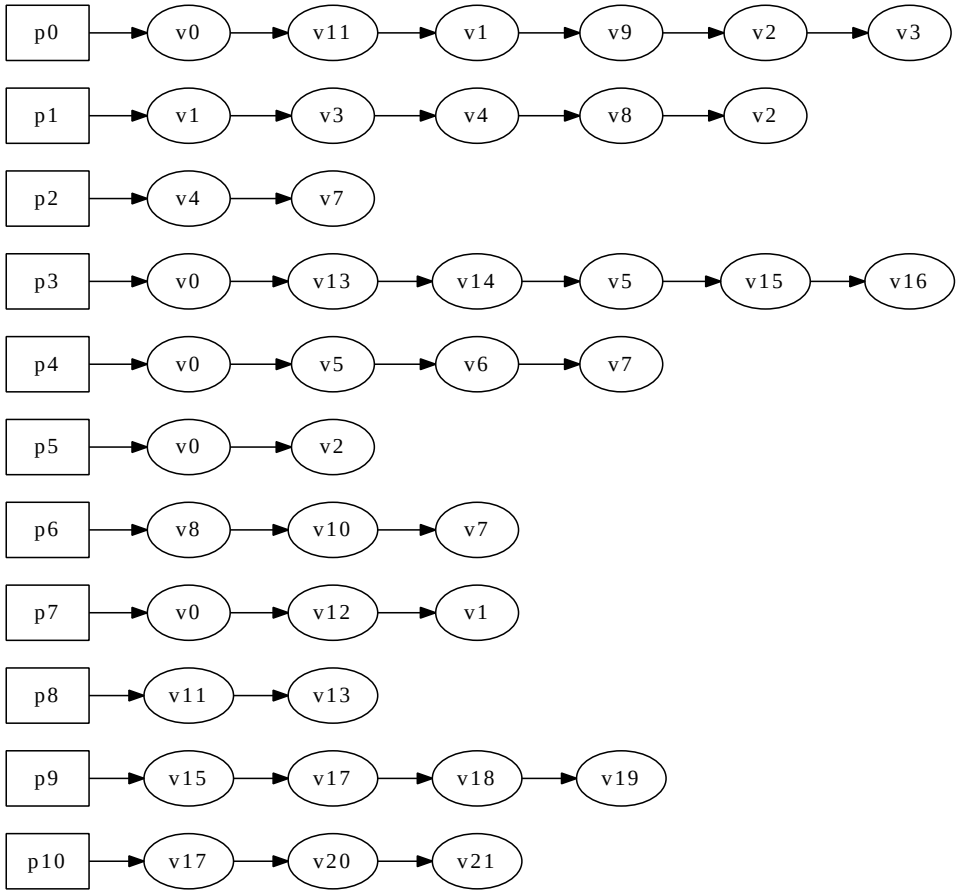
Η αναπαράσταση των δεδομένων μας γίνεται με τη μορφή Συλλογής Μονοπατιών(ΣΜ). Η Συλλογή Μονοπατιών αποτελεί ένα σύνολο από ακολουθίες αντικειμένων. Τα αντικείμενα αυτά τα ονομάζουμε κόμβους και μια ακολουθία κόμβων μονοπάτι. Για την απάντηση ερωτημάτων πάνω σε αυτά τα δεδομένα είναι απαραίτητη η κατασκευή διαφόρων ευρετηριών που προσφέρουν αποδοτικότερη πρόσβαση στα δεδομένα από τη συλλογή μονοπατιών.Περιγραφές και παραδείγματα αυτών των ευρετηριών ακολουθούν σε αυτή την ενότητα κάθως και σχόλια για την αποτελεσματικότητα του καθενός.

Ορισμός 3.1 (Μονοπάτι). *Εστω V ένα σύνολο από κόμβους. Ενα μονοπάτι $p(v_1, \ldots, v_k)$ πάνω στο V είναι μία ακολουθία από διακριτούς κόμβους $(v_1, \ldots, v_k) \in V$. Ως κόμβοι(p) υποδηλώνεται το σύνολο των κόμβων στο p. Το μήκος της διαδρομής p, που δηλώνεται από το l_p, είναι ο αριθμός των κόμβων του, αυτό σημαίνει $l_p = |$κόμβοι$(p)|$.*

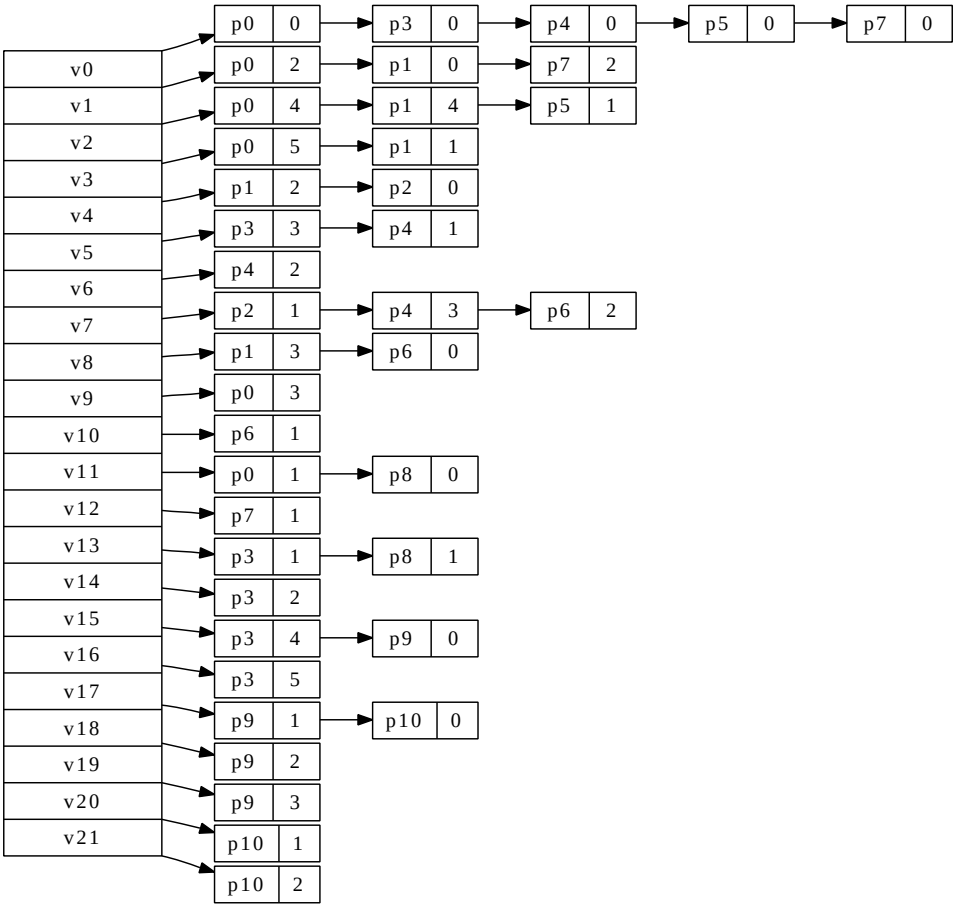
Ορισμός 3.2 (Συλλογή Μονοπατιών). *Εστω V ένα σύνολο από κόμβους. Μία συλλογή μονοπατιών στο V, που ορίζεται ως P, είναι το σύνολο των διαδρομών $\{p_1, \ldots, p_m\}$ στο V. Με τον όρο κόμβοι(P) δηλώνεται το σύνολο των κόμβων στο P.*

Παράδειγμα 3.1. *Μία πιθανή αναπαράσταση μιας συλλογής μονοπατιών είναι το σχήμα 3.1. Έχουμε συνολικά 11 μονοπάτια με αρίθμηση από το 0-10.*

Στον επόμενο ορισμό περιγράφουμε μία δομή, ένα ευρετήριο στην συλλογή μονοπατιών. Τη δομή αυτή την ονομάσαμε *P-Index*. Ο *P-Index* είναι ένα ανεστραμμένο ευρετήριο στη συλλογή μονοπατιών το οποίο αποθηκεύει για κάθε κόμβο στη συλλογή μονοπατιών τα μονοπατια στα οποία ανήκει καθώς και τη θέση του σε αυτά.



Σχήμα 3.1: Παράδειγμα συλλογής μονοπατιών.



Σχήμα 3.2: Παράδειγμα *P-Index*.

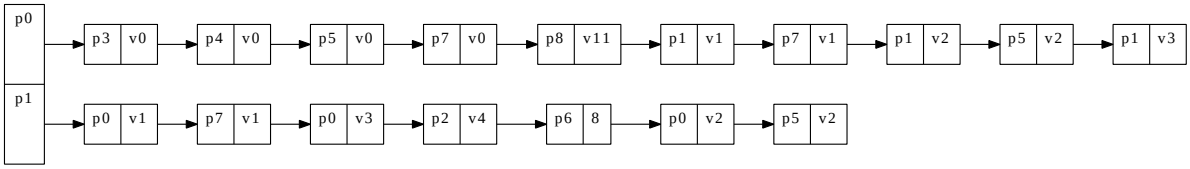
Ορισμός 3.3 (*P-Index*). *Το ευρετήριο της συλλογής μονοπατιών P, που ορίζεται ώς P-Index(P), περιλαμβάνει λίστες μονοπατιών για όλους τους κόμβους στο P. Η λίστα κόμβοι$[v_i]$ για κόμβο v_i περιέχει εγγραφές της μορφής (p_j, o_{ij}), όπου o_{ij} δείχνει τη θέση του v_i στο p_j, για όλα τα μονοπάτια που περιέχουν τον v_i. Οι λίστες των μονοπατιών είναι ταξινομημένες με βάση το αναγνωριστικό του μονοπατιού p_j.*

Παράδειγμα 3.2. *Ο P-Index για την συλλογή μονοπατιών του σχήματος 3.1 φαίνεται στο σχήμα 3.2.*

Στον επόμενο ορισμό περιγράφουμε μια δομή που αποθηκεύει επιπλέον πληροφορίες που βοηθούν στην απάντηση ερωτημάτων. Η δομή αυτή ονομάζεται H-Graph και το ευρετήριο σε αυτή τη δομή H-Index. Ο H-Index αποθηκεύει για καθε μονοπάτι τους κόμβους που οδηγούν σε άλλα μονοπάτια και ποια είναι αυτα.

Ορισμός 3.4 (H-Graph). *Εστω $P = p_1, ..., p_n$ μια συλλογή μονοπατιών. Ο H-Graph του P, συμβ. H-Graph(P), είναι ένας κατευθυνόμενος γράφος με ετικέτες (V, E) τέτοιος ώστε*

- *V αποτελείται από όλα τα μονοπάτια στο P*



Σχήμα 3.3: Παράδειγμα H-Index.

- *μία ακμή με ετικέτα* (p_i, p_j, v) *ανοίκει στο* E*, εάν τα μονοπάτια* p_i*,* p_j *έχουν κοινό κόμβο* $v \in$ *κόμβοι*(P)*, ο οποίος ονομάζεται σύνδεσμος, και δεν είναι ουτε ο πρώτος κόμβος του* p_i *ούτε ο τελευταιος του* p_j

Ορισμός 3.5 (H-Index). *Το ευρετήριο H-Graph*(P) *του* P*, που οριζεται ώς H-Index*(P) *περιλαμβάνει τις λίστες ακμων για όλα τα μονοπάτια του* P*. Η λίστα ακμές*$[p_i]$ *για καθε μονοπάτι* p_i *έχει εγγραφές της μορφής* $(p_j, v_k, o_{ki}, o_{kj})$*, για κάθε ακμή* (p_i, p_j, v_k) *του H-Graph*(P)*, όπου* o_{ki} (o_{kj}) *ορίζεται η θέση του κομβου-σύνδεσμου* v_k *στο μονοπάτι* p_i *(αντ.* p_j*). Οι εγγραφές ταξινομούνται κυρίως σύμφωνα με το μονοπάτι* p_j *της εξερχόμενης ακμής και δευτερευόντος σύμφωνα με το* o_{ki}*.*

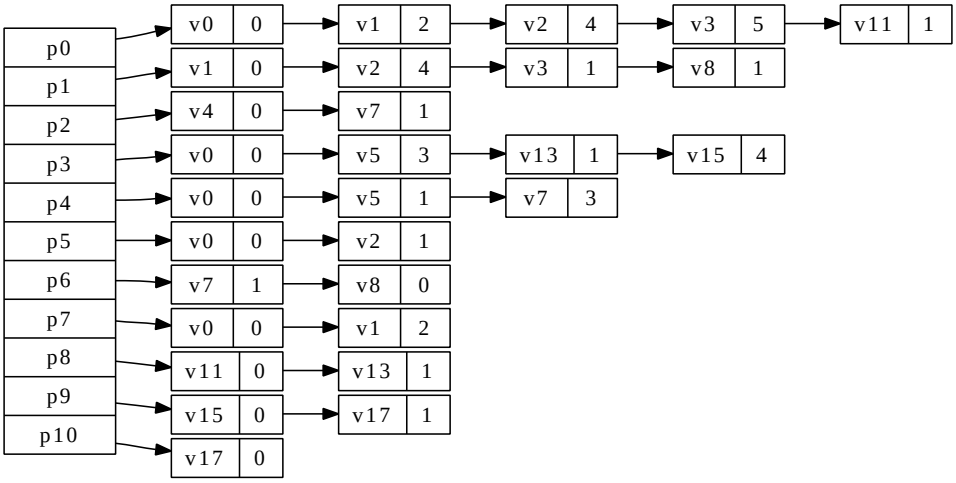
Παράδειγμα 3.3. *Οι εγγραφές του H-Index της συλλογή μονοπατιών του σχήματος 3.1 για τα μονοπάτια* p_0*,* p_1 *φαίνονται στο σχήμα 3.3.*

Οπως είναι εμφανές ακόμα και σε αυτό το μικρό παράδειγμα η αποθήκευση όλων των πιθανών μεταβιβάσεων μεταξύ των μονοπατιών μιας συλλογής μονοπατιών και των αντίστοιχων κόμβων-συνδέσμων χρειάζεται πολύ χώρο. Ο χώρος αυτός αυξάνει ραγδαία για μονοπάτια με πολλούς κοινούς κόμβους. Αυτή η παρατηρηση οδήγησε στην σκέψη ότι η αποθήκευση μόνο των κόμβων συνδέσμων απαιτεί πολύ λιγότερο χώρο. Η διαπίστωση αυτή οδήγησε τελικά στον ορισμό του *L-Index* που θα δούμε παρακάτω. Πχ. το μονοπάτι p_0 χρειάζεται δέκα εγγραφές στον H-Index αλλά μόνο πέντε στον *L-Index*.

Στον επόμενο ορισμό περιγράφουμε μία δομή που ονομάσαμε *L-Index*. Ο *L-Index* είναι ένα ευρετήριο στο οποίο αποθηκεύονται για κάθε μονοπάτι στη συλλογή μονοπατιών οι κοινοί κόμβοι με άλλα μονοπάτια (κόμβοι διασταύρωσης) και τη θέση τους σε αυτό.

Ορισμός 3.6 (*L-Index*). *Το ευρετήριο συνδέσμων του* P*, που ορίζεται ώς L-Index*(P)*, αποτελείται από τις λίστες για όλα τα μονοπάτια στο* P*. Η λίστα σύνδεσμοι*$[p_i]$ *για το μονοπάτι* p_i *έχει εγγραφές της μορφής* (v_k, o_{ki})*, για κάθε κόμβο-σύνδεσμο* v_k *που περιέχεται στο μονοπατι* p_i *στη θέση* o_{ki}*.Οι εγγραφές είναι ταξινομημένες με βάση την ταυτότητα συνδέσμου κόμβου.*

Παράδειγμα 3.4. *Στο σχήμα 3.4 απεικονίζεται ο L-Index για της συλλογής μονοπατιών του σχήματος 3.1.*



Σχήμα 3.4: Παράδειγμα *L-Index*.

3.2 Ορισμός Διαδικασιών Κατασκευής και Ενημέρωσης Ευρετηρίων

3.2.1 Διαδικασίες κατασκευής ευρετηρίων

Παρακάτω περιγράφουμε τις γενικές αρχές της διαδικασίας που κατασκευάζει τους *P-Index* και *L-Index*. Αρχικά η κατασκευή γίνεται στη μνήμη και μετά γίνεται η μεταφορά στον δίσκο. Παρουσιάζουμε τον κάθε αλγόριθμο χωρίς τις τεχνικές που χρησιμοποιήσαμε για την αποδοτικότερη κατασκευή των ευρετηρίων. Η απλή κατασκευή του *P-Index* (αλγ.1) ήταν αντικείμενο άλλης εργασίας, αναφέρεται όμως εδώ για λόγους πληρότητας και καλύτερης κατανόησης.

Αλγόριθμος 1: createPathIndex: Κατασκευή ευρετηρίου *P-Index*.

Είσοδοι: αρχείο συλλογή μονοπατιών ΣΜ

Έξοδοι : *P-Index* στη μνήμη, *P-Index* στον δίσκο

Για κάθε *μονοπάτι* p_j *στη συλλογή μονοπατιών ΣΜ* **{**
  Για κάθε *κόμβο* v_i *στο μονοπάτι* p_j *που είναι αποθηκευμένο στον δίσκο* **{**
    $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
    Αποθήκευσε στο *P-Index*$[v_i]$ στη μνήμη το ζευγάρι (p_j, pos_{ij}).
  }
}
Για κάθε *εγγραφή του P-Index στη μνήμη* **{**
  Αποθήκευση στον σκληρό δίσκο.
}

Κατασκευή ευρετηρίου *L-Index*

Δεδομένης μίας συλλογής μονοπατιών P και *P-Index*(P), ο *L-Index*(P) κατασκευάζεται ώς εξής (αλγ.2). Για κάθε κόμβο $v_k \in$ *κόμβοι*(P), αποκτούμε πρόσβαση στο *μονοπάτια*$[v_k]$ (*P-Index*$[v_k]$). Εάν v_k περιέχεται μόνο σε ένα μονοπάτι $|$*κόμβοι*$[v_k]| = 1$, ο κόμβος δεν είναι σύνδεσμος για κανένα μονοπάτι στη συλλογή. Αλλιώς, διασχίζουμε το *μονοπάτια*$[v_k]$ και προσθέτουμε την εγγραφή (v_k, pos_{ki}) στο *σύνδεσμοι*$[p_i]$ (*L-Index*$[p_i]$) για κάθε μονοπάτι p_i που περιέχει σύνδεσμο v_k στη θέση pos_{ki}.

Αλγόριθμος 2: createLinkNodesIndex: Κατασκευή ευρετηρίου *L-Index*.

Είσοδοι: αριθμός κόμβων στη συλλογή μονοπατιών ΣΜ, *P-Index* στον δίσκο
Έξοδοι : *L-Index* στη μνήμη, *L-Index* στον δίσκο

Για κάθε *κόμβο* v_i *στη συλλογή μονοπατιών ΣΜ* **{**
  Εαν |*P-Index*$[v_i]| > 1$ **{**
    Για κάθε p_j *από το μονοπάτια*$[v_i]$ **{**
      $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
      Αποθήκευσε το ζευγάρι (v_i, pos_{ij}) στο *L-Index*$[p_j]$ στη μνήμη.
    }
  }
}
Για κάθε *μονοπάτι στη μνήμη* **{**
  Αποθήκευση *L-Index* από την μνήμη στον σκληρό δίσκο.
}

3.2.2 Διαδικασίες ενημέρωσης ευρετηρίων

Παρακάτω περιγράφουμε τις γενικές αρχές της διαδικασίας που ενημερώνει τους *P-Index* και *L-Index*. Η ενημέρωση περιλαμβάνει την προσθήκη νέων μονοπατιών. Για να συμπεριλάβουμε ένα νέο μονοπάτι p_j στη συλλογή, χρειάζεται

1. η εισαγωγή της εγγραφής (p_j, pos_{ij}) στο *μονοπάτια*$[v_i]$ για κάθε κόμβο v_i που περιέχεται στο p_j (ενημέρωση *P-Index*)

2. η δημιουργία της *σύνδεσμοι*$[p_j]$ λίστας και η ενημέρωση της λίστας συνδέσμων των μονοπατιών που περιέχουν τους κόμβους v_i (ενημέρωση *L-Index*)

Στην πράξη, οι συλλογές μονοπατιών είναι πολύ μεγάλες για να χωρέσουν στην κύρια μνήμη. Για αυτό οι *P-Index* και *L-Index* μιας συλλογής μονοπατιών αποθηκεύονται με τη μορφή αρχείων σε δευτερεύον μέσο αποθήκευσης και συντηρούνται με ενημερώσεις. Συνήθως ενημερώνουμε μία συλλογή με ένα σύνολο νέων μονοπατιών. Προϋπόθεση για την αποδοτική χρήση των αρχείων είναι να αποθηκεύουμε τις λίστες *μονοπάτια* και *σύνδεσμοι* με συνεχή τρόπο στο μέσο αποθήκευσης. Για αυτό δέν μπορούμε να ενημερώνουμε την συλλογή με κάθε μονοπάτι ξεχωριστά. Για αυτό η ενημέρωση γίνεται σε δύο φάσεις. Αρχικά γίνεται η κατασκευή

των ευρετηρίων που αφορούν μόνο πληροφορία από τα νέα μονοπάτια στην μνήμη και έπειτα γίνεται η ένωση με τα αντίστοιχα ευρετήρια στον σκληρό δίσκο. Παρουσιάζουμε τον κάθε αλγόριθμο χωρίς τις τεχνικές που χρησιμοποιήσαμε για την αποδοτικότερη κατασκευή των ευρετηρίων. Η ενημέρωση του *P-Index* περιλαμβάνει και την ενημέρωση με τα νέα δεδομένα της συλλογής μονοπατιών.

Ενημέρωση ευρετηρίου *P-Index*

Με λίγα λογια παρουσιάζουμε την διαδικασία updatePathIndex (αλγ.3) για τη ενημέρωση του ευρετηρίου *P-Index*. Σε αυτή τη διαδικασία μετά την ενημέρωση της συλλογής μονοπατιών για κάθε κόμβο v_i που περιέχεται στη θέση pos_{ij} ενός νέου μονοπατιού p_j προστίθεται στο τέλος του *P-Index*$[v_i]$ η εγγραφή (p_j, pos_{ij}). Η διαδικασία mergePathIndex (αλγ.4)προσθέτει στο τέλος της λίστας $\mu o \nu o \pi \acute{\alpha} \tau \iota \alpha[v_i]$ με τα στοιχεία του *P-Index*$[v_i]$ στο δίσκο τα στοιχεία του *P-Index*$[v_i]$ στη μνήμη για κάθε κόμβο v_i που περιέχεται στα νέα μονοπάτια. Ετσι η λίστα $\mu o \nu o \pi \acute{\alpha} \tau \iota \alpha[v_i]$ αποθηκεύεται συνεχόμενα στο δίσκο.

Αλγόριθμος 3: updatePathIndex: Ενημέρωση ευρετηρίου *P-Index*.

Είσοδοι: αριθμός παλιών μονοπατίων στη συλλογή μονοπατιών ΣΜ, αρχείο ενημερώσεων Ε

Έξοδοι : *P-Index* ενημερώσεων στη μνήμη, ενημερωμένη συλλογή μονοπατιών ΣΜ

$j \leftarrow$ αριθμός παλιών μονοπατιών $+1$
Για κάθε *μονοπάτι* p_j *στο αρχείο* E **{**
  Για κάθε *κόμβο* v_i *στο μονοπάτι* p_j **{**
    $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
    Αποθήκευσε στο *P-Index*$[v_i]$ στη μνήμη το ζευγάρι (p_j, pos_{ij}).
  }
  Αποθήκευσε το μονοπάτι p_j στη συλλογή μονοπατιών ΣΜ στο σκληρό δίσκο.
  $j \leftarrow j + 1$
}

Ενημέρωση ευρετηρίου *L-Index*

Στη διαδικασία updateLinkNodesIndexMem (αλγ.5) λαμβάνουμε υπ'οψην δύο περιπτώσεις για την ενημέρωση του *L-Index*. Για κάθε κόμβο v_i που περιέχεται στα νέα μονοπάτια υπάρχουν τα εξής ενδεχόμενα:

1. Αν ο κομβος v_i είναι καινούριος,
   (α′) αν περιέχεται σε πάνω από ένα νέο μονοπάτι είναι κόμβος-σύνδεσμος στα νέα μονοπάτια οπότε απλά προστίθεται σε αυτα,
   (β′) αν όχι τότε δεν είναι κόμβος-σύνδεσμος.
2. Αν ο κομβος v_i είναι παλιός,

Αλγόριθμος 4: mergePathIndex: Συνένωση ευρετηρίων *P-Index*.

Είσοδοι: $P\text{-}Index_{mem}$ ενημερώσεων στη μνήμη, $P\text{-}Index_{disk}$ στον σκληρό δίσκο
Έξοδοι: ενημερωμένος $P\text{-}Index_{disk}$ στον σκληρό δίσκο

Για κάθε *κόμβο* v_i *στο* $P\text{-}Index_{mem}$ **{**
  Ανάγνωση του $P\text{-}Index_{disk}[v_i]$ από τον δίσκο.
  $\mu ονοπάτια[v_i] \leftarrow P\text{-}Index_{mem}[v_i] \cup P\text{-}Index_{disk}[v_i]$
  Εαν $|P\text{-}Index_{disk}[v_i]| > 0$ **{**
    // Προϋπάρχων κόμβος
    Διαγραφή του $P\text{-}Index_{disk}[v_i]$ από τον δίσκο.
  }
  Εγγραφή του $\mu ονοπάτια[v_i]$ στο $P\text{-}Index_{disk}$.
}

(α′) αν περιέχεται σε περισσότερα του ενός παλιά μονοπάτια προστίθεται ως κόμβος-σύνδεσμος στα νέα μονοπάτια μόνο,

(β′) αν περιέχεται σε ένα μόνο παλιό μονοπάτι πρέπει να προστεθεί ως κόμβος-σύνδεσμος όχι μόνο στα νέα μονοπάτια αλλά και στο παλιό.

Αλγόριθμος 5: updateLinkNodesIndexMem: Ενημέρωση ευρετηρίου *L-Index*.

Είσοδοι: $P\text{-}Index_{mem}$ ενημερώσεων στη μνήμη , (μη ενημερωμένος) $P\text{-}Index_{disk}$ στο δίσκο
Έξοδοι: $L\text{-}Index_{mem}$ ενημερώσεων στη μνήμη

Για κάθε *κόμβο* v_i *του* $P\text{-}Index_{mem}$ **{**
  Εαν $|P\text{-}Index_{mem}[v_i]| > 1$ **ή** $(|P\text{-}Index_{mem}[v_i]| = 1$ **και** $v_i \in P\text{-}Index_{disk})$ **{**
    Για κάθε *μονοπάτι* p_j *στο* $P\text{-}Index_{mem}[v_i]$ **{**
      $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
      Αποθήκευσε στο $L\text{-}Index_{mem}[p_j]$ το ζευγάρι (v_i, pos_{ij}).
    }
  }
  Εαν $v_i \in P\text{-}Index_{disk}$ **και** $|P\text{-}Index_{disk}[v_i]| = 1$ **{**
    // Παλιός κόμβος που έγινε σύνδεσμος λόγω των ενημερώσεων.
    Έστω (p_k, pos_{ik}) το μοναδικό ζευγάρι του $P\text{-}Index_{disk}[v_i]$.
    Αποθήκευσε στο $L\text{-}Index_{mem}[p_k]$ το ζευγάρι (v_i, pos_{ik}).
  }
}

Η διαδικασία mergeLinkNodesIndex (αλγ.6) προσθέτει στο τέλος της λίστας $κόμβοι[p_j]$ με τα στοιχεία του $L\text{-}Index_{disk}[p_j]$ στο δίσκο τα στοιχεία του $L\text{-}Index_{mem}[p_j]$ στη μνήμη για κάθε μονοπάτι p_j που επηρεάζεται από την διαδικασία ενημέρωσης. Ετσι η λίστα $κόμβοι[p_j]$

μετά από ταξινόμηση αποθηκεύεται συνεχόμενα στο δίσκο.

Αλγόριθμος 6: mergeLinkNodesIndex: Συνένωση ευρετηρίων *L-Index*.

Είσοδοι: $L\text{-}Index_{mem}$ ενημερώσεων στη μνήμη , $L\text{-}Index_{disk}$ στο δίσκο

Έξοδοι : ενημερωμένος $L\text{-}Index_{disk}$ στον δίσκο

Για κάθε *μονοπάτι* p_j *του* $L\text{-}Index_{mem}$ **{**
  Διάβασε το $L\text{-}Index_{disk}[p_j]$ από τον δίσκο.
  κόμβοι$[p_j] \leftarrow L\text{-}Index_{mem}[p_j] \cup L\text{-}Index_{disk}[p_j]$
  Ταξινόμηση λίστας *κόμβοι*$[p_j]$.
  Εαν $p_j \in L\text{-}Index_{disk}$ **{**
    `// Το μονοπάτι προϋπήρχε.`
    Διέγραψε το $L\text{-}Index_{disk}[p_j]$ από τον δίσκο.
  }
  Εγγραφή *κόμβοι*$[p_j]$ στο $L\text{-}Index_{disk}$.
}

Κεφάλαιο 4

Τεχνικές για την αποδοτική κατασκευή και ενημέρωση των ευρετηρίων

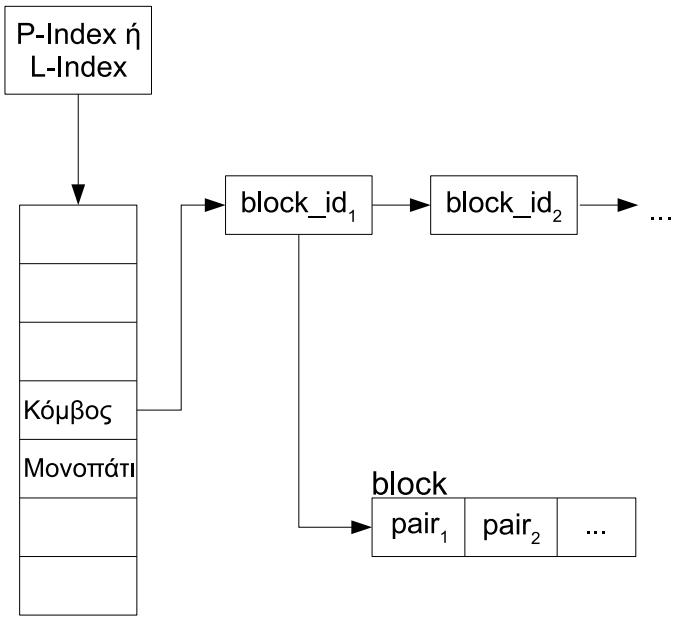
Σε αυτή την ενότητα λύνουμε το πρόβλημα του μεγάλου χώρου σε μνήμη που χρειάζεται για την αποθήκευση των ευρετηρίων *P-Index* και *L-Index*. Το μεγάλο μέγεθος των ευρετηρίων καθιστά αδύνατη την επεξεργασία συλλογών μονοπατιών πέραν ενός συγκεκριμένου μεγέθους. Η κατα τμήματα επεξεργασία τους στην κύρια μνήμη, που προτείνεται σε αυτό το κεφάλαιο, προσφέρει την δυνατότητα μελέτης συλλογών μονοπατιών με απεριόριστο μέγεθος.

4.1 Διαδικασίες κατασκευής ευρετηρίων

Στο προηγούμενο κεφάλαιο κάνουμε την υπόθεση ότι οι συλλογές μονοπατιών είναι πολύ μεγάλες για να χωρέσουν στην κύρια μνήμη κοι για αυτό οι *P-Index* και *L-Index* αποθηκεύονται ως αρχεία σε δευτερεύον αποθηκευτικό μέσο. Ομώς παρόλα αυτά οι *P-Index* και *L-Index* κατασκευάζονται εξόλοκλήρου στην κύρια μνήμη πρίν την αποθήκευση τους στον σκληρό δίσκο.

Λύνουμε αυτή την αντίφαση υποθέτωντας ότι ο διαθέσιμος χώρος στη κύρια μνήμη είναι περιορισμένος με αποτέλεσμα τα ευρετήρια να μην μπορούν να κατασκευαστούν εξόλοκλήρου στην κύρια μνήμη πριν την αποθήκευση τους στο δευτερεύον αποθηκευτικό μέσο. Συνεπώς κατασκευάζουμε τα ευρετήρια *P-Index* και *L-Index* τμηματικά. Κάθε φορά που ο διαθέσιμος χώρος στην κύρια μνήμη γεμίζει ενώνουμε το κομμάτι του ευρετηρίου που κατασκευάσαμε στην κύρια μνήμη με αυτό στο δευτερεύον αποθηκευτικό μέσο και αδειάζουμε τον χώρο που δεσμέυσαμε στην κύρια μνήμη. Η διαδικασία αυτή επαναλαμβάνεται μέχρι ολόκληρο το ευρετήριο να αποθηκευτεί στο δευτερεύον αποθηκευτικό μέσο. Παρακάτω περιγράφονται αναλυτικά οι διαδικασίες κατασκευής των ευρετηρίων *P-Index* και *L-Index* και παρατίθεται ο αλγόριθμος κατασκευής σε ψευδοκώδικα.

Ο χωρισμός της κύριας μνήμης που προτείνουμε είναι ο εξής: Η κύρια μνήμη χωρίζεται σε όσα τμήματα (blocks) χρειαζόμαστε, το κάθε ένα από αυτά με δικό του αναγνωριστικό



Σχήμα 4.1: Χρήση τμημάτων μνήμης για την αποθήκευση ζευγαριών ενός ευρετηρίου.

(block id). Κάθε τμήμα χωράει συγκεκριμένο, καθορισμένο από εμας, αριθμό ζευγαριών. Τα ζευγάρια αυτά μπορεί να είναι είτε μονοπάτι-θέση είτε κόμβος-θέση ανάλογα με το αν αφορούν τον *P-Index* ή τον *L-Index* αντίστοιχα. Καθε (δεσμευμένο) τμήμα συνδέεται μέσω του αναγνωριστικού του με έναν κόμβο του *P-Index* ή μονοπάτι του *L-Index*. Μια παραστατική περιγραφή αυτού που περιγράψαμε δίνει η εικόνα 4.1.

4.1.1 Κατασκευή ευρετηρίου *P-Index* με βάση το αρχείο μονοπατιών με περιορισμένη διάθεση της κύριας μνήμης

Η κατασκευή του ευρετηρίου *P-Index* βασίζεται στην γενική αρχή ότι δεν είναι εφικτή η κατασκευή του *P-Index* εξόλοκληρου στην κύρια μνήμη επειδή ο διαθέσιμος χώρος της είναι περιορισμένος. Για αυτό ο διαθέσιμος αυτός χώρος μοιράζεται σε τμήματα (blocks) τα οποία μπορούν να χωρέσουν συγκεκριμένο αριθμό ζευγών μονοπάτι-θέση στο μονοπάτι (path-pos pair) της μορφής (p_j, pos_{ij}). Στην πορεία της κατασκεύης του *P-Index* στην μνήμη (αλγ.7), κάθε φορά που θέλουμε να προσθέσουμε ένα ζευγάρι μονοπάτι-θέση (p_j, pos_{ij}) στην εγγραφή $P\text{-}Index_{mem}[v_i]$ για τον κόμβο v_i, ελέγχουμε τις εξής περιπτώσεις:

1. *Αν δεν έχει ανατεθεί τμήμα της κύριας μνήμης* (block) στην εγγραφή $P\text{-}Index_{mem}[vi]$, γίνεται έλεγχος αν υπάρχουν διαθέσιμα τμήματα.

   (α′) *Αν δεν υπάρχουν διαθέσιμα τμήματα*, τότε δεν υπάρχει άλλη κύρια μνήμη διαθέσιμη οπότε το τμήμα του *P-Index* που κατασκευάστηκε ήδη στην κύρια μνήμη μεταφέρεται σε δευτερεύον αποθηκευτικό μέσο.

   (β′) *Αν υπάρχουν διαθέσιμα τμήματα*, ένα νέο τμήμα ανατίθεται στην εγγραφή $P\text{-}Index_{mem}[v_i]$ και το ζευγάρι μονοπάτι-θέση (p_j, pos_{ij}) αποθηκεύεται σε αυτό.

2. *Αν έχει ανατεθεί τμήμα μνήμης* στην εγγραφή $P\text{-}Index_{mem}[v_i]$, τότε γίνεται έλεγχος αν υπάρχει χώρος σε αυτό το τμήμα για να αποθηκευθεί το ζευγάρι μονοπάτι-θέση.

   (α′) *Αν υπάρχει χώρος*, τότε απλά αποθηκεύουμε το ζευγάρι μονοπάτι-θέση (p_j, pos_{ij}).

   (β′) *Αν δεν υπάρχει χώρος*, τότε ελέγχουμε εάν υπάρχει διαθέσιμο τμήμα μνήμης για να ανατεθεί στην εγγραφή $P\text{-}Index_{mem}[v_i]$.

      i. *Αν υπάρχει διαθέσιμο νέο τμήμα*, ανατίθεται στην αντίστοιχη εγγραφή της μνήμης $P\text{-}Index_{mem}[v_i]$ και το ζευγάρι μονοπάτι-θέση (p_j, pos_{ij}) αποθηκεύεται σε αυτό.

      ii. *Αν όχι*, δεν υπάρχει άλλη κύρια μνήμη διαθέσιμη οπότε το τμήμα του *P-Index* που κατασκευάστηκε ήδη στην κύρια μνήμη μεταφέρεται σε δευτερεύον αποθηκευτικό μέσο.

Οι εγγραφές $P\text{-}Index_{mem}[v_i]$, οι οποίες περιέχουν ζευγάρια της μορφής (p_j, pos_{ij}), δεν απαιτούν ταξινόμηση με βάση το αναγνωριστικό (id) των μονοπατιών επειδή τα μονοπάτια διαβάζονται όντας ήδη ταξινομημένα.

Παράδειγμα 4.5 (Κατασκευή *P-Index*)**.** *Θεωρούμε την συλλογή μονοπατιών του πίνακα 4.1. Στον πίνακα 4.2 βλέπουμε την κατασκευή του αντίστοιχου P-Index. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.*

Μονοπάτι	Λίστα Κόμβων
p_0	$v_0, v_1, v_2, v_3, v_4, v_5, v_8, v_9$
p_1	v_9, v_4, v_3, v_6
p_2	v_9, v_7, v_6
p_3	v_6, v_7, v_2, v_{10}

Πίνακας 4.1: Συλλογή Μονοπατιών με 4 μονοπάτια και 11 κόμβους.

Τρέχον Μονοπάτι	Τρέχον Κόμβος	Ανάθεση block σε εγγραφή κόμβου	Ανάθεση ζευγαριού σε block
p_0	v_0	$b_0 \rightarrow v_0$	$(p_0, 0) \rightarrow b_0$
p_0	v_1	$b_1 \rightarrow v_1$	$(p_0, 1) \rightarrow b_1$
p_0	v_2	$b_2 \rightarrow v_2$	$(p_0, 2) \rightarrow b_2$
p_0	v_3	$b_3 \rightarrow v_3$	$(p_0, 3) \rightarrow b_3$
p_0	v_4	out of blocks	
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			
p_0	v_4	$b_0 \rightarrow v_4$	$(p_0, 4) \rightarrow b_0$
p_0	v_5	$b_1 \rightarrow v_5$	$(p_0, 5) \rightarrow b_1$
p_0	v_8	$b_2 \rightarrow v_8$	$(p_0, 6) \rightarrow b_2$
p_0	v_9	$b_3 \rightarrow v_9$	$(p_0, 7) \rightarrow b_3$
p_1	v_9		$(p_1, 0) \rightarrow b_3$
p_1	v_4		$(p_1, 1) \rightarrow b_0$
p_1	v_3	out of blocks	
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			
p_1	v_3	$b_0 \rightarrow v_3$	$(p_1, 2) \rightarrow b_0$
p_1	v_6	$b_1 \rightarrow v_6$	$(p_1, 3) \rightarrow b_1$
p_2	v_9	$b_2 \rightarrow v_9$	$(p_2, 0) \rightarrow b_2$
p_2	v_7	$b_3 \rightarrow v_7$	$(p_2, 1) \rightarrow b_3$
p_2	v_6		$(p_1, 3) \rightarrow b_1$
p_3	v_6	out of blocks	
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			
p_3	v_6	$b_0 \rightarrow v_6$	$(p_3, 0) \rightarrow b_0$
p_3	v_7	$b_1 \rightarrow v_7$	$(p_3, 1) \rightarrow b_1$
p_3	v_2	$b_2 \rightarrow v_2$	$(p_3, 2) \rightarrow b_2$
p_3	v_{10}	$b_3 \rightarrow v_{10}$	$(p_3, 3) \rightarrow b_3$
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			

Πίνακας 4.2: Κατασκευή *P-Index* για την συλλογή μονοπατιών του πιν.4.1. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.

Αλγόριθμος 7: createPIndexMemLimited: Κατασκευή ευρετηρίου *P-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: συλλογή μονοπατιών ΣΜ, συλλογή τμημάτων μνήμης Β
Έξοδοι : $P\text{-}Index_{disk}$ στο δίσκο

Αρχικοποίηση (βοηθητικού) $P\text{-}Index_{mem}$ στην μνήμη.
Για κάθε *μονοπάτι* p_j *στην συλλογή μονοπατιών* ΣM **{**
  Για κάθε *κόμβο* v_i *στο μονοπάτι* p_j **{**
    $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
    Αποθήκευσε στο $P\text{-}Index_{mem}[v_i]$ το ζευγάρι (p_j, pos_{ij}), χρησιμοποιώντας την ρουτίνα **push** (αλγ.8).
    `// Η ρουτίνα` **push** `χρησιμοποιεί τα blocks Β και αποθηκεύει κατά καιρούς αυτόματα στο` $P\text{-}Index_{disk}$ `στο δίσκο.`
  }
  Αποθήκευση των εναπομείναντων δεδομένων του $P\text{-}Index_{mem}$ στον δίσκο, χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.9).
  Διαγραφή του $P\text{-}Index_{mem}$.
}

Αλγόριθμος 8: push: Αποθήκευση ζευγαριού μονοπατιού-θέσης στο *P-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: κόμβος v_i, ζευγάρι μονοπατιού-θέσης p, συλλογή τμημάτων μνήμης B
Έξοδοι: ενημερωμένα $P\text{-}Index_{mem}$ και $P\text{-}Index_{disk}$

χρεία_νέου_block ← **ψευδές**
Εαν *δεν έχει ανατεθεί block στην εγγραφή* $P\text{-}Index_{mem}[v_i]$ **{**
  χρεία_νέου_block ← **αληθές**
}
αλλιώς {
  b ← το τελευταίο block που έχει ανατεθεί στην $P\text{-}Index_{mem}[v_i]$
  χρεία_νέου_block ← (b είναι γεμάτο)
}
Εαν όχι *χρεία_νέου_block* **{**
  Προσθήκη του ζευγαριού p στο block b.
}
αλλιώς εαν *υπάρχει διαθέσιμο block* b_{new} *στη συλλογή* B **{**
  Ανάθεση του b_{new} στην $P\text{-}Index_{mem}[v_i]$.
  Προσθήκη του ζευγαριού p στο block b_{new}.
}
αλλιώς {
  `// Τα διαθέσιμα blocks στην συλλογή B έχουν εξαντληθεί.`
  Αποθήκευση των δεδομένων του $P\text{-}Index_{mem}$ στο $P\text{-}Index_{disk}$ στο δίσκο,
  χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.9).
  Απελευθέρωση των blocks της συλλογής B.
  Ανάθεση νέου block b_{new} στην $P\text{-}Index_{mem}[v_i]$.
  Προσθήκη του ζευγαριού p στο block b_{new}.
}

Αλγόριθμος 9: save-to-disk: Μεταφορά του *P-Index* από την μνήμη στον δίσκο.

Είσοδοι: $P\text{-}Index_{mem}$ στη μνήμη, $P\text{-}Index_{disk}$ στο δίσκο
Έξοδοι: ενημερωμένο $P\text{-}Index_{disk}$ στο δίσκο

Για κάθε *κάθε εγγραφή* $P\text{-}Index_{mem}[v_i]$ **{**
  Ανάγνωση της εγγραφής $P\text{-}Index_{disk}[v_i]$ από τον δίσκο.
  $\mu ονοπάτια[v_i] \leftarrow P\text{-}Index_{mem}[v_i] \cup P\text{-}Index_{disk}[v_i]$
  Εαν $|P\text{-}Index_{disk}[v_i]| > 0$ **{**
    `// Προϋπάρχων κόμβος.`
    Διαγραφή της εγγραφής $P\text{-}Index_{disk}[v_i]$ από τον δίσκο.
  }
  Προσθήκη του $\mu ονοπάτια[v_i]$ στο $P\text{-}Index_{disk}$ στον δίσκο.
}

4.1.2 Κατασκευή ευρετηρίου *L-Index* με βάση το ευρετήριο *P-Index* με περιορισμένη διάθεση κύριας μνήμης

Η κατασκευή του ευρετηρίου *L-Index* βασίζεται στην γενική αρχή με την οποία κατασκευάσαμε τον *P-Index* ότι δηλαδή δεν είναι εφικτή η κατασκευή του *L-Index* εξόλοκληρου στην κύρια μνήμη επειδή ο διαθέσιμος χώρος της είναι περιορισμένος. Για αυτό ο διαθέσιμος αυτός χώρος μοιράζεται σε τμήματα (blocks) τα οποία μπορούν να χωρέσουν συγκεκριμένο αριθμό ζευγαριών κόμβος-θέση (v_i, pos_{ij}). Στην πορεία κατασκεύης του *L-Index* (αλγ.10), κάθε φορά που θέλουμε να προσθέσουμε ένα ζευγάρι κόμβος-θέση σε μια εγγραφή $L\text{-}Index_{mem}[p_j]$, για κάποιο μονοπάτι p_j της συλλογής μονοπατιών, ελέγχονται οι εξής περιπτώσεις:

1. *Αν δεν έχει ανατεθεί τμήμα της κύριας μνήμης* στην εγγραφή $L\text{-}Index_{mem}[p_j]$, τότε γίνεται έλεγχος αν υπάρχουν διαθέσιμα τμήματα μνήμης.

   (α′) *Αν δεν υπάρχει άλλη κύρια μνήμη διαθέσιμη*, το τμήμα του *L-Index* που κατασκευάστηκε ήδη στην κύρια μνήμη μεταφέρεται σε δευτερεύον αποθηκευτικό μέσο.

   (β′) *Αν υπάρχει διαθέσιμο block*, ανατίθεται στην εγγραφή $L\text{-}Index_{mem}[p_j]$ και το ζευγάρι κομβος-θέση αποθηκεύεται σε αυτό.

2. *Αν έχει ήδη ανατεθεί τμήμα μνήμης* στην εγγραφή $L\text{-}Index_{mem}[p_j]$ τότε γίνεται έλεγχος αν υπάρχει ελεύθερος χώρος σε αυτό το τμήμα.

   (α′) *Αν υπάρχει χώρος*, τότε απλά αποθηκεύουμε το ζευγάρι κόμβος-θέση αποθηκεύεται.

   (β′) *Αν το τμήμα είναι γεμάτο*, τότε συνεχίζουμε όπως στην περίπτωση 1.

Σημειώνεται ότι η διαφορά σε σχέση με την δημιουργία του *P-Index* είναι ότι απαιτείται ταξινόμηση των ζευγών (v_i, pos_ij) με βάση το αναγνωριστικό (id) των κόμβων για κάθε εγγραφή $L\text{-}Index_{mem}[p_j]$ πριν την αποθήκευση της στο δίσκο.

Παράδειγμα 4.6 (Κατασκευή *L-Index*). *Θεωρούμε την συλλογή μονοπατιών του πίνακα 4.1. Στον πίνακα 4.3 βλέπουμε την κατασκευή του αντίστοιχου L-Index. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.*

Αλγόριθμος 10: createLinkedNodeIndexMemLimited: Κατασκευή ευρετηρίου *L-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: συλλογή μονοπατιών ΣΜ, συλλογή τμημάτων μνήμης Β, $P\text{-}Index_{disk}$ στο δίσκο

Έξοδοι: $L\text{-}Index_{disk}$ στο δίσκο

Αρχικοποίηση (βοηθητικού) $L\text{-}Index_{mem}$ στην μνήμη.
Για κάθε *κόμβο* v_i *στην συλλογή μονοπατιών* ΣM **{**
  Εαν $|P\text{-}Index_{disk}[v_i]| > 1$ **{**
    Για κάθε *μονοπάτι* p_j *από το μονοπάτια*$[v_i]$ **{**
      $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
      Αποθήκευσε στο $L\text{-}Index_{mem}[p_j]$ το ζευγάρι (v_i, pos_{ij}), χρησιμοποιώντας την ρουτίνα **push** (αλγ.11).
      `// Η ρουτίνα` **push** `χρησιμοποιεί τα blocks Β και αποθηκεύει κατά καιρούς αυτόματα στο` $L\text{-}Index_{disk}$ `στο δίσκο.`
    }
  }
}
Αποθήκευση των εναπομείναντων δεδομένων του $L\text{-}Index_{mem}$ στον δίσκο, χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.12).
Διαγραφή του $L\text{-}Index_{mem}$.

Αλγόριθμος 11: push: Αποθήκευση ζευγαριού μονοπατιού-θέσης στο *L-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: μονοπάτι p_j, ζευγάρι μονοπατιού-θέσης p, συλλογή τμημάτων μνήμης B

Έξοδοι : ενημερωμένα $L\text{-}Index_{mem}$ και $L\text{-}Index_{disk}$

χρεία_νέου_block ← **ψευδές**
Εαν *δεν έχει ανατεθεί block στην εγγραφή* $L\text{-}Index_{mem}[p_j]$ **{**
  χρεία_νέου_block ← **αληθές**
}
αλλιώς {
  b ← το τελευταίο block που έχει ανατεθεί στην $L\text{-}Index_{mem}[p_j]$
  χρεία_νέου_block ← (b είναι γεμάτο)
}
Εαν όχι *χρεία_νέου_block* **{**
  Προσθήκη του ζευγαριού p στο block b.
}
αλλιώς εαν *υπάρχει διαθέσιμο block* b_{new} *στη συλλογή* B **{**
  Ανάθεση του b_{new} στην $L\text{-}Index_{mem}[p_j]$.
  Προσθήκη του ζευγαριού p στο block b_{new}.
}
αλλιώς {
  `// Τα διαθέσιμα blocks στην συλλογή B έχουν εξαντληθεί.`
  Αποθήκευση των δεδομένων του $L\text{-}Index_{mem}$ στο $L\text{-}Index_{disk}$ στο δίσκο,
  χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.12).
  Απελευθέρωση των blocks της συλλογής B.
  Ανάθεση νέου block b_{new} στην $L\text{-}Index_{mem}[p_j]$.
  Προσθήκη του ζευγαριού p στο block b_{new}.
}

Τρέχον Κόμβος	Τρέχον Μονοπάτι	Ανάθεση block σε εγγραφή μονοπατιού	Ανάθεση ζευγαριού σε block
v_0	p_0		
v_1	p_0		
v_2	p_0	$b_0 \rightarrow p_0$	$(v_2, 2) \rightarrow b_0$
v_2	p_3	$b_1 \rightarrow p_3$	$(v_2, 2) \rightarrow b_1$
v_3	p_0		$(v_3, 3) \rightarrow b_0$
v_3	p_1	$b_2 \rightarrow p_1$	$(v_3, 2) \rightarrow b_2$
v_4	p_0	$b_3 \rightarrow p_0$	$(v_4, 4) \rightarrow b_3$
v_4	p_1		$(v_4, 1) \rightarrow b_2$
v_5	p_0		
v_6	p_1	out of blocks	
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			
v_6	p_1	$b_0 \rightarrow p_1$	$(v_6, 3) \rightarrow b_0$
v_6	p_2	$b_1 \rightarrow p_2$	$(v_6, 2) \rightarrow b_1$
v_6	p_3	$b_2 \rightarrow p_3$	$(v_6, 0) \rightarrow b_2$
v_7	p_2		$(v_7, 1) \rightarrow b_1$
v_7	p_3		$(v_7, 1) \rightarrow b_2$
v_8	p_0		
v_9	p_0	$b_3 \rightarrow p_3$	$(v_9, 7) \rightarrow b_3$
v_9	p_1		$(v_9, 0) \rightarrow b_0$
v_9	p_2	out of blocks	
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			
v_9	p_2	$b_0 \rightarrow p_2$	$(v_9, 0) \rightarrow b_0$
v_{10}	p_3		
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			

Πίνακας 4.3: Κατασκευή *L-Index* για την συλλογή μονοπατιών του πιν.4.1. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.

Αλγόριθμος 12: save-to-disk: Μεταφορά του *L-Index* από την μνήμη στον δίσκο.

Είσοδοι: $L\text{-}Index_{mem}$ στη μνήμη, $L\text{-}Index_{disk}$ στο δίσκο

Έξοδοι : ενημερωμένο $L\text{-}Index_{disk}$ στο δίσκο

Για κάθε *κάθε εγγραφή* $L\text{-}Index_{mem}[p_j]$ **{**
  Ανάγνωση της εγγραφής $L\text{-}Index_{mem}[p_j]$ από τον δίσκο.
  $\kappa\acute{o}\mu\beta o\iota[p_j] \leftarrow L\text{-}Index_{mem}[p_j] \cup L\text{-}Index_{disk}[p_j]$
  Ταξινομηση της λίστας $\kappa\acute{o}\mu\beta o\iota[p_j]$ με βάση το αναγνωριστικό των κόμβων.
  Εαν $|L\text{-}Index_{disk}[p_j]| > 0$ **{**
    `// Προϋπάρχων κόμβος.`
    Διαγραφή της εγγραφής $L\text{-}Index_{mem}[p_j]$ από τον δίσκο.
  }
  Προσθήκη του $\kappa\acute{o}\mu\beta o\iota[p_j]$ στο $L\text{-}Index_{disk}$ στον δίσκο.
}

4.2 Διαδικασίες ενημέρωσης ευρετηρίων

Η ίδια μεθοδολογία που χρησιμοποιήθηκε για την κατασκευή των ευρετηρίων εφαρμόζεται και στην ενημέρωση τους. Οι συναρτήσεις push και save-to-disk που χρησιμοποιούνται είναι οι ίδιες που χρησιμοποιήθηκαν για την κατασκευή των ευρετηρίων, οπότε δεν αναφέρονται ξάνα. Οι αλγοριθμοι όμως διαφέρουν από τους αλγόριθμους ενημέρωσης που αναφέρθηκαν στην προηγούμενη ενότητα και ειδικά ο αλγόριθμος ενημέρωσης του *L-Index*.

4.2.1 Ενημέρωση αρχείου διαδρομών και ευρετηρίου *P-Index* με την προυπόθεση της ύπαρξης απεριόριστης μνήμης

Παράδειγμα 4.7 (Ενημέρωση *P-Index*). *Έστω οτι θέλουμε να προσθέσουμε τα μονοπάτια της ΣΜ του πιν.4.4 στην συλλογή μονοπατιών του πίνακα 4.1 με το αντίστοιχο P-Index (βλέπε πιν.4.2).*

Μονοπάτι	Λίστα Κόμβων
p_4	v_8, v_2, v_{11}, v_{12}
p_5	v_8, v_{11}

Πίνακας 4.4: Συλλογή Μονοπατιών ενημέρωσης με 2 μονοπάτια και 6 κόμβους.

Στον πίνακα 4.5 δείχνουμε την διαδικασία ενημέρωσης του P-Index με χρήση περιορισμένης μνήμης. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.

Τρέχον Μονοπάτι	Τρέχον Κόμβος	Ανάθεση block σε εγγραφή κόμβου	Ανάθεση ζευγαριού σε block
p_4	v_8	$b_0 \rightarrow v_8$	$(p_4, 0) \rightarrow b_0$
p_4	v_2	$b_1 \rightarrow v_2$	$(p_4, 1) \rightarrow b_1$
p_4	v_{11}	$b_2 \rightarrow v_{11}$	$(p_4, 2) \rightarrow b_2$
p_4	v_{12}	$b_3 \rightarrow v_{12}$	$(p_4, 3) \rightarrow b_3$
p_5	v_8		$(p_5, 0) \rightarrow b_0$
p_5	v_{11}		$(p_5, 1) \rightarrow b_2$
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			

Πίνακας 4.5: Ενημέρωση *P-Index* της ΣΜ του πιν.4.1 (κατασκευή *P-Index* στον πιν.4.2) με την ΣΜ του πιν.4.4. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.

Αλγόριθμος 13: updatePIndexMemLimited: Ενημέρωση ευρετηρίου *P-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: αριθμός παλιών μονοπατίων στη συλλογή μονοπατιών ΣΜ, αρχείο ενημερώσεων Ε, συλλογή τμημάτων μνήμης Β, $P\text{-}Index_{disk}$ στο δίσκο

Έξοδοι : $P\text{-}Index_{mem}$ ενημερώσεων στη μνήμη, ενημερωμένη συλλογή μονοπατιών ΣΜ, ενημερωμένο $P\text{-}Index_{disk}$ στο δίσκο

$j \leftarrow$ αριθμός παλιών μονοπατιών $+1$
Για κάθε *μονοπάτι* p_j *στο αρχείο* E **{**
  Για κάθε *κόμβο* v_i *στο μονοπάτι* p_j **{**
    $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
    Αποθήκευσε στο $P\text{-}Index_{mem}[v_i]$ το ζευγάρι (p_j, pos_{ij}) με χρήση της διαδικασίας **push** (αλγ.8).
    `// Η ρουτίνα` **push** `χρησιμοποιεί τα blocks Β και αποθηκεύει κατά καιρούς αυτόματα στο` $P\text{-}Index_{disk}$ `στο δίσκο.`
  }
  Αποθήκευσε το μονοπάτι p_j στη συλλογή μονοπατιών ΣΜ στο σκληρό δίσκο.
  $j \leftarrow j + 1$
}
Αποθήκευση των εναπομείναντων δεδομένων του $P\text{-}Index_{mem}$ στον δίσκο, χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.9).

4.2.2 Ενημέρωση ευρετηρίου *L-Index* με περιορισμένη διάθεση της κύριας μνήμης

Η διαδικασία ενημέρωσης του ευρετηρίου *L-Index* διαφοροποιείται από την διαδικασία ενημέρωσης του προηγούμενου κεφαλαίου όπου είχε θεωρηθεί απεριόριστη η διαθέσιμη κύρια μνήμη. Υπενθυμίζεται ότι εκείνος ο αλγόριθμος είχε βασιστεί στην υπόθεση ότι όταν κατασκευάζεται το ευρετήριο *L-Index* των ενημερώσεων στη μνήμη, ο *P-Index* δεν έχει ενημερωθεί ήδη με βάση τα νέα μονοπάτια. Η μέθοδος που χρησιμοποιήθηκε σε αυτή την περίπτωση είναι πιο γρήγορη αλλά η εφάρμοφή της έγινε εφικτή απο το γεγονός ότι υπήρχε η δυνατότητα διαχωρισμού της κατασκευής των δύο ευρετηρίων ενημέρωσης από την εγγραφή των ενημερώσεων στο δίσκο.Κάτι τέτοιο δεν είναι πια δυνατό.

Η νέα μεθοδολογία (αλγ.14) λειτουργεί ως εξής. Για κάθε νέο μονοπάτι, ελέγχουμε αν οι κόμβοι του βρίσκονται σε περισσότερο του ενός μονοπάτια. Αν ναι, τότε ο κόμβος είναι σύνδεσμος και τον προσθέτουμε στον *L-Index* στη μνήμη. Επιπλέον, ελέγχουμε σε πόσα παλιά μονοπάτια περιλαμβάνεται ο κόμβος. Αν περιλαμβάνεται σε ένα μόνο παλιό μονοπάτι, ο κόμβος έγινε συνδεσμος λόγω των ενημερώσεων και επομένως πρέπει να ενημερώσουμε και την εγγραφή του *L-Index* για το παλιό μονοπάτι.

Παράδειγμα 4.8 (Ενημέρωση *L-Index*). *Στον πίνακα 4.6 δείχνουμε την διαδικασία ενημέρωσης του L-Index (πιν.4.3) της ΣΜ του πιν.4.1 με τα μονοπάτια της ΣΜ του πιν.4.4. Η ενημέρωση γίνεται με χρήση περιορισμένης μνήμης: έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση.*

Τρέχον Μονοπάτι	Τρέχον Κόμβος	Ανάθεση block σε εγγραφή κόμβου	Ανάθεση ζευγαριού σε block
p_4	v_8	$b_0 \rightarrow p_4$	$(v_8, 0) \rightarrow b_0$
$^*p_0$	$^*v_8$	$b_1 \rightarrow p_0$	$(v_8, 2) \rightarrow b_1$
p_4	v_2		$(v_2, 1) \rightarrow b_0$
p_4	v_{11}	$b_2 \rightarrow p_4$	$(v_{11}, 2) \rightarrow b_2$
p_4	v_{12}		
p_5	v_8	$b_3 \rightarrow p_5$	$(v_8, 0) \rightarrow b_3$
p_5	v_{11}		$(v_{11}, 1) \rightarrow b_3$
ΕΓΓΡΑΦΗ ΣΤΟ ΔΙΣΚΟ			

Πίνακας 4.6: Ενημέρωση *L-Index* της ΣΜ του πιν.4.1 (κατασκευή *L-Index* στον πιν.4.3) με την ΣΜ του πιν.4.4. Έχουμε 4 διαθέσιμα τμήματα μνήμης και το καθένα χωράει 2 ζευγάρια μονοπάτι-θέση. (*)Παρατηρούμε οτι ο κόμβος v_8 γίνεται κόμβος-σύνδεσμος λόγω των ενημερώσεων και επομένως ο αλγόριθμος ανανεώνει και την εγγραφή του παλιού μονοπατιού p_0.

Αλγόριθμος 14: updateLinkNodesIndexMemLimited: Ενημέρωση ευρετηρίου *L-Index* με χρήση περιορισμένης μνήμης.

Είσοδοι: συλλογή τμημάτων μνήμης B, $P\text{-}Index_{disk}$ στο δίσκο, $P\text{-}Index_{disk}$ στο δίσκο

Έξοδοι : $L\text{-}Index_{mem}$ ενημερώσεων στη μνήμη, ενημερωμένη συλλογή μονοπατιών ΣM, ενημερωμένο $L\text{-}Index_{disk}$ στο δίσκο

Για κάθε *νέο μονοπάτι* p_j **{**
  Για κάθε *κόμβο* $v_i \in$ *κόμβοι*$[p_j]$ **{**
    Εαν $|P\text{-}Index_{disk}[v_i]| > 1$ **{**
      $pos_{ij} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_j
      Αποθήκευσε στο $L\text{-}Index_{mem}[p_j]$ το ζευγάρι (p_j, pos_{ij}) με χρήση της διαδικασίας **push** (αλγ.11).
      `// Η ρουτίνα` **push** `χρησιμοποιεί τα blocks B και αποθηκεύει κατά καιρούς αυτόματα στο` $L\text{-}Index_{disk}$ `στο δίσκο.`
      αριθμός_παλιών_μονοπατιών $\leftarrow$ πλήθος $p_k \in$ *μονοπάτια*$[v_i]$ με $k < j$
      Εαν *αριθμός_παλιών_μονοπατιών*$= 1$ **{**
        $pos_{ik} \leftarrow$ θέση κόμβου v_i στο μονοπάτι p_k
        Αποθήκευσε στο $L\text{-}Index_{mem}[p_k]$ το ζευγάρι (p_j, pos_{ik}) με χρήση της διαδικασίας **push** (αλγ.11).
      }
    }
  }
}
Αποθήκευση των εναπομείναντων δεδομένων του $L\text{-}Index_{mem}$ στον δίσκο, χρησιμοποιώντας την ρουτίνα **save-to-disk** (αλγ.12).

Κεφάλαιο 5

Αξιολόγηση

Στο κεφάλαιο αυτό παρουσιάζουμε την πειραματική αξιολόγηση των τεχνικών μας.

5.1 Οργάνωση πειραμάτων

Υπάρχουν δύο ομάδες πειραμάτων.

- Πειράματα που αφορούν την κατασκευή των ευρετηρίων.
- Πειράματα που αφορούν την ενημέρωση των ευρετηρίων.

Στο πλαίσιο των πειραμάτων που αφορούν την κατασκευή των ευρετηρίων χρησιμοποιήσαμε συνθετικούς γράφους των 50K, 100K, 500K και 1M κόμβων. Στο πλαίσιο των πειραμάτων που αφορούν την ενημέρωση των ευρετηρίων χρησιμοποιήσαμε συνθετικό γράφο 100K με μεταβαλλόμενη παράμετρο το μέγεθος του αρχείου ενημέρωσης σε σχέση με το αρχείο κατασκευής. Και στις δύο ομάδες πειραμάτων οι συνθετικοί γράφοι αναφέρονται με ονόματα $VkPkZL$, όπου V ο αριθμός των κόμβων, P ο αριθμός των μονοπατιών στη συλλογή μονοπατιών, Z το ποσοστό δημοτικότητας κόμβων και L το μέσο μήκος μονοπατιού.

5.1.1 Πειράματα που αφορούν την κατασκευή των ευρετηρίων

Για αυτή τη ομάδα πειραμάτων κατασκευάσαμε αρχεία με συνθετικές συλλογές μονοπατιών. Χρησιμοποιούμε έξι διαφορετικές παραμέτρους για τα πειράματα:

1. V: ο αριθμός των διαφορετικών κόμβων στη Συλλογή Μονοπατιών,
2. P: ο αριθμός των μονοπατιών στη Συλλογή Μονοπατιών,
3. Z: η τάξη της κατανομής της συχνότητας κόμβου,
4. B: ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη
5. $PAIRS$: ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης.

Παράμετροι	Τιμές	Τιμή εξ ορισμού
V	10K, 50K, 100K, 500K, 1M	100K
P	50K, 100K, 500K, 1M	100K
L	5, 10, 30	10
Z	0, 0.3, 0.6, 0.8, 0.999	0.6
B	2K, 10K, 20K, 50K, 70K	10K, 70K*
$PAIRS$	2, 5, 10, 15, 20	10

Πίνακας 5.1: Παράμετροι που χρησιμοποιούνται στα πειράματα κατασκευής των ευρετηρίων.

Οι συλλογές μονοπατιών περιέχουν 10K, 50K, 100K, 500K, 1M διαφορετικούς κόμβους και 50K, 100K, 500K, 1M μονοπάτια. Το μέσο μήκος μονοπατιών είναι 5, 10, 30. Η συχνότητα κόμβου έιναι κατανομή Zipf με μέτρια λόξευση που μεταβάλλεται από 0, 0.3, 0.6, 0.8, 0.999. Σημειώνεται ότι κόμβοι με μεγαλύτερη συχνότητα απαντώνται σε περισσότερα μονοπάτια. Ο αριθμός των τμημάτων στα οποία χωρίσαμε την κύρια μνήμη μεταβάλλεται από 2K, 10K, 20K, 50K και 70K ενώ ο αριθμός των ζευγαριών που μπορούν να τοποθετηθούν σε ένα τμήμα ποικίλει ανάμεσα σε 2 και 20. Στον πίνακα 5.1 μπορούμε να δούμε συνολικά της παραμέτρους που χρησιμοποιούνται στα πειράματα κατασκευής των ευρετηρίων.

Πραγματοποιούμε 6 ομάδες πειραμάτων για να δείξουμε το αποτέλεσμα σε χρόνο στην κατασκευή των ευρετηρίων. Σε κάθε ομάδα πειραμάτων μεταβάλλουμε μία παράμετρο ενώ διατηρούμε τις υπόλοιπες σταθερές σε μία εξ ορισμού τιμή. Σημειώνουμε οτι για τα πειράματα με δαιφορετικό αριθμό ζευγών η εξ ορισμού τιμή για τον αριθμό των τμημάτων μνήμης είναι 70K για λόγους που θα εξηγήσουμε παρακάτω.

5.1.2 Πειράματα που αφορούν την ενημέρωση των ευρετηρίων

Για τα πειράματα αυτή της ομάδας χρησιμοποιούμε συνθετικό γραφο με σταθερές παραμέτρους V, P, L, Z, B, $PAIRS$ στις εξ ορισμού τιμές που αναφέρθηκαν παραπάνω και μεταβαλλόμενες παραμέτρους:

1. U: το μέγεθος του αρχείου των ενημερώσεων σε σχέση με το αρχικό αρχείο,

2. B_{up}: ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη για τα ευρετήρια ενημερώσεων,

3. $PAIRS_{up}$: ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* των ενημερώσεων αντίστοιχα που μπορούν να τοποθετηθούν σε ένα τμήμα (block) κύριας μνήμης.

Στον πίνακα 5.2 μπορούμε να δούμε συνολικά της παραμέτρους που χρησιμοποιούνται στα πειράματα κατασκευής των ευρετηρίων. Σημειώνουμε για τα πειράματα με δαιφορετικό αριθμό ζευγών η εξ ορισμού τιμή για τον αριθμό των τμημάτων μνήμης είναι 7K για λόγους που θα εξηγήσουμε παρακάτω.

Παράμετροι	Τιμές	Τιμή εξ ορισμού
V	100K	-
P	100K	-
L	10	-
Z	0.6	-
B	10K	-
$PAIRS$	10	-
U	1%, 5%, 10%	10%
B_{up}	100, 200, 500, 1000, 5000	1000, 7000*
$PAIRS_{up}$	2, 3, 4, 5	4

Πίνακας 5.2: Παράμετροι που χρησιμοποιούνται στα πειράματα ενημέρωσης των ευρετηρίων.

5.2 Αποτελέσματα

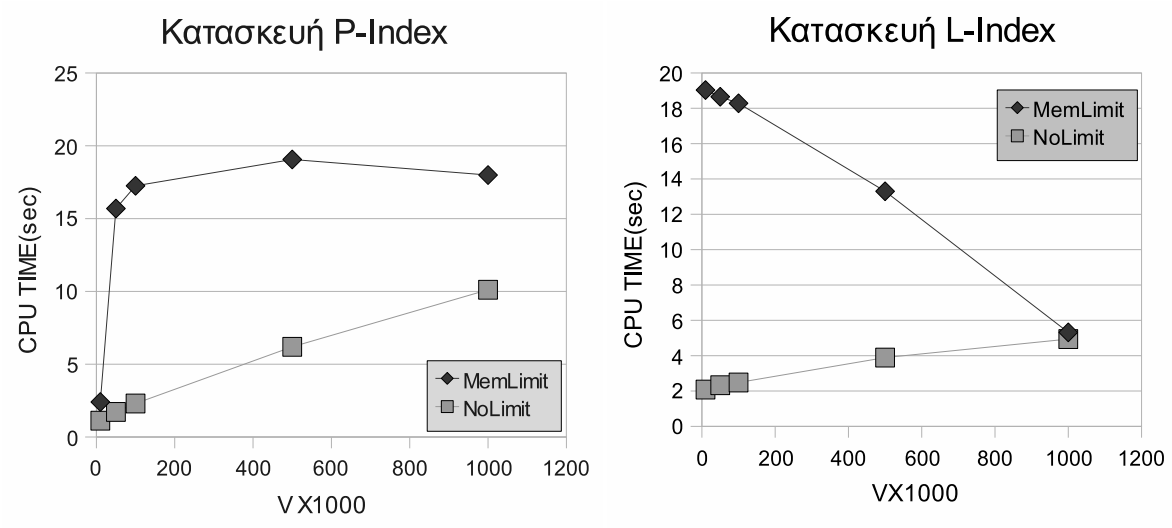
5.2.1 Κατασκευή Ευρετηρίων

Μεταβολή του αριθμού των κόμβων

Αρχικά εξετάζουμε το αποτέλεσμα της μεταβολής του αριθμού των κόμβων στο χρόνο που χρειάζεται για να κατασκευαστούν τα ευρετήρια *P-Index* και *L-Index*. Κατασκευάσαμε πέντε αρχεία με συλλόγες μονοπατιών που περιλαμβάνουν 10K, 50K, 100K, 500K, 1000K κόμβους. Ορίζουμε τον αριθμό των μονοπατιών που περιέχονται στο αρχείο σταθερά στα 100K μονοπατια και το μέσο μήκος τους στο 10. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται σταθερά στο 10K ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης στο 10. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων *P-Index* και *L-Index*.

Το σχήμα 5.1 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*. Με τον συμβολισμό MemLimit αναφερόμαστε στους αλγόριθμους κατασκευή με περιορισμό μνήμης ενώ με τον συμβολισμό NoLimit αναφερόμαστε στους αλγόριθμους κατασκευής στη μνήμη χωρίς περιορισμό και την ολοκληρωτική μεταφορά των ευρετηρίων στο δίσκο, τα αποτελέσματα των οποίων παραθέτουμε για λόγους πληρότητας.

Παρατηρήσεις: (Για τον αλγόριθμο με περιορισμό μνήμης.) Οπως παρατηρούμε οι απαιτήσεις σε χρόνο για την κατασκευή του *P-Index* αυξάνουν καθώς αυξάνει ο αριθμός των μονοπατιών στη ΣΜ. Οσο αυξάνει το V αυξάνει ο αριθμός των εγγραφών στο ευρετήριο *P-Index* οπότε χρειάζεται περισσότερος χρόνος για την κατασκευή του. Σε αντίθεση η κατασκευή του *L-Index* χρειάζεται λιγότερο χρόνο καθώς αυξάνεται το V. Με την αύξηση του αριθμού των κόμβων και κρατώντας τα υπόλοιπα μεγέθη σταθερά μειώνεται ο αριθμός των κόμβων συνδέσμων οπότε μικραίνει το μέγεθός του *L-Index*. Η ανακολουθία μεταξύ των αποτελεσματων με περιορισμό κύρια μνήμης και χωρίς περιορισμό στην κατασκευή του *L-Index* οφείλεται στο γεγονός ότι ο χρόνος κατασκευής του εξαρτάται από το μέγεθος που μειώνεται



Σχήμα 5.1: Χρόνοι κατασκευής *P-Index* και *L-Index* για διάφορα πλήθη κόμβων, με και χωρίς περιορισμό μνήμης.

με την αυξηση του V και τον αριθμό των εγγραφών του *P-Index* που επισκεπτεται για την κατασκευή του ο οποίος αυξανεται πολύ. Ομως η κατασκευή με περιορισμό επηρεάζεται επίσης από έναν καθοριστικό παράγοντα, τον αριθμό των φορών που γράφει στο δίσκο κάθε φορά που τελειώνουν τα διαθέσιμα τμήματα μνήμης. Με σταθερό των αριθμό των τμημάτων και το μέγεθος του *L-Index* να μικράινει, η κατασκευή γίνεται σε λιγότερες επαναλήψεις στο σκληρό δίσκο.

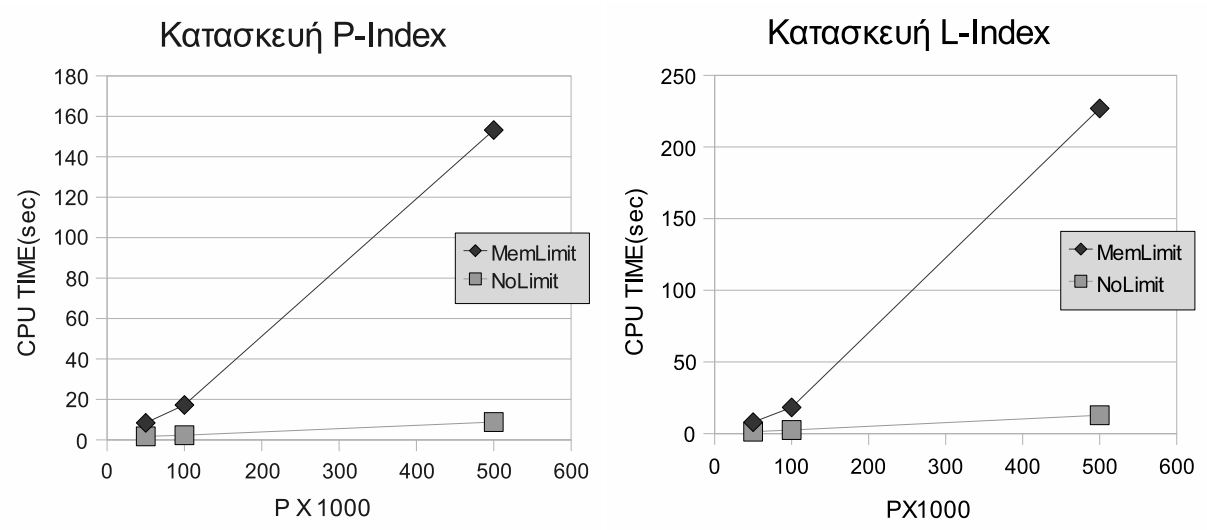
Σημείωση: Η πολύ μικρή τιμή για την κατασκευή του *P-Index* που παρατηρείται στους 10K κόμβους οφείλεται στο γεγονός οτι έχει διατεθεί μεγάλος αριθμός τμημάτων μνήμης, περισσότερα από όσα χρειάζονται για τον αριθμό των κόμβων που περιέχει.

Μεταβολή του αριθμού των μονοπατιών

Σε αυτό το πείραμα μελετάμε την επίδραση της μεταβολής στον αριθμό των μονοπατιών στον χρόνο κατασκευής των ευρετηρίων *P-Index* και *P-Index*. Κατασκευάσαμε τέσσερα αρχεία με συλλόγες μονοπατιών που περιλαμβάνουν 50K, 100K, 500K, 1,000K μονοπάτια. Ορίζουμε τον αριθμό των κόμβων που περιέχονται στο αρχείο σταθερά στα 100K μονοπατια και το μέσο μήκος τους στο 10. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται σταθερά στο 10K. Ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης ορίζεται στο 10. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων *P-Index* και *L-Index*.

Το σχήμα 5.2 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*.

Παρατηρήσεις: (Για τον αλγόριθμο με περιορισμό μνήμης.) Ο χρόνος κατασκευής του



Σχήμα 5.2: Χρόνοι κατασκευής *P-Index* και *L-Index* για διάφορα πλήθη μονοπατιών, με και χωρίς περιορισμό μνήμης.

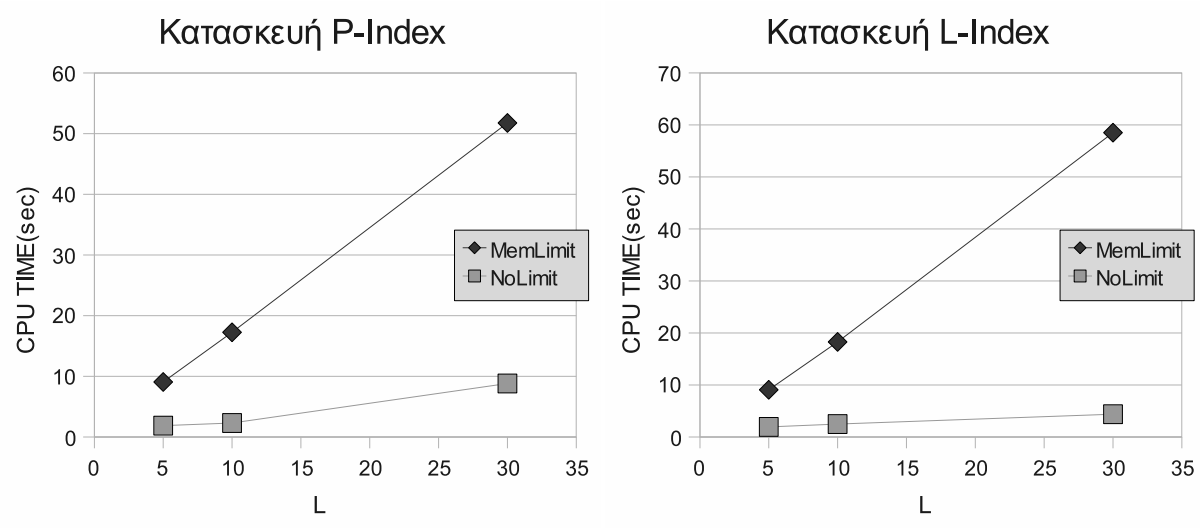
ευρετηρίων αυξάνει με την αυξηση του αριθμού των μονοπατιών P. Καθώς το P αυξάνεται, κάθε κόμβος περιέχεται σε περισσότερα μονοπάτια οπότε το μέγεθος της λίστας μονοπατιών για κάθε εγγραφή του *P-Index* αυξάνεται όπως και ο αριθμός των κόμβων-συνδέσμων του ευρετηρίου *L-Index*. Αρα απαιτείται περισσότερος χρόνος για την κατασκευή τους. Τα αποτελέσματα των αλγόριθμων χωρίς περιορισμό στην κύρια μνήμη συμφωνούν με τις προηγούμενες παρατηρήσεις.

Μεταβολή του μέσου μήκους μονοπατιού

Σε αυτήν την ενότητα εξετάζουμε τα αποτελέσματα που έχει η μεταβολή του μέσου μήκους μονοπατιού στον χρονο που χρειάζεται η κατασκευή των ευρετηρίων. Κατασκευάσαμε τρία αρχεία με συλλόγες μονοπατιών που περιλαμβάνουν μονοπατια με μέσο μήκος 5, 10, 30. Ορίζουμε τον αριθμό των μονοπατιών που περιέχονται στο αρχείο σταθερά στα 100K μονοπατια και τον αριθμό κόμβων στους 100K διακριτούς κόμβους. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται σταθερά στο 10K. Ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης τίθεται στο 10. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων.

Το σχήμα 5.3 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*.

Παρατηρήσεις: (Για τον αλγόριθμο με περιορισμό μνήμης.) Οσο το μέσο μήκος μονοπατιού L αυξάνεται, υπάρχουν περισσότερες άμεσες συνδέσεις μεταξύ των κόμβων. Αυτό οδηγεί σε περισσότερους κοινούς κόμβους ανάμεσα στα μονοπάτια εφόσων ο αριθμός κόμβων παραμένει σταθερός. Ετσι μεγαλώνει το μέγεθος και των δύο ευρετηρίων εφόσον μεγαλώνει



Σχήμα 5.3: Χρόνοι κατασκευής *P-Index* και *L-Index* για διάφορες τιμές του μέσου μήκους μονοπατιών, με και χωρίς περιορισμό μνήμης.

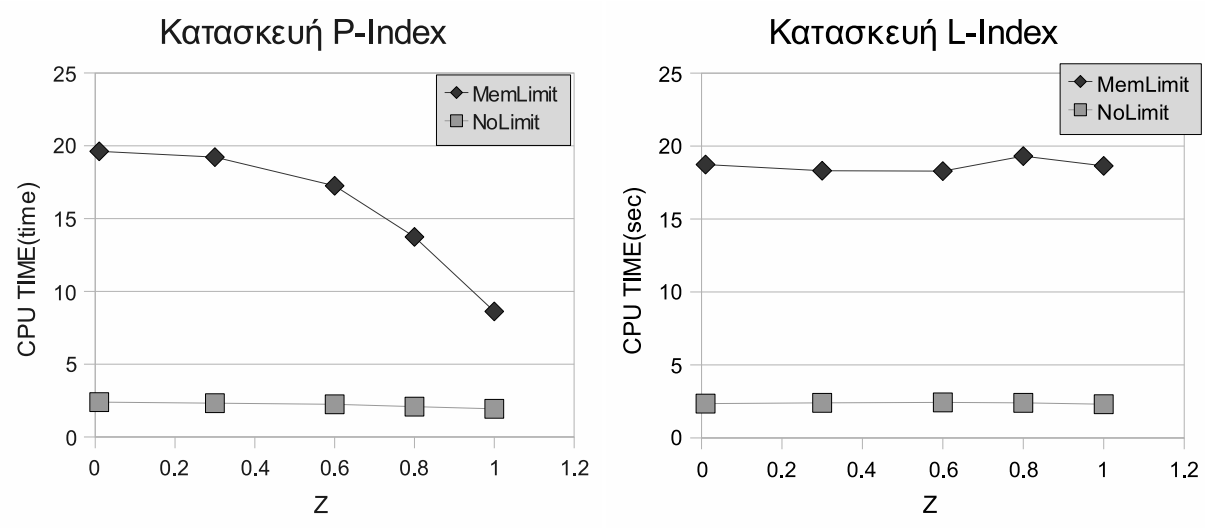
η λίστα μονοπατιών για κάθε εγγραφή του *P-Index* και αυξάνονται οι κόμβοι-σύνδεσμοι για τον *L-Index*. Αρα αυξάνεται και ο χρόνος κατασκευής. Τα αποτελέσματα των αλγόριθμων χωρίς περιορισμό στην κύρια μνήμη συμφωνούν με τις προηγούμενες παρατηρήσεις.

Μεταβολή της συχνότητας κόμβου

Εδώ εξετάζουμε την επίδραση της συχνότητας κόμβου στον χρόνο κατασκευής των ευρετηρίων *P-Index* και *L-Index*. Κατασκευάσαμε τέσσερα αρχεία με συλλόγες μονοπατιών που που έχουν συχνότητα κόμβου που ακολουθεί την κατανομή Zipf τάξης 0, 0.3, 0.6, 0.8, 0.999. Ορίζουμε όμως σταθερό τον αριθμό των μονοπατιών που περιέχονται στο αρχείο στα 100K μονοπατια, το μέσο μήκος τους στο 10 και τον αριθμό κόμβων στους 100K διακριτούς κόμβους.Ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται σταθερά στο 10K και ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης τίθεται στο 10. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων.

Το σχήμα 5.4 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*.

Παρατηρήσεις: (Για τον αλγόριθμο με περιορισμό μνήμης.) Καθώς αυξάνεται η τάξη για την κατανομή Zipf της συχνότητας κόμβου άλλοι κόμβοι γίνονται περισσότερο δημοφιλείς ενώ άλλοι λιγότερο. Αυτό έχει ως αποτέλεσμα το μέγεθος του *P-Index* να μικραίνει αλλά το μέγεθος του *L-Index* να παραμένει περίπου το ίδιο, εφόσων οι εμφανίσεις των κόμβων συνδέσμων στα μονοπάτια παραμένουν στα ίδια περίπου επίπεδα.Τα αποτελέσματα των αλγόριθμων χωρίς περιορισμό στην κύρια μνήμη συμφωνούν με τις προηγούμενες παρατηρήσεις.



Σχήμα 5.4: Χρόνοι κατασκευής *P-Index* και *L-Index* για συχνότητες κόμβων που ακολουθούν κατανομές Zipf διαφόρων τάξεων, με και χωρίς περιορισμό μνήμης.

Μεταβολή του αριθμού των τμημάτων (blocks) μνήμης

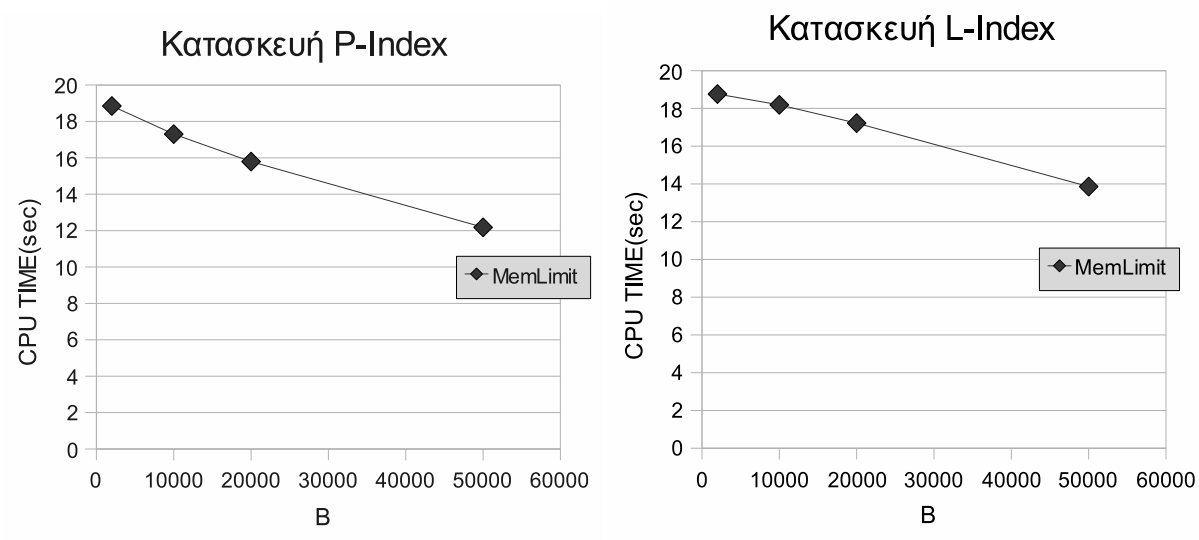
Εδώ εξετάζουμε την επίδραση του αριθμού των τμημάτων μνήμης στον χρόνο κατασκευής των ευρετηρίων *P-Index* και *L-Index*. Πειραματιστήκαμε με τέσσερις διαφορετικές επιλογές για τον αριθμό των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη, με 2K, 10K, 20K και 50K τμήματα. Ορίζουμε όμως σταθερό τον αριθμό των μονοπατιών που περιέχονται στο αρχείο στα 100K μονοπατια, το μέσο μήκος τους στο 10 και τον αριθμό κόμβων στους 100K διακριτούς κόμβους. Η συχνότητα κόμβου που ακολουθεί την κατανομή Zipf είναι τάξης 0.6 και ο αριθμός των ζευγαριών κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης στο 10. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων *P-Index* και *L-Index*.

Το σχήμα 5.5 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*.

Παρατηρήσεις: Καθώς αυξάνεται ο αριθμός B των διαθέσιμών τμημάτων μνήμης μειώνεται ο χρόνος κατασκευής των ευρετηρίων γιατί μειώνεται ο αριθμός των φορών που απαιτείται εγγραφή σε δευτερευον αποθηκευτικό μέσο (σκληρό δίσκο).

Μεταβολή του αριθμού των ζευγαριών ανά τμήμα μνήμης

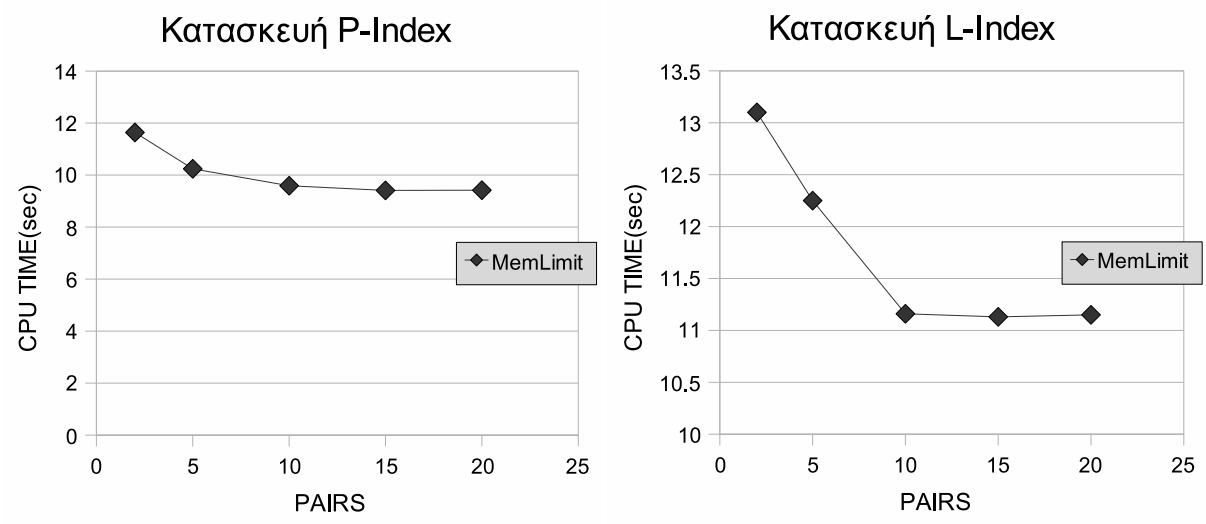
Εδώ εξετάζουμε την επίδραση της χωρητικότητας των τμημάτων μνήμης σε ζευγάρια στον χρόνο κατασκευής των ευρετηρίων *P-Index* και *L-Index*. Πειραματιστήκαμε με πέντε διαφορετικές επιλογές για τον αριθμό των ζευγων κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα κύριας μνήμης. Χρησιμοποιήσαμε τις χωρητικότητες 2, 5, 10, 15 και 20 ζευγάρια. Ορίζουμε σταθερό τον αριθμό των μονοπατιών που περιέχονται στο αρχείο στα 100K μονοπατια, το μέσο μήκος τους στο 10



Σχήμα 5.5: Χρόνοι κατασκευής *P-Index* και *L-Index* για διάφορα πλήθη διαθέσιμων τμημάτων κύριας μνήμης.

και τον αριθμό κόμβων στους 100K διακριτούς κόμβους. Η συχνότητα κόμβου ακολουθεί την κατανομή Zipf τάξης 0.6 και ο αριθμόω των τμημάτων στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται στα 70K. Οι ίδιες παράμετροι ισχύουν για την κατασκευή και των δύο ευρετηρίων. Η αλλαγή στον αριθμό των τμημάτων σε σχέση με τα προηγούμενα πειράματα έγινε επειδή έχουμε λίγους κοινούς κόμβους στα συγκεκριμένα δεδομένα, οπότε θέλουμε να διαβάζουμε όσο δυνατόν περισσότερους κοινούς κόμβους με ένα πέρασμα.

Το σχήμα 5.6 δείχνει τον χρόνο που απαιτείται για την κατασκευή των *P-Index* και *L-Index*.



Σχήμα 5.6: Χρόνοι κατασκευής *P-Index* και *L-Index* για διάφορες χωρητικότητες των τμημάτων κύριας μνήμης σε ζευγαρια.

Παρατηρήσεις: Καθώς αυξάνεται ο αριθμός των ζευγαριών μειώνεται ο χρόνος κατασκευής για τα ευρετήρια γιατί αυξανεται η διαθέσιμη κύρια μνήμη. Ολα αυτά μέχρι εναν αριθμό τμημάτων. Περα απο αυτό μεγαλύτερη αυξηση των ζευγαρίων δεν προσφέρει κέρδος σε χρόνο γιατί έχουν δωθεί περισσότερα από όσα χρειάζονται. Στα συγκεκριμένα πειράματα χρησιμοποιούμε μικρό αριθμό ζευγαριών γιατί δεν έχουμε πολλούς κοινούς κόμβους στα δεδομένα μας.

5.2.2 Ενημέρωση Ευρετηρίων

Τέλος μελετάμε τις μεθόδους για την ενημέρωση των ευρετηρίων συνολικά. Δηλαδή μετράμε τον χρόνο που απαιτείται για την ενημέρωση της συλλογής μονοπατιών και των δύο ευρετηρίων *P-Index* και *L-Index* μαζί. Η ενημέρωση περιλαμβάνει προσθήκη νέων μονοπατιών τα οποία μπορούν και περιέχουν νέους κόμβους που εμφανίζονται πρώτη φορά στη συλλογή μονοπατιών αλλά και με συλλόγες μονοπατιών με σταθερό τον αριθμό κόμβων της αρχικής συλλογής.

Εχουμε δυό κατηγορίες πειραμάτων σε αυτη την ενότητα:

- αρχεία ενημέρωσης με καινούριους κόμβους που δεν υπήρχαν στην αρχική συλλογή
- αρχεία ενημέρωσης μόνο με κόμβους της αρχικής συλλογής (*σταθ. V*)

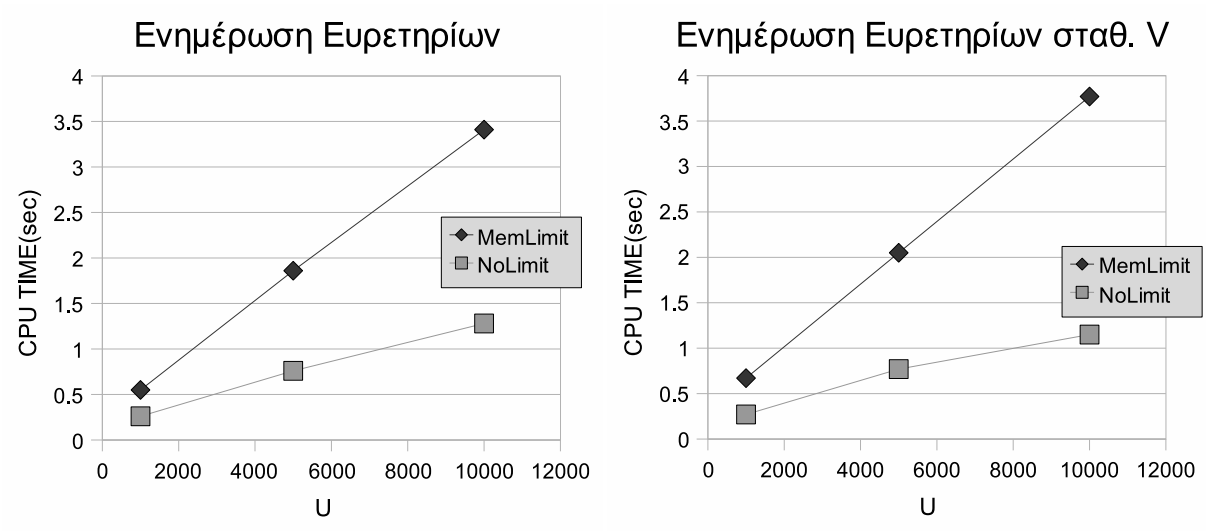
Οι χρόνοι που αναφέρονται στα γραφήματα αφορούν την ενημέρωση *και* του *P-Index* *και* του *L-Index*.

Μεταβολή του μεγέθους του αρχείου ενημέρωσης

Αρχικά εξετάζουμε την επίδραση της μεταβολής του μεγέθους του αρχείου ενημέρωσης σε σχέση με το αρχικό αρχείο στην ενημερωση των ευρετηρίων *P-Index* και *L-Index*. Πειραματιστήκαμε με τρία αρχεία διαφορετικού μεγέθους 1%, 5% 10% του αρχείου της αρχικής ΣΜ. Επειδή έχουμε ορίσει τον αριθμό των μονοπατιών που περιέχονται στο αρχείο σταθερά στα 100Κ μονοπατια, τα αρχεία ενημερώσεων περιέχουν αντίστοιχα 1Κ, 5Κ και 10Κ μονοπάτια. Το μέσο μήκος τους είναι 10 και ο αριθμός των κόμβων 100Κ. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη ορίζεται σταθερά στο 200 και ο αριθμός των ζευγων κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα (block) κύριας μνήμης στο 100. Οι ίδιες παράμετροι ισχύουν για την ενημέρωση και των δύο ευρετηρίων *P-Index* και *L-Index*.

Το σχήμα 5.7 δείχνει τον χρόνο που απαιτείται για την ενημέρωση των ευρετηρίων.

Παρατηρήσεις: Όπως είναι εμφανές και για τις δύο περιπτώσεις, όταν αυξάνεται το μέγεθος του αρχείου ενημέρωσης αυξάνεται και ο χρόνος για την ενημέρωση των ευρετηρίων. Το αποτέλεσμα είναι αναμενόμενο, δεδομένου ότι όταν αυξάνεται ο αριθμός των μονοπατιών αυξάνεται το μέγεθος των ευρετηρίων με αποτέλεσμα να απαιτείται περισσότερος χρόνος για την ενημέρωση τους.



Σχήμα 5.7: Χρόνοι ενημέρωσης ευρετηρίων για διάφορα μεγέθη αρχείου ενημέρωσης. Για αρχεία με καινούριους κόμβους και αρχεία χωρις καινούριους κόμβους (σταθ. V).

Μεταβολή του αριθμού των τμημάτων (blocks) κύριας μνήμης

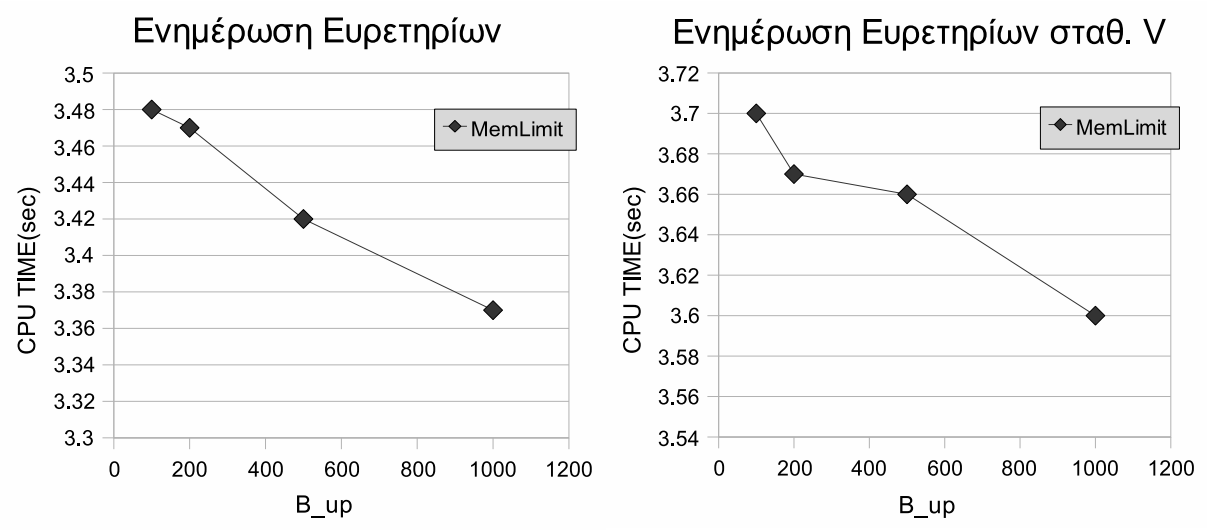
Εξετάζουμε την επίδραση της μεταβολής του αριθμού των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη. Πειραματιστήκαμε αρχικά με τέσσερις διαφορετικές επιλογές στο πλήθος των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη: 100, 200, 500 και 1000 τμήματα. Έχουμε ορίσει τον αριθμό των μονοπατιών που περιέχονται στο αρχείο σταθερά στα 100K μονοπατια. Το μέσο μήκος τους είναι 10 και ο αριθμός των κόμβων 100K. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των ζευγων κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα (block) κύριας μνήμης στο 10. Οι ίδιες παράμετροι ισχύουν για την ενημέρωση και των δύο ευρετηρίων *P-Index* και *L-Index*.

Το σχήμα 5.8 δείχνει τον χρόνο που απαιτείται για την ενημέρωση των ευρετηρίων.

Παρατηρήσεις: Όπως είναι εμφανές και στις δύο περιπτώσεις, η αυξήση των τμημάτων μνήμης οδηγεί σε λιγότερες επαναλήψεις για εγγραφή στον δίσκο με αποτέλεσμα την μείωση του χρόνου ενημέρωσης.

Μεταβολή του αριθμού των ζευγών

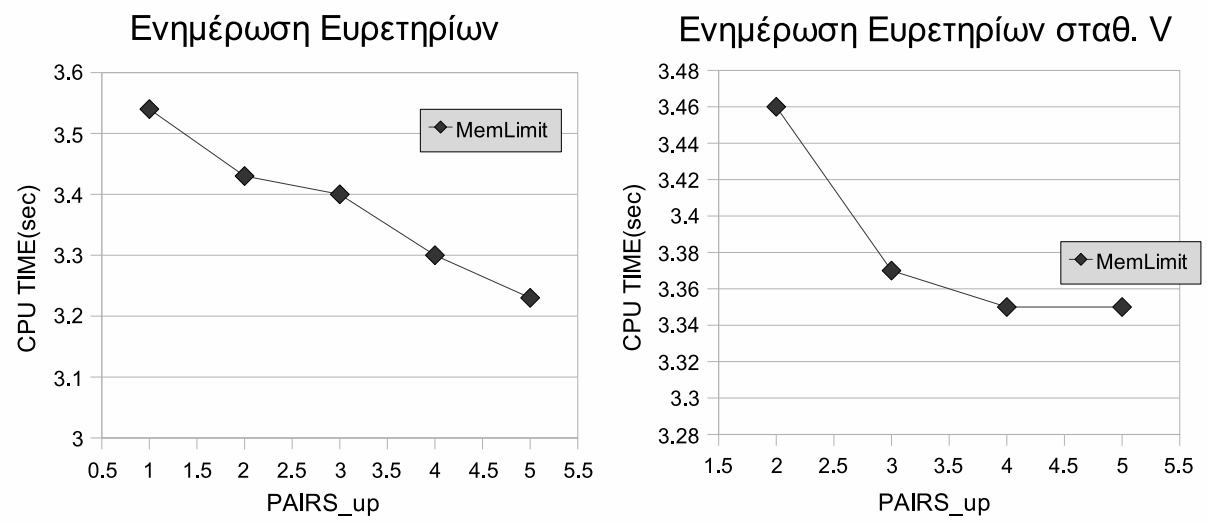
Εξετάζουμε την επίδραση της μεταβολής του αριθμού των ζευγων κόμβος-θέση ή μονοπάτι-θέση για τους *L-Index*, *P-Index* αντίστοιχα, που μπορούν να τοποθετηθούν σε ένα τμήμα (block) κύριας μνήμης. Οι τιμές που ελέγχουμε είναι 2 ,3 ,4 ,5. Έχουμε ορίσει τον αριθμό των μονοπατιών που περιέχονται στο αρχείο σταθερά στα 100K μονοπατια. Το μέσο μήκος τους είναι 10 και ο αριθμός των κόμβων 100K. Η συχνότητα κόμβου ακολουθεί κατανομή Zipf τάξης 0.6 ενώ ο αριθμός των τμημάτων (blocks) στα οποία χωρίσαμε την διαθέσιμη κύρια μνήμη για την κατασκευή των ευρετηρίων ενημέρωσης στο 7000. Οι ίδιες παράμετροι ισχύουν για την ενημέρωση και των δύο ευρετηρίων *P-Index* και *L-Index*. Η αλλαγή στον αριθμό των



Σχήμα 5.8: Χρόνοι ενημέρωσης ευρετηρίων για διάφορα πλήθη block κύριας μνήμης. Για αρχεία με καινούριους κόμβους και αρχεία χωρις καινούριους κόμβους (σταθ. V).

τμημάτων σε σχέση με τα προηγούμενα πειράματα έγινε επειδή έχουμε μικρές ΣΜ, θέλουμε να διαβάζουμε όσο δυνατόν περισσότερους κοινούς κόμβους με ένα πέρασμα.

Το σχήμα 5.9 δείχνει τον χρόνο που απαιτείται για την ενημέρωση των ευρετηρίων.



Σχήμα 5.9: Χρόνοι ενημέρωσης ευρετηρίων για διάφορες χωρητικότητες των τμημάτων μνήμης σε ζευγάρια. Για αρχεία με καινούριους κόμβους και αρχεία χωρις καινούριους κόμβους (σταθ. V).

Παρατηρήσεις: Καθώς αυξάνεται ο αριθμός των ζευγαριών μειώνεται ο χρόνος κατασκευής για τα ευρετήρια διότι αυξανεται η διαθέσιμη κύρια μνήμη. Ολα αυτά μέχρι εναν αριθμό τμημάτων. Περα απο αυτό μεγαλύτερη αυξηση των ζευγαρίων δεν προσφέρει κέρδος σε χρόνο γιατί έχουν δωθεί περισσότερα από όσα χρειάζονται. Στα συγκεκριμένα πειράματα χρησιμο-

ποιούμε μικρό αριθμό ζευγαριών γιατί δεν έχουμε πολλούς κοινούς κόμβους στα δεδομένα μας (περισσότερους κοινούς κόμβους στο αρχείο ενημέρωσης με σταθερό αριθμό κόμβων).

Κεφάλαιο 6

Τεχνικές Λεπτομέρειες

Στα προηγούμενα κεφάλαια περιγράψαμε τους αλγόριθμους για την αποδοτική κατασκευή και ενημέρωση των ευρετηρίων *P-Index* και *L-Index*. Σε αυτό το κεφάλαιο παρουσιάζουμε τις τεχνικές λεπτομέρειες της υλοποίησης.

6.1 Λεπτομέρειες υλοποίησης

6.1.1 Δομές της $C++$ που χρησιμοποιήσαμε

Για την υλοποίηση των απαραίτητων για την διπλωματική αυτή εργασία έγινε χρήση των παρακάτω δομών της $C++$ (Standard Template Library) και των συναρτήσεων που αυτές υποστηρίζουν:

- `unordered_map`: Μη ταξινομημένο ευρετήριο που περιέχει μοναδικα ζευγαρια κλειδί-τιμή
- `vector`: Υλοποίηση διατεταγμένης λίστας από στοιχεία με δυνατότητα τυχαίας πρόσβασης
- `list`: Υλοποίση λίστας στοιχείων

6.1.2 Κλάση `Graph`

Η κλάση αυτή αντιπροσωπεύει την Συλλογή Μονοπατιών που είναι αποθηκευμένη στο δίσκο. Πρόκειται για τη μεταφορά του αρχείου εισόδου, έτσι όπως το διαβάζουμε.

Οι βασικές μεθόδοι της `Graph` που χρησιμοποιήσαμε είναι οι:

- `int* Graph::getPath(int pid)`
  Η συνάρτηση επιστρέφει ένα μονοπάτι της Συλλογής Μονοπατιων από το δίσκο με βαση το δοθέν αναγνωριστικό μονοπατιού.
- `int Graph::addPath(Path path)`
  Προσθέτει μία εγγραφή μονοπατιου `path` στη Συλλογή Μονοπατιών στο δίσκο.

Το βασικό πεδίο της `Graph` που χρησιμοποιήσαμε είναι η

- `int numPaths`
  Ο αριθμός μονοπατιών στη Συλλογή Μονοπατιών στο δίσκο

6.1.3 Κλάση `path_index`

Η κλάση αυτή αντιπροσωπεύει το ευρετήριο *P-Index* που είναι αποθηκευμένο στο δίσκο. Οι βασικές μεθόδοι της `path_index` που χρησιμοποιήσαμε είναι οι:

- `int PathIndex::addNode(int nid, int lsize, int *paths, int *positions)`
  Προσθέτει μία εγγραφή κόμβου του *P-Index* στο δίσκο, δοθέντος αναγνωριστικού κόμβου, συνολικού μεγέθους εγγραφής, και της λίστας των μονοπατιών και των θέσεων.

- `int PathIndex::delNode(int nid)`
  Διαγράφει μία εγγραφή κόμβου του *P-Index* στο δίσκο δοθέντος αναγνωριστικού κόμβου.

- `int PathIndex::getPathList(int nid, int **paths, int **positions)`
  Επιστρεφει, δοθέντος αναγνωριστικού κόμβου, μία εγγραφή κόμβου του *P-Index* από το δίσκο δηλαδή το συνολικό μεγεθος της εγγραφής, τη λίστα των μονοπατιών και τη λίστα των θέσεων.

Το βασικό πεδίο της `path_index` που χρησιμοποιήσαμε είναι η

- `int numNodes`
  Ο αριθμός κόμβων στο ευρετήριο *P-Index* στο δίσκο και επομένως στη Συλλογή Μονοπατιών στο δίσκο

6.1.4 Κλάση `link_nodes_index`

Η κλάση αυτή αντιπροσωπεύει το ευρετήριο *L-Index* που είναι αποθηκευμένο στο δίσκο. Οι βασικές μεθόδοι της `link_nodes_index` που χρησιμοποιήσαμε είναι οι:

- `int LinkNodesIndex::addPath(int nid, int lsize, int *lnodes, int *positions)`
  Προσθέτει μία εγγραφή μονοπατιού του *L-Index* στο δίσκο, δοθέντος αναγνωριστικού μονοπατιού, συνολικού μεγέθους εγγραφής, και της λίστας των κόμβων και των θέσεων.

- `int LinkNodesIndex::delPath(int nid)`
  Διαγράφει μία εγγραφή μονοπατιού του *L-Index* στο δίσκο δοθέντος αναγνωριστικού μονοπατιού.

- `int LinkNodesIndex::getLinkNodesList(int nid, int **nodes, int **positions)`
  Επιστρεφει, δοθέντος αναγνωριστικού μονοπατιού, μία εγγραφή μονοπατιού του *L-Index* από το δίσκο δηλαδή το συνολικό μεγεθος της εγγραφής, τη λίστα των κόμβων και τη λίστα των θέσεων.

6.1.5 Κλάση `pathpospair`

Η κλάση αυτή αντιπροσωπεύει ένα ζευγαρι από ακέραιους που αναπαριστούν ένα ζευγαρι μονοπατιού-θέσης για τον *P-Index* και κόμβου-θέσης για τον *L-Index*.

6.1.6 Δομή `BlockManager`

Υλοποιήσαμε αυτή τη δομή η οποία διαχειρίζεται τον χωρισμό, τη δέσμευση και την απελευθέρωση της μνήμης για την αποδοτικη κατασκευή και ενημέρωση των ευρετηρίων.

Οι βασικές μεθόδοι της `BlockManager` είναι:

- `BlockManager(int Nblocks,int Mpairs)`
  [Constructor] Δέχεται ως εισόδο τον αριθμό των τμημάτων μνήμης που χρειαζόμαστε να δεσμεύσουμε και των αριθμό των ζευγαρίων που θέλουμε να περιέχει το καθένα.

- `int getCurrentBlockID()`
  Επιστρέφει το αναγνωριστικό του τρέχοντος διαθέσιμου τμήματος μνήμης και ορίζει ως τρέχον το επόμενο διαθέσιμο τμήμα. Επιστρέφει -1 αν δεν υπάρχει άλλο διαθέσιμο τμήμα μνήμης.

- `void clear()`
  Η μέθοδος αυτή ξανακάνει διαθέσιμα όλα τα τμήματα μνήμης.

6.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Για την υλοποίηση των μεθόδων της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμμάτισμου $C++$. Ο μεταγλωτιστής ήταν ο GNU g++ v.4.2.0. Ως περιβάλλον εργασίας χρησιμοποιήθηκε το EasyEclipse σε λειτουργικό σύστημα Linux (διανομή Ubuntu 8.10). Ο υπολογιστής είχε μνήμη 1GB και επεξεργαστή Intel Pentium 4 στα 3.2GHz. Οι μετρήσεις παραστάθηκαν με λογιστικό φύλλο OpenOffice και για την σχεδίαση της Συλλογής Μονοπατιών και των ευρετηρίων χρησιμοποιήθηκε το πρόγραμμα Graphviz.

Κεφάλαιο 7

Επίλογος

Στο κεφάλαιο αυτό θα κάνουμε μια σύνοψη των όσων αναφέραμε στη διπλωματική εργασία και θα αναφέρουμε τα συμπεράσματα στα οποία καταλήξαμε. Επιπλέον θα προτείνουμε διάφορες ιδέες που μπορούν να επεκτείνουν την μέχρι τώρα εργασία μας.

7.1 Σύνοψη και συμπεράσματα

Στην διπλαματική αυτή εργασία αντιμετωπίσαμε το πρόβλημα της υλοποίησης μηχανισμών κατασκευής και ενημέρωσης ευρετηρίων για μεγάλες Συλλογές Μονοπατιών, που δεν μπορούν να κατασκευαστούν πλήρως στην περιορισμένου χώρου κύρια μνήμη. Συγκεκριμένα υλοποιήσαμε τις μεθόδους κατασκευής:

1. Κατασκευή ευρετηριου *P-Index*.
2. Κατασκευή ευρετηρίου *L-Index*.

Και τις μεθόδους ενημέρωσης:

1. Ενημέρωση ευρετηρίου *P-Index*.
2. Ενημέρωση ευρετηρίου *L-Index*.

Συμπεράσματα: Όλες οι μέθοδοι που υλοποιήσαμε τόσο για την κατασκευή όσο και για την ενημέρωση των ευρετηρίων με βάση τα πειραματικά δεδομένα είναι εφαρμόσιμες σε λογικά χρονικά περιθώρια. Οι μηχανισμοί που υλοποιήθηκαν στην παρούσα διπλωματική εργασία είναι σαφώς πιο χρονοβόροι από τους μηχανισμούς που κατασκευάζουν καΆ ενημερώνουν τα ευρετήρια εξολοκλήρου στη μνήμη. Ομως απευθύνονται σε πολύ μεγάλο όγκο δεδομένων όπου δεν υπάρχει αυτή η επιλογή. Η αύξηση σε χρονικές απαιτήσεις όταν αυξάνεται το μέγεθος των ευρετηρίων είναι η αναμενόμενη. Ολα αυτά οδηγούν στο συμπέρασμα ότι οι μέθοδοι αυτοί μπορούν να εφαρμοστούν χωρίς υπερβολικά μεγάλο κόστος σε χρόνο.

Βελτιώσεις: Θα είναι πολύ χρήσιμη η πειραματική εφαρμογή των μεθόδων μας σε συλλογές με μονοπάτια με μεγάλο αριθμό κοινών κόμβων. Με αυτό τον τρόπο οι μέθοδοι εκμεταλλεύονται πλήρως τους μηχανισμούς κατασκευής και ενημέρωσης των ευρετηρίων.

7.2 Μελλοντικές επεκτάσεις

Στην ενότητα αυτή θα προτείνουμε μερικές ιδέες για την επέκταση της μέχρι τώρα δουλείας μας. Οι ιδέες αυτές αφορούν δύο θέματα:

1. Την επεκταση της διαδικασίας ενημέρωσης
2. Μελλοντική εφαρμογή των μεθόδων και σε άλλα ευρετήρια

7.2.1 Επέκταση διαδικασίσς ενημέρωσης

Στην παρούσα διπλωματική αντιμετωπίσαμε το προβλημα ενημέρωσης για προσθήκη νέων μονοπατιών. Η επεκταση της διαδικασίας ενημέρωσης και στην αφαίρεση μονοπατιών είναι ένα πολύ ενδιαφέρον πρόβλημα και θα ολοκληρώσει την διαδικασία ενημέρωσης.

7.2.2 Εφαρμογή μεθόδων και σε άλλα ευρετήρια

Η απάντηση και άλλων ερωτημάτων σε δεδομένα με τη μορφή συλλογών αντικειμένων οδηγεί στη δημιουργία και άλλων μορφών ευρετηρίων για την αποδοτική πρόσβαση στα δεδομένα των συλλογών. Οι μέθοδοι αυτής της διπλωματικής εργασίας μπορούν να εφαρμοστούν στην κατασκευή και ενημέρωση και άλλου είδους ευρετηρίων.

Βιβλιογραφία

[1] Lenia Boula. Graphit-db: Graph data management system (1). Diploma thesis, Knowledge and Database Systems Laboratory, School of Electrical and Computer Engineering, NTUA, 2007.

[2] Panagiotis Bouros, Spiros Skiadopoulos, Theodore Dalamagas, Dimitris Sacharidis και Timos K. Sellis. Evaluating reachability queries over path collections. Στο *SSDBM*, σελίδες 398–416, 2009.