Project Εξαμήνου

Δομές Δεδομένων

Δουράτσος Ιωάννης *4978

Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Πατρών

September 25, 2013

*douratsos@ceid.upatras.gr

Contents

1 Γενικές Πληροφορίες

Η υλοποίηση του project έχει γίνει σε python.Το σύστημα στο οποίο δοκιμάστηκε ήταν ubuntu 13.04 . Απαραίτητη είναι και η ύπαρξη του matplotlib module καθώς αυτό αναλαμβάνει το plotting των διαγραμμάτων χρόνου. Αν εξαιρεθεί το plotting, οποιοσδήποτε python interpreter αρκεί. Για την ευκολότερη εκτέλεση των test, υπάρχουν δύο εναρκτήρια αρχεία: το setup.sh και το create.py. Το setup.sh αναλαμβάνει την εγκατάσταση των module που είναι αναγκαία και το create.py φτιάχνει μια μικρή βάση από βιβλία τα οποία και θα χρησιμοποιηθούν ως δείγμα για την αναζήτηση πάνω στη πραγματική βάση, δηλαδή το books.csv. Τα Book, Author και Books έχουν όλα υλοποιηθεί ως κλάσεις, ενώ συγκεκριμένα το Books περιέχει μια list από book και όλες τις κατάλληλες συναρτήσεις για αυτά.
Αν για κάποιο λόγο η εκτέλεση των main.py, test.py, create.py οδηγήσει σε οποιοδήποτε πρόβλημα, προτείνεται η εκτέλεση του setup.sh. Αυτό θα πάρει τη τελευταία έκδοση των αρχείων από το github ώστε να βεβαιώσει την απροβλημάτιστη εκτέλεσης τους. Τέλος, για ευκολότερη ανάγνωση των module, υπάρχει online documentation μέσω του sphinx, του επίσημου εργαλείου για δημιουργία documentation σε python modules. Το documentation βρίσκεται στη σελίδα μου στο ceid students: http://students.ceid.upatras.gr/~douratsos/data_structs/index.html.

2 Μέρος Α

Για αυτό το ερώτημα εκτυπώνεται ένα menu από το οποίο ο χρήστης μπορεί να διαλέξει την συνάρτηση που θέλει να εκτελέσει πάνω στα Books. Όπως είπαμε και πριν, όλα τα book βρίσκονται στη μνήμη σε μια list άπαξ και γίνουν load μέσω της επιλογής 1.
Όλα αυτά έχουν υλοποιηθεί στο module main.py το οποίο χρησιμοποιεί το bookservice.py, οπότε για το μέρος Α αρκεί η εκτέλεση του main.py. Η επιλογή αυτή απαιτεί ένα αρχείο σε μορφή csv, ενώ από default διαβάζει το αρχείο books.csv που βρίσκεται στον ίδιο φάκελο και είναι αυτό που είχε δωθεί στο forum του μαθήματος. Όπου χρειάζεται αναζήτηση, το κάνουμε με μια for πάνω στη λίστα ώστε να έχουμε γραμμικό χρόνο αναζήτησης.

3 Μέρος Β

Χρησιμοποιείται η κλασική μέθοδος δυαδικής αναζήτησης, η οποία είναι υλοποιημένη στη συνάρτηση binary_search μέσα στη κλάση Books. Για τη χρήση της δυαδικής αναζήτησης υπάρχει η ανάγκη να διατηρείται η λίστα των βιβλίων ταξινομημένη στο σύστημα μας, για αυτό δημιουργούμε ένα ταξινομένο αντίγραφο της, την arrsorted. Σε κάθε insert/delete γίνεται ξανά sorting της λίστας.

4 Μέρος Γ

Τα digital tries βρίσκονται υλοποιημένα στο module trie.py. Θα χρειαστούμε δύο διαφορετικά trie για το μέρος αυτό, ένα για να οργανώσουμε με βάση τον τίτλο και ένα για να οργανώσουμε τη βάση με βάση το επίθετο του συγγραφέα. Στην περίπτωση του δεύτερου, επιστρέφεται ως αποτέλεσμα μια λίστα από όλα τα βιβλία στα οποία αναφέρεται κάποιος ως συγγραφέας, είτε μόνος του είτε ως συνεργάτης με άλλους συγγραφείς.

5 Μέρος Δ

Στο module AVL.py είναι υλοποιημένο ένα κομβοπροσανατολισμένο AVL δέντρο οργανωμένο με βάση τα ids των βιβλίων.Σε κάθε πράξη ελέγχεται η ζύγιση του υποδέντρου και αν έχει χαλάσει γίνονται κατάλληλες επαναζυγιστικές κινήσεις. Σε αυτο πρόσθέτουμε τη βάση των βιβλίων. Προφανώς για την αρχικοποίηση του δέντρου πρέπει να "περάσουμε" όλη τη βάση ακριβώς μια φορά.

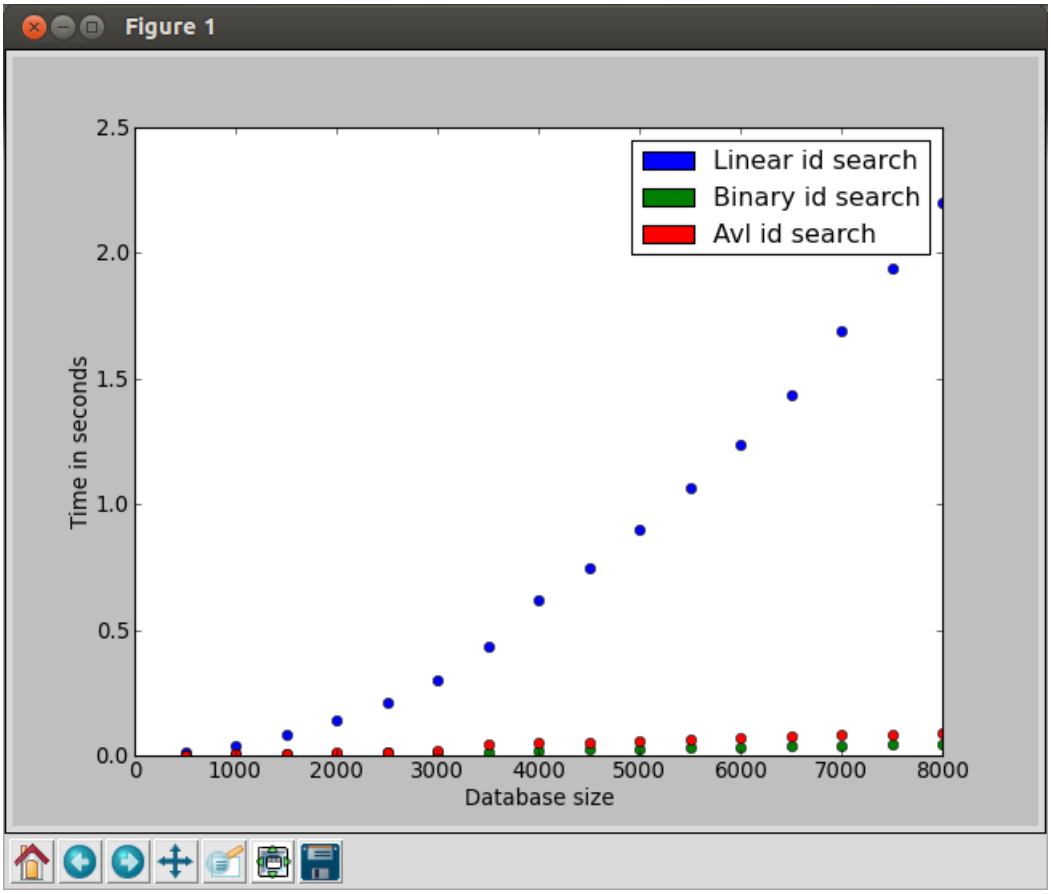
6 Μέρος Ε

Για το ερώτημα αυτό χρησιμοποιείται το αρχείο test.py σε συνδιασμό με το create.py. Το αρχείο create.py αρχικά δημιουργεί μια λίστα από βιβλία τα οποία και θα αναζητήσουμε και το αποθηκεύει μέσω του pickle module για εύκολη αργότερη χρήση. Έτσι στο test.py χρησιμοποιούμε αυτή τη λίστα από βιβλία δείγματα ώστε να αναζητήσουμε πάνω στη βάση των βιβλίων. Το μέγεθος της βάσης το μεταβάλουμε ώστε να δούμε την απόδοση των αλγορίθμων ανάλογα με το μέγεθος και για κάθε μέγεθος βάσης παίρνουμε το μέσο χρόνο αναζήτησης.
Για να είναι πιο αντικειμενικές οι μετρήσεις, όσο μεγαλύτερη είναι η βάση μας κάθε φορά τόσες περισσότερες αναζητήσεις εκτελούμε πάνω της. Τελικά χρησιμοποιούμε το module matplotlib ώστε να οπτικοποιήσουμε αυτά τα αποτελέσματα και να είναι πιο εύκολο να καταλάβουμε την διαφορά της απόδοσης τους ανάλογα με το μέγεθος της βάσης που χρησιμοποιούμε. Συνολικά η διαδικασία που κάνουμε για να εκτελέσουμε πλήρως το ερώτημα αυτό είναι

- Εκτέλεση του module create.py
  Αυτό δημιουργεί τη λίστα που αναφέραμε παραπάνω, που περιέχει το ψευδοτυχαίο δείγμα των αναζητήσεων μας
- Εκτέλεση του module test.py
  Η εκτέλεση αυτή μπορεί να γίνει είτε ώς python test.py είτε ώς python test.py -p, με τη δεύτερη να προτείνεται αν υπάρχει εγκατεστημένη η matplotlib στο σύστημα καθώς θα αναλάβει να φτιάξει και performance charts για τους αλγορίθμους.

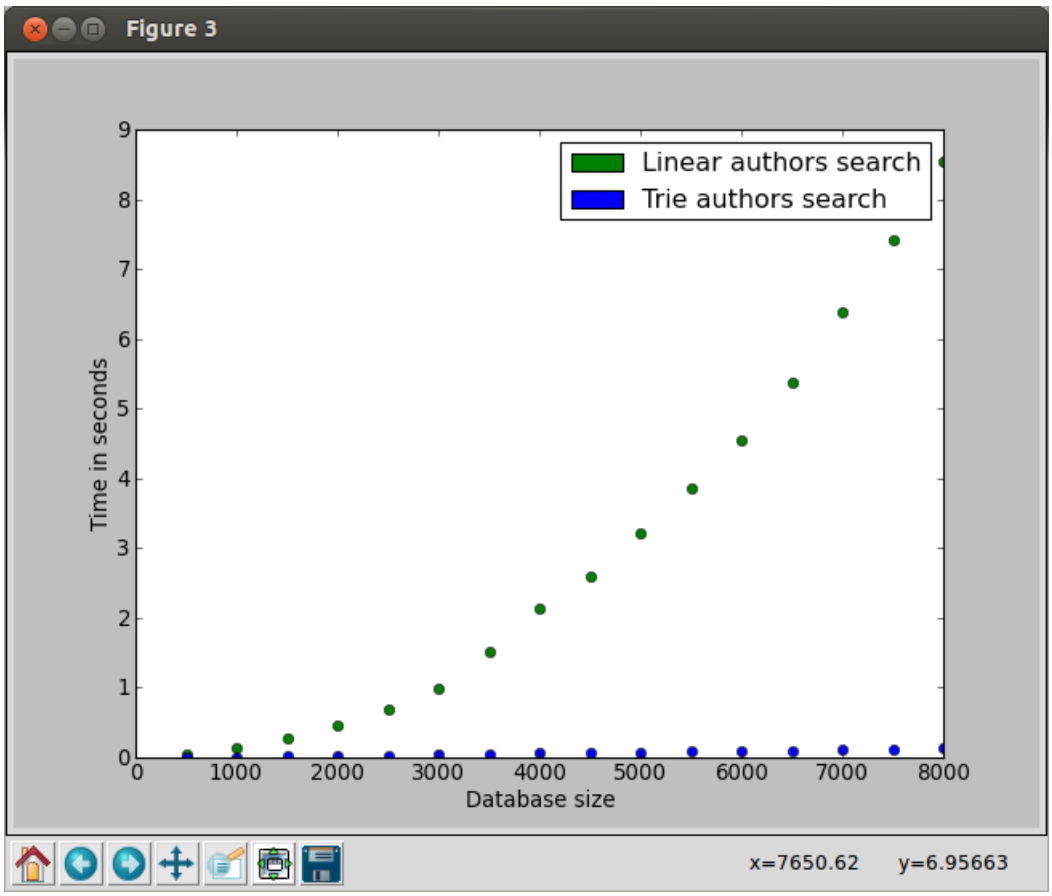
Παρατηρούμε ότι η απόδοση των γραμμικών αναζητήσεων πέφτει πολύ γρήγορα όσο μεγαλώνει το μέγεθος της βάσης, σε αντίθεση με τις αναζητήσεις που έχουν ως βάση AVL ή TRIES καθώς και την binary search. Αυτές διατηρούν πολύ καλή απόδοση ακόμη και παρά τη μεγάλη αύξηση του μεγέθους της βάσης των βιβλιών. Αυτό φαίνεται και στα screenshot που παρατήθονται και κάνουν σύγκριση των χρόνων αναζήτησης ανάμεσα στις διαφορετικές υλοποιήσεις.

6.1 Αναζήτηση με βάση το ID



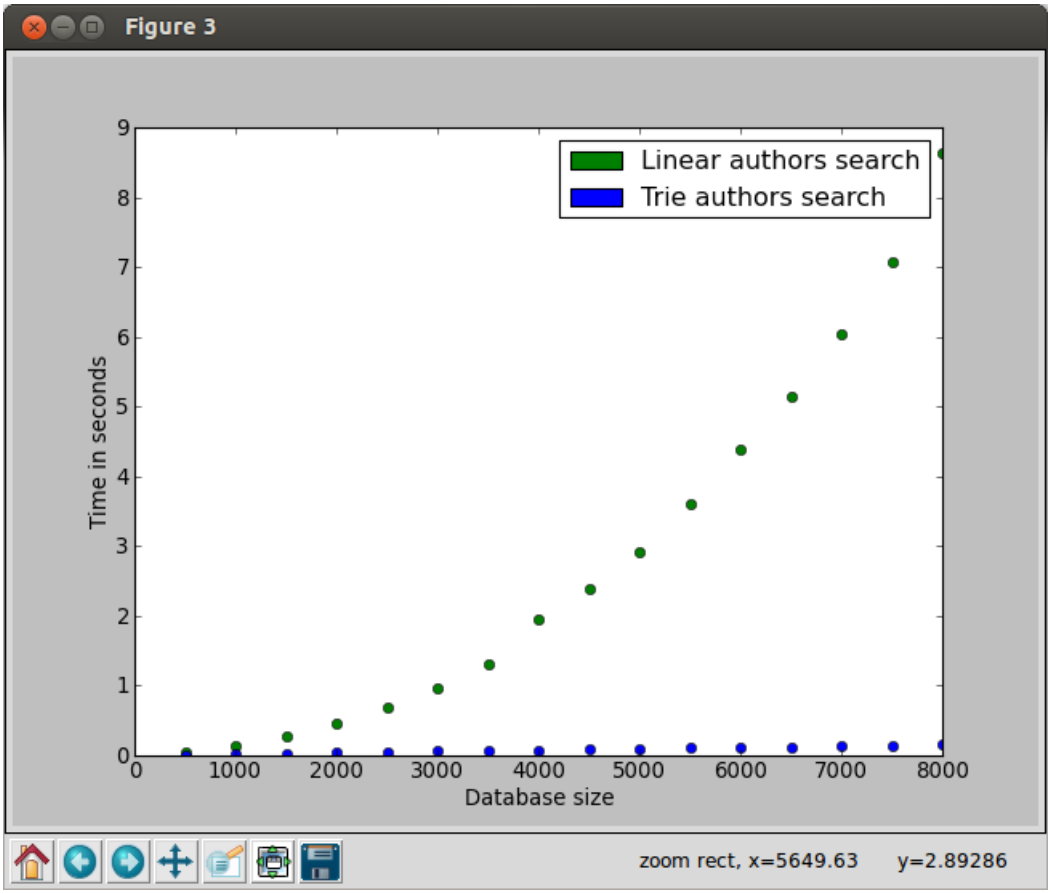
Εδώ παρατηρούμε ότι όπως αναμέναμε η γραμμική αναζήτηση αυξάνει σχεδόν γραμμικά με την αύξηση του μεγέθους της βάσης των βιβλίων. Προφανώς και πάλι υπάρχουν κάποιες αυξομειώσεις καθώς κάποιες αναζητήσεις μπορεί να σταθούν "τυχαιρές" και έτσι δεν έχουμε τη γραφική παράσταση μιας τέλειας ευθείας. Αντίθετα, τόσο η δυαδική αναζήτηση όσο και η αναζήτηση με το AVL δέντρο κινούνται σε λογαριθμικούς χρόνους και η αύξηση του χρόνου αναζήτησης με αυτές τις δύο είναι μικρή καθώς αυξάνει η βάση των βιβλιών μας. Αυτό συμφωνέι με τη θεωρία που μας λέει ότι η δυαδική αναζήτηση έχει λογαριθμικό μέσο χρόνο αναζήτησης. Το ίδιο complexity έχει και το AVL, το οποίο και αυτό επαληθεύεται μέσω της γραφικής αυτής παράστασης.

6.2 Αναζήτηση με βάση το title



Εδώ συγκρίνουμε τη γραμμική αναζήτηση μέσω τίτλου με την αναζήτηση μέσω ψηφιακών δέντρων (TRIES). Βλέπουμε ότι ενώ η γραμμική αναζήτηση αυξάνει τους χρόνους της όσο αυξάνεται η βάση των βιβλίων, η αναζήτηση μέσω των TRIE μένει σχεδόν σταθερή.
Αυτό δεν πρέπει να μας εκπλήσει, καθώς η αναζήτηση μέσω TRIES εξαρτάται κυρίως από το μέγεθος του εκάστοτε τίτλου και όχι άμεσα από τη βάση των βιβλίων. Αυτό έχει ως αποτέλεσμα κάποια entries, ιδιαίτερα αυτά που ξεκινούν από κάποιο πιο "σπάνιο" prefix να έχουν πολύ γρήγορους χρόνους αναζήτησης. Μπορούμε να πούμε ότι γενικά τα TRIE έχουν πάρα πολύ καλή απόδοση στην αναζήτηση των βιβλίων, που δεν εξαρτάται τόσο από το μέγεθος της βάσης πάνω στην οποία αναζητούμε αλλά μόνο από την ίδια τη πληροφορία των βιβλίων. Αυτό θα το επαληθεύσουμε και με το παρακάτω τελευταίο πείραμα.

6.3 Αναζήτηση με βάση τους author



Εδώ βλέπουμε τη διαφορά στις αναζητήσεις με βάση το επώνυμο. Λόγω του ότι το επώνυμο δεν είναί ένα μοναδικό κλειδί για τη βάση μας, απαιτείται να προσπελάσουμε πάντα όλη τη λίστα των βιβλίων, ακόμη και εαν το πρώτο βιβλίο που προσπελάσουμε είναι γραμμένο από το συγγραφέα που αναζητούμε.
Η γραφική αυτή παράσταση δεν έχει ακριβώς γραμμική μορφή λόγω της επιπλέον πολυπλοκότητας που έχει προστεθεί ώστε να μπορούμε να γυρίζουμε μια λίστα με authors. Παρατηρούμε επίσης ότι η διαφορά στους χρόνους τώρα είναι πολύ μεγαλύτερη από ότι πριν. Αυτό συμβαίνει λόγω ακριβώς αυτού που περιγράψαμε παραπάνω: πρέπει κάθε φορά να προσπελαύνουμε ολόκληρη τη λίστα των βιβλίων. Αντίθετα, το TRIE δεν επιρρεάζεται καθόλου από αυτό, καθώς κατά τη κατασκευή του, μέσω της συνάρτησης build_authors έχει αποθηκευθεί σε κάθε key μια list από τα βιβλία που αντιστοιχούν σε κάθε author.
Με αυτό το τρόπο, στο TRIE πληρώνουμε λίγη έξτρα πολυπλοκότητα για την αρχικοποίηση του, αφού πρέπει να περάσουμε μία φορά τη λίστα, αλλά κερδίζουμε πάρα πολύ στο ότι έχουμε όλα τα βιβλία του κάθε author προσβάσιμα στο χρόνο που χρειαζόμαστε να για φτάσουμε σε ένα οποιοδήποτε κλειδί του trie. Ο χρόνος όμως αυτός εξαρτάται μόνο από το μήκος του κλειδιού και καθόλου από το μέγεθος της βάσης των βιβλίων. Αυτό έχει ως αποτέλεσμα ο μέσος χρόνος που απαιτείται για τις αναζητήσεις να μένει σχεδόν σταθερός, και να φαντάζει σχεδόν μηδενικός όταν συγκριθεί στο γράφημα με το χρόνο της γραμμικής αναζήτησης.

6.4 Παρατηρήσεις για τα γραφήματα

Τα γραφήματα βλέπουμε ότι δεν επαληθεύουν πλήρως τη θεωρητική τιμή. Όμως, αυτό μπορεί να οφείλεται τόσο στο ότι τα δείγματα μας δεν είναι πραγματικά τυχαία, αλλά φτιάχνουνται από εμάς μέσω μιας βάσης

όσο και σε κάποια έξτρα πολυπλοκότητα που προστέθηκε στο πογραμμα μας για τη μεγαλύτερη ευκολία μας όσο είχε να κάνει με τη βάση των βιβλίων.
Παρόλα αυτά μπορούμε να πούμε ότι με λίγη ελαστικότητα επαληθεύονται οι γνώσεις μας από τη θεωρία. Σημαντικό είναι πως για το TRIE ιδιαίτερα, λόγω της μεγάλης διαφοροποίησης του με τη γραμμική αναζήτηση, οι χρόνοι φαντάζουν μηδενικοί. Όμως, αν παρατηρήσουμε τους απόλυτους χρόνους που βγάζει το πρόγραμμα στη κονσόλα, τότε θα δούμε ότι υπάρχει μια διακύμανση και σε αυτούς, απλά είναι η διαφορά τους με τη γραμμική αναζήτηση είναι τόσο μεγάλη που δε μπορεί να απεικονιστεί αυτή η διακύμανση σε αυτό το γράφημα.

7 References

- https://github.com/pgrafov/python-avl-tree/blob/master/pyavltree.py
- http://en.wikipedia.org/wiki/Trie