Αυτορυθμιζόμενα Δυαδικά Δένδρα Αναζήτησης

Βησσαρίων Φυσικόπουλος

16 Ιανουαρίου 2008

Περίληψη

Στην παρούσα εργασία ασχολούμαστε με το online πρόβλημα της αναζήτησης στοιχείων σε δυαδικά αυτορυθμιζόμενα δένδρα αναζήτησης και το εξετάζουμε από τη σκοπιά της θεωρίας ανταγωνισμού. Online είναι ένα πρόβλημα όταν δεν γνωρίζουμε την είσοδο εξ αρχής αλλά μας δίνεται σταδιακά. Παρουσιάζουμε τον move-to-root αλγόριθμο που είναι $\Theta(n)$-competitive. Μια δομή δεδομένων που προτείνεται στο [4] είναι τα splay δένδρα. Τα δένδρα αυτά έχουν ιδιότητες πολλών γνωστών δένδρων αναζήτησης αλλά δεν έχει αποδεικτεί ότι είναι καλύτερα από τα ισοζυγισμένα δένδρα αναζήτησης. Η εικασία είναι ότι τα splay δένδρα είναι $O(1)$-competitive.

1 Εισαγωγή

Τα δυαδικά δένδρα αναζήτησης αποτελούν ένα πολύ διαδεδομένο είδος *δομών δεδομένων* με πλήθος εφαρμογών. Μια υποκατηγορία τους, τα ισοζυγισμένα δυαδικά δένδρα αναζήτησης υποστηρίζουν *πράξεις* επί του συνόλου των n στοιχείων τους, amortized χρονικής πολυπλοκότητας $O(\log n)$. Στην περίπτωση όμως που το πρόβλημα αναζήτησης είναι άμεσο (online) τα δένδρα αυτά δεν είναι βέλτιστα.

1.1 Περιγραφή του Προβλήματος

Στην παρούσα εργασία θα ασχοληθούμε με το πρόβλημα της αναζήτησης σε δυαδικά αυτορυθμιζόμενα δένδρα αναζήτησης. Εξετάζουμε το πρόβλημα από τη σκοπιά της θεωρίας ανταγωνισμού όπου η απόδοση ενός online αλγορίθμου συγκρίνεται με την απόδοση του βέλτιστου offline. Πιο συγκεκριμένα, δεδομένης μιας άγνωστης ακολουθίας αιτήσεων σε ένα δυαδικό δένδρο αναζήτησης μας ενδιαφέρει ο συνολικός χρόνος της εύρεσης των στοιχείων (αιτήσεων) στο δένδρο. Όσο πιο μικρή είναι η απόσταση (μήκος μονοπατιού) ενός στοιχείου

από τη ρίζα τόσο πιο γρήγορη είναι η εύρεση του. Ένας αλγόριθμος είναι μετά την αίτηση του στοιχείου i να μετακινούμε το i στην ρίζα. Για τη μετακίνηση του στοιχείου στη ρίζα μπορούμε να χρησιμοποιήσουμε τις πράξεις ισοζυγίσματος (απλή (Σχήμα 1 a) και διπλή περιστροφή (Σχήμα 1 c)) που χρησιμοποιούν τα ισοζυγισμένα δένδρα που όπως θα δούμε δεν είναι καλή λύση από τη σκοπιά της θεωρίας ανταγωνισμού. Η λύση που προτάθηκε στο [4] χρησιμοποιεί πράξεις που πραγματοποιούν περιστροφές ζευγαριών για να μετακινήσει ένα στοιχείο στη ρίζα. Οι πράξεις αυτές ονομάζονται *splay* πράξεις και το δένδρο που τις υποστηρίζει *splay* δένδρο. Το [1] κάνει μια επισκόπηση στις αυτορυθμιζόμενες δομές δεδομένων και εξετάζει και τα splay δένδρα.

1.2 Ιστορική Αναδρομή

Τα ισοζυγισμένα δυαδικά δένδρα είναι υψοζυγισμένα όπως το AVL-δένδρο (1962), Κόκκινο-Μαύρο δένδρο (1978), BB-δένδρο (1983) και βαροζυγισμένα όπως το BB[a] (1973) δένδρο. Το 1985 οι Sleaton και Tarjan προτείνουν το splay δένδρο το οποίο χρησιμοποιεί splay πράξεις όχι για ισοζύγισμα αλλά για να επιτυγχάνει ικανοποιητικές εξυπηρετήσεις νέων αιτήσεων. Το splay δένδρο εικάζεται ότι έχει σταθερό *λόγο ανταγωνισμού (competitive ratio)* αντίθετα με τα γνωστά ισοζυγισμένα δυαδικά δένδρα. Η εικασία (Dynamic Optimality Conjecture) όμως αυτή δεν έχει αποδειχθεί ούτε απορριφθεί μέχρι σήμερα και παραμένει το πιο βασικό ανοιχτό πρόβλημα στην περιοχή.

Η αντίστοιχη έρευνα πάνω στο πρόβλημα έχει δημιουργήσει μια σειρά από συσχετιζόμενες με την βασική εικασίες αλλά και κάποια αποτελέσματα σχετικά με υποπεριπτώσεις του γενικού προβλήματος εξετάζοντας ειδικές περιπτώσεις ακολουθιών αιτήσεων. Επίσης ο Cole στα [2, 3] επέδειξε την εικασία του dynamic finger.

Στο [7] παρουσιάζεται μια παραλλαγή των splay δένδρων τα multi-splay δένδρα τα οποία έχουν λόγο ανταγωνισμού $O(\log\log n)$ και amortized χρονική πολυπλοκότητα $O(\log n)$.

1.3 Σχετικές Δημοσιεύσεις

Στη βασική δημοσίευση των Sleaton και Tarjan [4] αποδεικνύεται ότι ο αποσβησμένος χρόνος των splay πράξεων σε ένα δυαδικό δένδρο n κόμβων είναι $3\log n + 1$ που συνεπάγεται λόγο ανταγωνισμού των splay δένδρων $O(\log n)$. Επίσης αποδεικνύονται μια σειρά από *ιδιότητες* του splay δένδρου.

Balanced Το splay δένδρο μετά από μια σειρά από προσπελάσεις, είναι τόσο αποδοτικό όσο και ένα ισοζυγισμένο δυαδικό δένδρο.

Static Optimality Το splay δένδρο δεδομένης μιας ακολουθίας αιτήσεων είναι τόσο αποδοτικό όσο το βέλτιστο στατικό δυαδικό δένδρο.

Static Finger Για ένα σταθεροποιημένο στοιχείο s το splay δένδρο υποστηρίζει προσπελάσεις στην γειτονική περιοχή του s με την ίδια αποδοτικότητα με τα finger δένδρα αναζήτησης*.

Working Set Ο χρόνος προσπέλασης ένος στοιχείου i σε ένα splay δένδρο είναι ίσος με το λογάριθμο του 1 συν το πλήθους των διαφορετικών στοιχείων που είχαν προσπελαστεί μεταξύ των δύο τελευταίων προσπελάσεων του i. Μια άλλη σκοπιά από την από την οποία μπορούμε να εξετάσουμε αυτή την ιδιότητα συνοψίζεται στην εξής παρατήρηση: Όταν ένα στοιχείο προσπελαστεί το κόστος της επόμενης προσπέλασης του αυξάνει λογαριθμικά με το πλήθος των διαφορετικών στοιχείων που προσπελαύνονται.

1.4 Ανοικτά Προβλήματα

Η βασική εικασία για τα splay δένδρα αποτελεί και το κύριο ανοικτό πρόβλημα.

Εικασία 1.1 (Dynamic Optimality [4]) *Το splay δένδρο είναι έχει λόγο ανταγωνισμού* $O(1)$.

Η άλλη εικασία που είναι υποπερίπτωση της προηγούμενης αλλά παραμένει επίσης ανοικτή είναι η εξής.

Εικασία 1.2 (Traversal [4]) *Έστω* T_1, T_2 *δύο δυαδικά δένδρα αναζήτησης με* n *ακριβώς ίδιους κόμβους το καθένα. Έστω ότι προσπελαύνουμε τους κόμβους του* T_1 *με βάση την προδιάταξη τους στο* T_2, *και εφαρμόζουμε σε αυτούς τις splay πράξεις τότε ο συνολικός χρόνος προσπέλασης είναι* $O(n)$.

Έστω η αναπαράσταση μιας διπλής ουράς Q από ένα δυαδικό δένδρο T στο οποίο ο i-οστός κόμβος του σε συμμετρική διάταξη είναι το i-οστό στοιχείο της ουράς Q. Χρησιμοποιώντας splay πράξεις για να υλοποιήσουμε τις πράξεις της Q έχουμε (ορίζουμε την εισαγωγή και και εξαγωγή από τη μια άκρη της ουράς. Η άλλη περίπτωση είναι συμμετρική.) :

POP Εφαρμόζουμε splay πράξεις στο μικρότερο κόμβο του T και τον αφαιρούμε από το T.

*Ένα δακτυλοδεικτούμενο (finger) δένδρο αναζήτησης αποθηκεύει μια λίστα στοιχείων στα φύλλα του και οι ενδιάμεσοι κόμβοι είναι βοηθητικοί. Έχει την ιδιότητα να επιτυγχάνει γρήγορες αναζητήσεις κοντά στο τέλος ή στην αρχή της λίστας. Ειδικότερα η αναζήτηση ενός στοιχείου που απέχει απόσταση d από την αρχή ή το τέλος της λίστας γίνεται σε χρόνο $O(\log d)$.

PUSH Ο κόμβος που εισάγεται γίνεται η νέα ρίζα του T με αριστερό παιδί κενό και δεξί παιδί την παλιά ρίζα.

Εικασία 1.3 (Deque [6]) *Το κόστος της εκτέλεσης μιας ακολουθίας m πράξεων διπλής ουράς σε ένα οποιοδήποτε δυαδικό δένδρο n κόμβων χρησιμοποιώντας splay πράξεις είναι $O(m+n)$.*

2 Ορισμός του Προβλήματος

Έστω δυαδικό δένδρο εύρεσης T με n κόμβους-στοιχεία $1, 2, \ldots, n$ αριθμημένα με βάση τη συμμετρική διάταξη και μια ακολουθία σ από m αιτήσεις κάποιων στοιχείων του T. Η ακολουθία αιτήσεων είναι άγνωστη. Σε κάθε αίτηση ενός στοιχείου i ορίζουμε ως *κόστος* αίτησης το βάθος $d(i)$ του κόμβου i στο δένδρο. Μετά από την ικανοποίηση της αίτησης μπορούμε να εκτελέσουμε επαναζυγιστικές πράξεις στο T αυθαίρετου πλήθους με σταθερό κόστος. Σκοπός είναι η δημιουργία ενός online αλγορίθμου που να ελαχιστοποιεί το συνολικό κόστος των m αιτήσεων. Για την ελαχιστοποίηση του κόστους, σε κάθε αίτηση ενός στοιχείου i θέλουμε το στοιχείο i να βρίσκεται όσο πιο κοντά γίνεται στη ρίζα. Από την σκοπιά της θεωρίας ανταγωνισμού, η απόδοση του online αλγορίθμου (*λόγος ανταγωνισμού - coordination ratio(CR)*) ορίζεται ως το πηλίκο του κόστους του online αλγορίθμου προς το κόστος του offline αλγορίθμου, ο οποίος γνωρίζει εξ αρχής την ακολουθία αιτήσεων. Πιο συγκεκριμένα λέμε ότι ένας online αλγόριθμος είναι CR-ανταγωνιστικός όταν ισχύει για κάθε ακολουθία οποιουδήποτε μήκους $cost_{online}(\sigma) \leq CR \cdot cost_{offline}(\sigma) + k$ όπου k μια σταθερά.

Κάθε ισοζυγισμένο δυαδικό δένδρο είναι $O(\log n)$-ανταγωνιστικό αφού διατηρεί μέγιστο ύψος $O(\log n)$. Ενδιαφερόμαστε για δένδρα που εκτελούν μια σειρά από πράξεις ανάμεσα στις ικανοποιήσεις των αιτήσεων προσπαθώντας να ικανοποιήσουν αποδοτικά τις μελλοντικές αιτήσεις. Τα δένδρα αυτά ονομάζονται αυτορυθμιζόμενα (self-organizing). Ένας αλγόριθμος για αυτορυθμιζόμενα δένδρα που προκύπτει ως φυσική συνέπεια του αλγορίθμου move-to-front που είναι βέλτιστος για το πρόβλημα της λίστας [1] είναι ο move-to-root όπου μετά από την ικανοποίηση μιας αίτησης του στοιχείου i, μετακινεί το i στη ρίζα. Το πρόβλημα που προκύπτει είναι «τι πράξεις θα χρησιμοποιήσει ο αλγόριθμος για να μετακινήσει ένα στοιχείο στη ρίζα». Αξίζει να σημειωθεί ότι οι πράξεις που θα χρησιμοποιήσουμε είναι υπεύθυνες για την βάθος του δένδρου μετά από μια συγκεκριμένη σειρά αιτήσεων.

Μια λύση είναι να χρησιμοποιήσουμε επαναζυγιστικές πράξεις μονής και διπλής περιστροφής (Σχήμα 1 a,c) όμοιες με των υψοζυγισμένων δένδρων. Η λύση που χρησιμοποιεί αυτές τις πράξεις δεν είναι αποδοτική υπό τη σκοπιά της

θεωρίας ανταγωνισμού αφού μπορεί να συμβεί το εξής σενάριο που αποτελεί και ένα *κάτω φράγμα* για την απόδοση του αλγορίθμου:

> Έστω $\sigma^k = 1, 2, \ldots, n$ η ακολουθία αιτήσεων που περιλαμβάνει την υποακολουθία αιτήσεων $1, 2, \ldots, n$ k φορές. Τότε μετά την ικανοποίηση των πρώτων n αιτήσεων με κόστος τουλάχιστον n, το δένδρο θα έχει εκφυλιστεί σε ένα αριστερό μονοπάτι με ρίζα το στοιχείο n και μοναδικό φύλλο το στοιχείο 1. Στη συνέχεια κάθε αίτηση θα έχει κόστος n οπότε το amortized κόστος της αίτησης θα τείνει n για μεγάλο k.

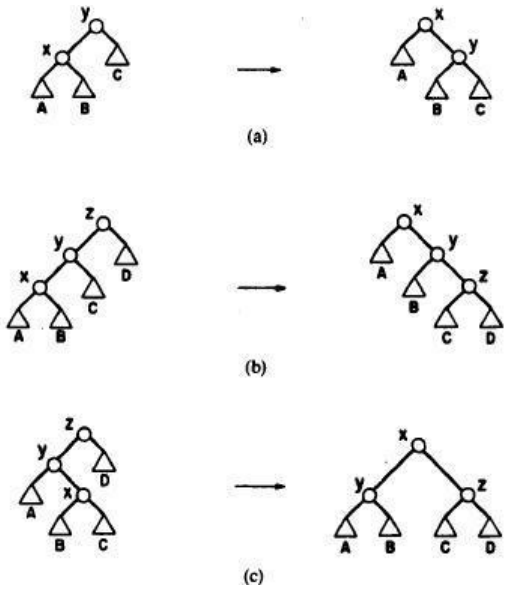
Βέλτιστος offline. Ένας βέλτιστος offline αλγόριθμος για την ακολουθία αιτήσεων $\sigma^k = 1, 2, \ldots, n$ αρχικά εκφυλίζει σε $n-1$ βήματα το δένδρο T σε ένα δεξί μονοπάτι με ρίζα το στοιχείο 1 και μοναδικό φύλλο το στοιχείο n. Στη συνέχεια, ο αλγόριθμος ικανοποιεί κάθε αίτηση και εκτελεί μια αριστερή περιστροφή. Το κόστος κάθε αίτησης θα είναι 1 αφού το αιτούμενο στοιχείο θα βρίσκεται στη ρίζα. Αυτό μπορούμε να το δούμε επαγωγικά ως εξής: στην πρώτη αίτηση το στοιχείο 1 είναι στη ρίζα και η αριστερή περιστροφή[†] μεταφέρει στη ρίζα το στοιχείο 2. Γενικά στην αίτηση του στοιχείου i το i θα βρίσκεται στη ρίζα και με μια αριστερή περιστροφή μεταφέρεται το $i+1$ στη ρίζα που είναι το επόμενο αιτούμενο στοιχείο. Σύμφωνα με την παραπάνω ανάλυση το συνολικό κόστος της ακολουθίας αιτήσεων $\sigma = 1, 2, \ldots, n$ θα είναι $2n-1$ οπότε το amortized κόστος της μιας αίτησης θα είναι $2 - \frac{1}{n}$. Το αποτέλεσμα αυτό ισχύει για οποιοδήποτε k καθώς θα εφαρμόσουμε την παραπάνω διαδικασία k φορές.

Παρατήρηση. Ο αλγόριθμος move-to-root που χρησιμοποιεί επαναζυγιστικές πράξεις απλής και διπλής περιστροφής έχει λόγο ανταγωνισμού $\Theta(n)$.

3 Splay Δένδρα

Όπως είδαμε οι επαναζυγιστικές πράξεις με τον αλγόριθμο move-to-root δεν έχουν σταθερό λόγο ανταγωνισμού. Οι Sleator και Tarjan στο [4] προτείνουν μια δομή δεδομένων που υποστηρίζει 3 ειδών επαναζυγιστικές πράξεις. Η δομή αυτή ονομάζεται *splay δένδρο*. Οι πράξεις που υποστηρίζει το splay δένδρο (splay πράξεις) είναι οι η απλή (zig) και διπλή περιστροφή (zig-zag) και επιπλέον μια πράξη που ονομάζεται zig-zig και αντιστοιχεί σε 2 απλές περιστροφές. Η διαδικασία μετακίνησης ενός κόμβου στη ρίζα με splay πράξεις ονομάζεται *splaying*. Το splaying αποτελείται από πράξεις zig-zag και zig-zig

[†]Η αριστερή περιστροφή είναι συμμετρική της δεξιάς που φαίνεται στο σχήμα 1 a)



Σχήμα 1: Οι 3 splay πράξεις: (a) Zig: απλή περιστροφή (b) zig-zig: 2 απλές περιστροφές (c) zig-zag: διπλή περιστροφή

που χρησιμοποιούνται ανάλογα με την περίπτωση όπως φαίνεται στο σχήμα 1 και στο τελευταίο βήμα εκτελεί μια πράξη zig αν το μονοπάτι έχει περιττό μήκος. Η zig-zig πράξη διαισθητικά οδηγεί σε πιο ισοζυγισμένα δένδρα αποτρέποντας τη δημιουργία μεγάλων σε μήκος μονοπατιών σε περιπτώσεις όπως αυτή στην ενότητα 2 που η ακολουθία αιτήσεων $\sigma^k = 1, 2, \ldots, n$ οδήγησε σε μη σταθερό λόγο ανταγωνισμού.

Άνω Φράγμα. Βρίσκουμε άνω φράγματα στο amortized κόστος της αίτησης ενός κόμβου χρησιμοποιώντας την τεχνική των *συναρτήσεων δυναμικού*. Συνοπτικά μια συνάρτηση δυναμικού Φ αντιστοιχίζει κάθε δυνατό στιγμιότυπο του προβλήματος με ένα *δυναμικό* που είναι ένας πραγματικός αριθμός. Αν ορίσουμε $a_j = cost_j + \Phi_j - \Phi_{j-1}$ όπου $cost_j$ το κόστος της αίτησης j και Φ_j το δυναμικό μετά από την αίτηση j. Γενικά συμβολίζουμε Φ, Φ' το δυναμικό πριν και μετά από μια πράξη splay. Αθροίζοντας τα a_j για όλες τις m αιτήσεις έχουμε $\sum a_j = \sum cost_j + \Phi_m - \Phi_0$ και αν $\Phi_0 \leq \Phi_m$ τότε $\sum cost_j \leq \sum a_j$ οπότε το $\sum a_j$ είναι ένα άνω φράγμα στο συνολικό κόστος.

Συνάρτηση Δυναμικού. Σε κάθε κόμβο i αντιστοιχίζουμε ένα θετικό βάρος w_i. Ορίζουμε s_i το άθροισμα των βαρών στο υποδένδρο με ρίζα το i, $r_i = \log s_i$ και συνάρτηση δυναμικού $\Phi = \sum_{1 \leq i \leq n} r_i$. Ισχύει το επόμενο

λήμμα.

Λήμμα 3.1 (Access Lemma[4]) *Το κόστος της αίτησης στον κόμβο x είναι το πολύ $3(r_t - r_x) + 1$ όπου t η ρίζα του δένδρου.*

Απόδειξη: Θα φράξουμε το $a_j = cost_j + \Phi_j - \Phi_{j-1}$ για κάθε splay πράξη. Χρησιμοποιούμε ιδιότητες που προκύπτουν από το σχήμα 1 καθώς και την εξής ιδιότητα των λογαρίθμων:

$$\max_{\substack{x+y\leq 1\\ x,y\geq 0}} (\log x + \log y) = -2 \tag{1}$$

Θα δείξουμε ότι ισχύει για την zig-zig και ομοίως αποδεικνύεται και για τις άλλες 2.

zig Ισχύει ότι $cost + \Phi' - \Phi \leq 1 + r'_x - r_x$

zig-zig Αφού έχουμε δύο περιστροφές το cost=2 και επειδή αλλάζει μόνο το r των κόμβων που συμμετέχουν στην πράξη ισχύει $\Phi' - \Phi = r'_x + r'_y + r'_z - r_x - r_y - r_z$. Από σχήμα 1 b φαίνεται ότι $r'_x = r_z$, $r'_x \geq r'_y$ και $r_y \geq r_x$. Από όλα τα παραπάνω ισχύει $cost + \Phi' - \Phi \leq 2 + r'_x + r'_z - 2r_x$ το οποίο θέλουμε να είναι το πολύ $3(r'_x - r_x)$ δηλαδή $2r'_x - r'_z - r_x \geq 2$. Όμως με βάση την σχέση (1) και ότι $s_x + s'_z \leq s'_x$ έχουμε $r'_z + r_x - 2r'_x = (\log s_x - \log s'_x) + (\log s'_z - \log s'_x) = \log \frac{s_x}{s'_x} + \log \frac{s_z}{s'_x} \leq -2$ οπότε $cost + \Phi' - \Phi \leq 3(r'_x - r_x)$.

zig-zag Ισχύει ότι $cost + \Phi' - \Phi \leq 2(r'_x - r_x)$

Από τα παραπάνω αποτελέσματα συμπεραίνουμε ότι ένα γενικό άνω φράγμα και για τις 3 πράξεις είναι το $3(r_t - r_x) + 1$. Παρατηρούμε επίσης ότι η πιο «ακριβή» πράξη είναι η zig-zig που είναι και η πράξη που διαχωρίζει το splay δένδρο από τα ισοζυγισμένα δένδρα. □

Ιδιότητες Splay Δένδρων. Για διαφορετικές αναθέσεις βαρών w_i στους κόμβους του δένδρου παρουσιάζονται στο [4] διαφορετικές ιδιότητες των splay δένδρων. Στη συνέχεια παρουσιάζουμε και αποδεικνύουμε αυτές τις ιδιότητες των splay δένδρων (βλ.επίσης ενότητα 1.3).

Θεώρημα 3.2 (Balance [4]) *Σε ένα splay δένδρο n κόμβων, μια ακολουθία m προσπελάσεων έχει συνολικό χρόνο προσπέλασης $O((m+n)\log n + m)$.*

Απόδειξη: Επιλέγουμε βάρη $w_i = 1/n$ για κάθε κόμβο και ορίζουμε $W = \sum_{i=1}^{n} w_i = 1$ το άθροισμα των βαρών των κόμβων. Σύμφωνα με το λήμμα 3.1 το κόστος μιας προσπέλασης φράσσεται από $3(r_t - r_x) + 1 = 3\log\frac{s_t}{s_x} + 1 = 3\log\frac{W}{\sum_{i\in T_x} w_i} + 1 \leq 3\log\frac{W}{w_i} + 1 = 3\log n + 1$ όπου t η ρίζα του δένδρου και T_x το υποδένδρο με ρίζα x. Οπότε ο συνολικός χρόνος προσπέλασης της ακολουθίας θα είναι $m(3\log n + 1) + \Phi_m - \Phi_0 \leq m(3\log n + 1) + n\log n$. Η τελευταία ανισότητα προκύπτει αφού για τη μείωση δυναμικού ισχύει $w_i \leq s_i \leq W \Rightarrow 1 \leq \frac{s_i}{w_i} \leq \frac{W}{w_i} \Rightarrow 0 \leq \log\frac{s_i}{w_i} \leq \log W w_i \Rightarrow \log s_i - \log w_i \leq \log\frac{W}{w_i} \leq n\log n$ και το θεώρημα έπεται. □

Θεώρημα 3.3 (Static Optimality[4]) *Σε ένα splay δένδρο n κόμβων, αν κάθε κόμβος i προσπελαύνεται τουλάχιστον μια φορά με συχνότητα $f(i)$ τότε μια ακολουθία m προσπελάσεων έχει συνολικό χρόνο προσπέλασης*

$$O\left(m + \sum_{i=1}^{n} f(i)\log\left(\frac{m}{f(i)}\right)\right)$$

Απόδειξη: Επιλέγουμε βάρη $w_i = f(i)/m$ για κάθε κόμβο και $W = 1$. Όμοια με την ανάλυση του θεωρήματος 3.2 έχουμε το κόστος μιας προσπέλασης να φράσσεται από $3\log(m/f(i)) + 1$ και μείωση δυναμικού είναι το πολύ $\sum_{i=1}^{n}\log\frac{m}{f(i)}$ οπότε ο συνολικός χρόνος προσπέλασης της ακολουθίας θα είναι $\sum_{i=1}^{n} f(i)(3\log\frac{m}{f(i)} + 1) + \sum_{i=1}^{n}\log\frac{m}{f(i)} = \sum_{i=1}^{n} 3f(i)\log\frac{m}{f(i)} + \sum_{i=1}^{n}\log\frac{m}{f(i)} + m$ και το θεώρημα έπεται. □

Θεώρημα 3.4 (Static Finger[4]) *Σε ένα splay δένδρο n κόμβων, μια ακολουθία m προσπελάσεων $i_1, i_2, \ldots, i_m$, όπου έχουμε αριθμήσει τα στοιχεία από 1 μέχρι n με βάση τη συμμετρική διάταξη και s ένα σταθερό στοιχείο, έχει συνολικό χρόνο προσπέλασης*

$$O(n\log n + m + \sum_{k=1}^{m}\log(|i_k - s| + 1))$$

Απόδειξη: Επιλέγουμε $w_i = \frac{1}{(|i-s|+1)^2}$ οπότε $W = \sum_{i=1}^{n}\frac{1}{((|i-s|+1)^2} \leq 2\sum_{k=1}^{\infty}\frac{1}{k^2} = O(1)$ αφού το s είναι σταθερό σημείο. Όμοια με την ανάλυση του θεωρήματος 3.2 έχουμε το κόστος της k προσπέλασης να φράσσεται από $6\log(|i_k - s| + 1) + 1$ και η μείωση δυναμικού είναι το πολύ $2\sum_{i=1}^{n}\log n = 2n\log n$ αφού $w_i \leq 1/n^2$ και το θεώρημα έπεται. □

Για τη διατύπωση του επόμενου θεωρήματος θα χρειαστούμε πρώτα τους εξής ορισμούς. Έστω $1, \ldots, m$ οι προσβάσεις με τη σειρά που προκύπτουν. Έστω ότι στην j πρόσβαση προσπελαύνουμε το στοιχείο i, ορίζουμε $t(j)$ το πλήθος των διαφορετικών στοιχείων που προσπελάστηκαν στο χρονικό διάστημα μεταξύ των 2 τελευταίων προσπελάσεων του στοιχείου i. Αν το i προσπελάστηκε για πρώτη φορά στην προσπέλαση j τότε $t(j)$ το πλήθος των διαφορετικών στοιχείων που προσπελάστηκαν στο χρονικό διάστημα μεταξύ της πρώτης προσπέλασης και της προσπέλασης j.

Θεώρημα 3.5 (Working Set[4]) *Σε ένα splay δένδρο n κόμβων, μια ακολουθία m προσπελάσεων όπου έχουμε αριθμήσει τα στοιχεία από 1 μέχρι n με βάση τη συμμετρική διάταξη έχει συνολικό χρόνο προσπέλασης*

$$O(n \log n + m + \sum_{j=1}^{m} \log(t(j)+1))$$

Απόδειξη: Αναθέτουμε τα βάρη $1, 1/4, 1/9, \ldots, 1/n^2$ στους κόμβους με βάση την σειρά των προσπελάσεων. Πιο συγκεκριμένα, ο πιο πρόσφατα προσπελάσιμος κόμβος έχει βάρος 1 και οι κόμβοι που δεν έχουν προσπελαστεί καθόλου έχουν τα μικρότερα βάρη. Έστω μια προσπέλαση j και i ο κόμβος με βάρος w_i που προσπελαύνεται τότε κάθε κόμβος k με βάρος $w_k > w_i$ θα αποκτήσει βάρος $w'_k = \frac{1}{\sqrt{\frac{1}{w_k}}+1}$ και ο i θα αποκτήσει βάρος 1. Παρατηρούμε ότι για κάθε κόμβο i αν το i είναι η ρίζα τότε το $s_i = \sum_{i=1} n1/i^2$ και παραμένει σταθερό κατά την ακολουθία των αιτήσεων. Αν το i είναι κάποιος εσωτερικός κόμβος τότε το s_i μειώνεται ή παραμένει σταθερό καθώς σε κάθε αίτηση το μόνο βάρος που αυξάνεται είναι αύτο του στοιχείου που ανεβαίνει στην ρίζα. Συμπεραίνουμε ότι οι ανακατατάξεις στα βάρη μειώνουν ή διατηρούν σταθερό το δυναμικό.

Από τα παραπάνω αλλά και με παρόμοια ανάλυση με τις αποδείξεις των προηγούμενων θεωρήματων έχουμε $W = \sum_{i=1}^{n} 1/k^2 = O(1)$. Κατά την j προσπέλαση το στοιχείου i που προσπελαύνεται έχει βάρος $w_i = \frac{1}{(t(j)+1)^2}$ αφού ανάμεσα σε 2 προσβάσεις του i η βάση του παρονομαστή αυξάνει κατά 1 για κάθε διαφορετικό στοιχείο που προσπελαύνεται. Το συνολικό κόστος των προσπελάσεων είναι $O(\sum_{j=1}^{m} \log(t(j)+1))$. Η μείωση δυναμικού είναι το πολύ $\sum_{i=1}^{n} \log \frac{W}{w_i} = \sum_{i=1}^{n} \log(t(j)+1)^2 = O(n \log n)$ αφού $t(j) \leq n$ και το θεώρημα έπεται. □

4 Ειδικές Περιπτώσεις Splay Δένδρων

Όπως αναφέρεται και στην ενότητα 1.4 ο λόγος ανταγωνισμού του splay δένδρου εικάζεται ότι είναι σταθερός αλλά δεν έχει αποδειχθεί. Στην παρούσα ενότητα αναφέρουμε συνοπτικά κάποια θεωρήματα που αποδεικνύουν κάποιες ειδικές περιπτώσεις των ανοικτών προβλημάτων στην περιοχή των splay δένδρων.

Αρχικά εξετάζεται η απόδοση του splay δένδρου για την ακολουθία αιτήσεων $\sigma = 1, \ldots, n$ την οποία ο βέλτιστος offline αλγόριθμος ικανοποιεί σε χρόνο $2n - 1$ (ενότητα 2). Το παρακάτω θεώρημα δείχνει ότι για την συγκεκριμένη ακολουθία το splay δένδρο είναι βέλτιστο.

Θεώρημα 4.1 (Scanning [4, 5]) *Σε ένα splay δένδρο n κόμβων, μια ακολουθία n προσπελάσεων που προσπελαύνει ένα διαφορετικό στοιχείο κάθε φορά*

με βάση τη συμμετρική διάταξη έχει συνολικό χρόνο προσπέλασης $O(n)$.

Σχετικά με την εικασία της διπλής ουράς αποδεικνύεται το εξής θεώρημα για την ειδική περίπτωση που οι πράξεις στην ουρά είναι μόνο POP, PUSH και INJECT.

Θεώρημα 4.2 ([6]) *Μια ακολουθία* m *οποιωνδήποτε πράξεων POP, PUSH και INJECT σε μια διπλή ουρά υλοποιημένη με ένα splay δένδρο που περιέχει αρχικά* n *κόμβους έχει συνολικό χρόνο εκτέλεσης* $O(n+m)$.

Το επόμενο θεώρημα παρουσιάζεται ως εικασία στο [4], αποδεικνύεται στα [3, 2] και είναι μια ειδική περίπτωση της dynamic optimality εικασίας.

Θεώρημα 4.3 (Dynamic Finger [4]) *Σε ένα splay δένδρο* n *κόμβων, μια ακολουθία* m *προσπελάσεων έχει συνολικό χρόνο προσπέλασης*

$$O(m+n+\sum_{j=1}^{m-1}\log(|i_(j+1)-i_j|+1))$$

όπου i_j *το στοιχείο* $i \in [1,m]$ *που προσπελαύνεται κατά την* j *αίτηση.*

Αναφορές

[1] S. Albers and J. Westbrook. *Self-organizing data structures*, volume 1442 of *Springer Lecture Notes in Computer Science*, pages 13–51. Springer, Berlin, Germany, January 1998.

[2] R. Cole. On the dynamic finger conjecture for splay trees. In *STOC*, pages 8–17, 1990.

[3] R. Cole. On the dynamic finger conjecture for splay trees. part ii: The proof. 30:44–85, 2000.

[4] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. Assoc. Comput. Mach.*, 32(3):652–686, 1985.

[5] R. Sundar. Twists, turns, cascades, deque conjecture, and scanning theorem. In *FOCS*, pages 555–559, 1989.

[6] R. E. Tarjan. Sequential access in splay trees takes linear time. *Combinatorica*, 5(4):367–378, 1985.

[7] C. C. Wang, J. Derryberry, and D. D. Sleator. O(log log n)-competitive dynamic binary search trees. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 374–383, New York, NY, USA, 2006. ACM.