Εργασία Δομών Δεδομένων

Αναστασιάδης Λουκάς,ΑΕΜ: 2333

Κατσαντούρας Ραφαήλ,ΑΕΜ:2650

Εισαγωγή,η εκφώνηση της εργασίας :

Στην εργασία αυτή μας δινόταν ένα αρχείο κειμένου,που περιέχει τις συνδέσεις μεταξύ web σελίδων. Για παράδειγμα,αν η σελίδα 1 περιέχει ένα σύνδεσμο προς τη σελίδα 2 ,τότε στο αρχείο θα υπάρχει μια γραμμή με περιεχόμενο 1 2 (μεταξύ των IDs μπορεί να υπάρχει κενός χαρακτήρας ή TAB). Υποθέστε ότι οι γραμμές υπάρχουν αποθηκευμένες με **τυχαία σειρά** μέσα στο αρχείο εισόδου. Έπρεπε να κατασκευάσουμε πρόγραμμα που θα διαβάζει το αρχείο εισόδου και θα κατασκευάζει έναν κατάλογο για τις συνδέσεις με τα ακόλουθα χαρακτηριστικά :

1) Όλα τα διαφορετικά IDs των ιστοσελίδων αποθηκεύονται σε ένα δέντρο AVL. Τα IDs είναι θετικοί ακέραιοι αριθμοί.
2) Οι γείτονες της κάθε κορυφής οργανώνονται επίσης με τη βοήθεια ενός δέντρου AVL έτσι ώστε να υποστηρίζεται **εισαγωγή** νέου συνδέσμου καθώς επίσης και διαγραφή υπάρχοντος συνδέσμου.
3) Θα πρέπει επίσης να μπορούμε να τυπώσουμε τον κατάλογο σε ένα αρχείο εξόδου. Κάθε γραμμή του αρχείου αυτού αποτελείται από το ID της σελίδας, ακολουθούμενο από το πλήθος των συνδέσμων και στη συνέχεια με τα IDs των σελίδων **σε αύξουσα διάταξη.** Η μορφή της κάθε γραμμής είναι:

   id σελίδας,πλήθος γειτόνων ,σ1 ,σ2,σ3…..,σκ

Το πρόγραμμά μας δεν θα πρέπει να διαβάζει τίποτε από το πληκτρολόγιο . Οι εντολές εισαγωγής και διαγραφής συνδέσμων καθώς και η εντολή ανάγνωσης του αρχείου εισόδου και παραγωγής του αρχείου εξόδου θα βρίσκονται μέσα στο αρχείο **commands.txt** που θα περιέχει τις εντολές: READ_DATA, WRITE_INDEX, INSERT_LINK, DELETE_LINK (κεφαλαία γράμματα). Δίνεται ένα παράδειγμα του αρχείου αυτού:

```
READ_DATA input.txt
INSERT_LINK 1 2
INSERT_LINK 5 6
DELETE_LINK 3 4
INSERT_LINK 6 7
WRITE_INDEX output.txt
```

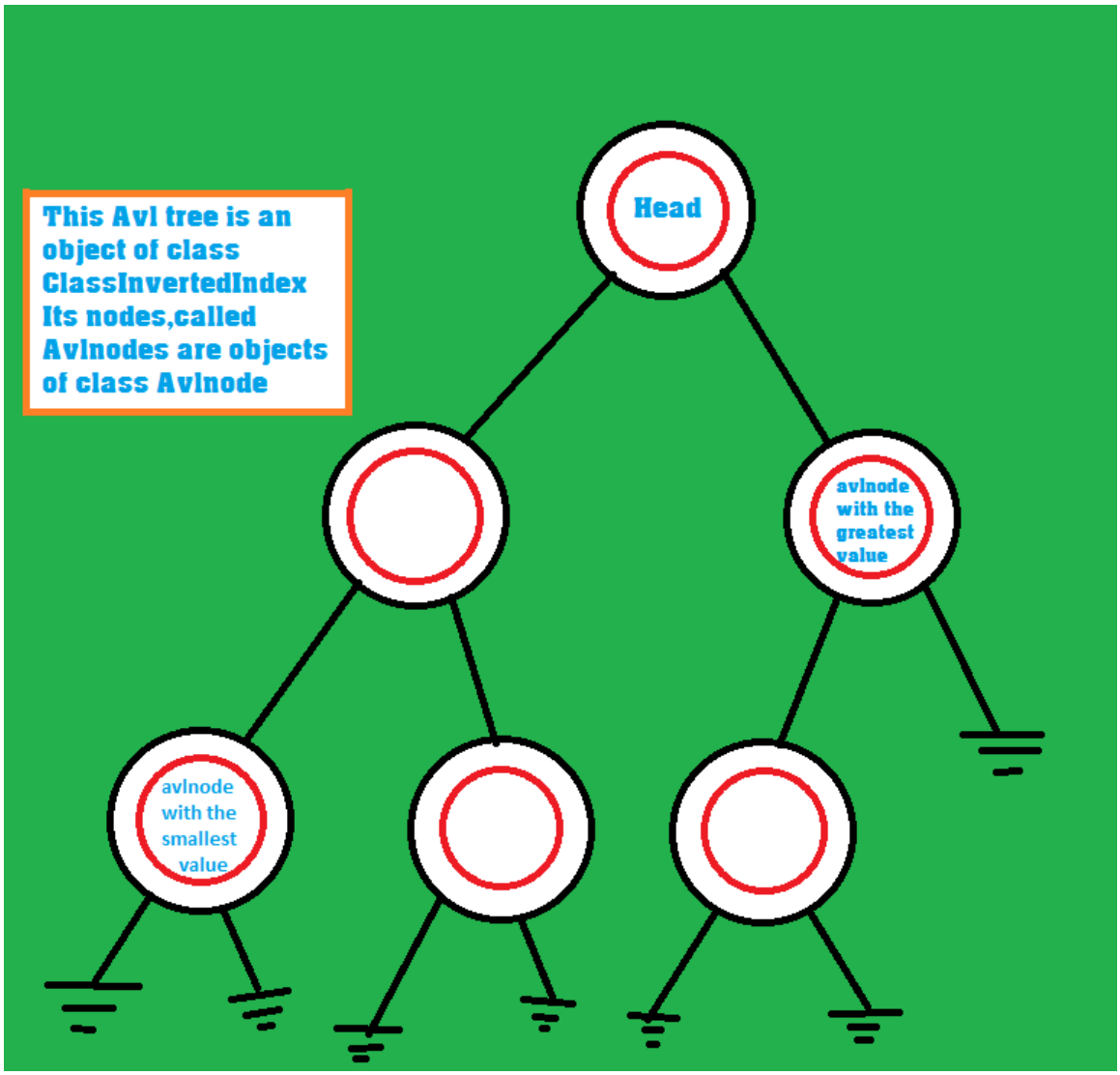
Προσέχουμε ότι η εντολή READ_DATA θα υπάρχει μία φορά στην αρχή και η εντολή WRITE_INDEX μία φορά στο τέλος. Οι άλλες εντολές μπορεί να είναι είτε INSERT_LINK x y είτε DELETE_LINK x y. Αν κατά την INSERT_LINK η ακμή υπάρχει ήδη, τότε δεν εισάγεται τίποτα. Επίσης, αν η ακμή που πάμε να διαγράψουμε με την εντολή DELETE_LINK δεν υπάρχει, τότε δε συμβαίνει τίποτε και το πρόγραμμα συνεχίζει κανονικά.

Η αρχική σκέψη, η δομή της εργασίας :

Προκειμένω να υλοποιήσουμε μία δομή ανθεκτική,ατνικειμενοστραφής ,ευκολονόητη ,γρήγορη και επεκτάσιμη σκαρφιστήκαμε την εξής ιδέα .Να τοποθετήσουμε τους συνδέσμους της κάθε σελίδας σε ένα AVL δέντρο .

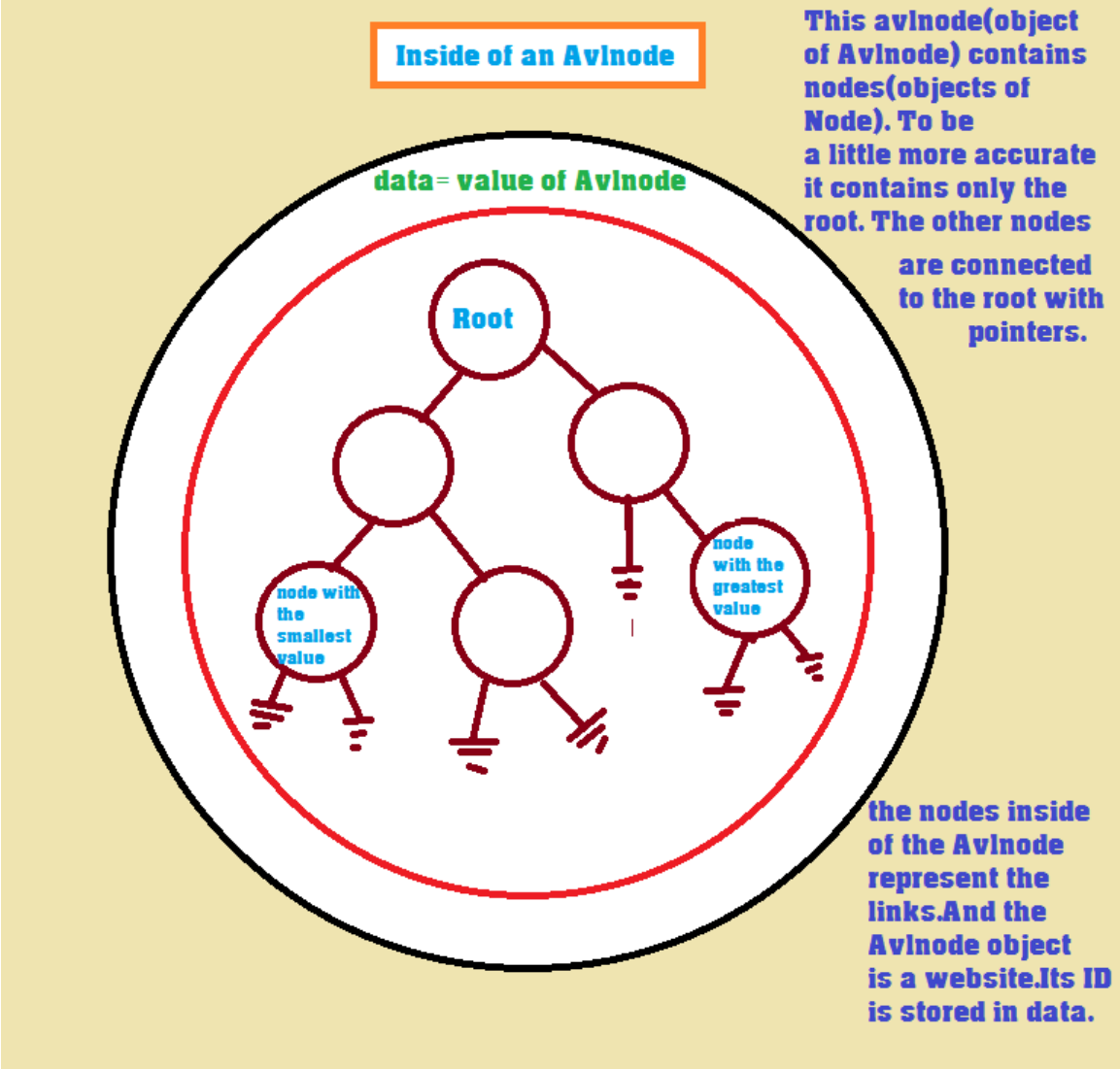
Όλες οι ιστοσελίδες οι οποίες εμπεριέχουν συνδέσμους οργανώνονται σε ένα δέντρο AVL,και κάθε ιστοσελίδα με τη δικιά της σειρά έχει ένα δικό της

AVL δέντρο το οποίο αποτελείται από τους συνδέσμους της.Τις ιστοσελίδες τις αντιμετωπίσαμε ως αντικείμενα που ονομάσαμε avlnodes (δηλαδή nodes που εμπεριέχουν AVL tree).



Τους συνδέσμους των ιστοσελίδων τους αντιμετωπίσαμε ως αντικείμενα

που ονομάσαμε nodes.



Τα nodes (οι σύνδεσμοι) δημιουργούνται στο AVL tree που βρίσκεται μέσα στα Avlnodes.Κάθε Avlnode έχει ένα δικό του μοναδικό tree από nodes. Και όλα τα Avlnodes με τη σειρά τους οργανώνονται σε ένα Avltree το οποίο είναι object της κλάσης ClassInvertedIndex.

Οι φωτογραφίες χρησιμοποίθηκαν για να γίνει η δομή μας πιο κατανοητή . Στη πραγματικότητα ως προς την υλοποίηση όμως η δομή μας δεν είναι ακριβώς έτσι. Κάθε avlnode **εμπεριέχει μόνο τη τιμή της ιστοσελίδας που αντιπροσωπεύει και τη διεύθυνση της ρίζας του δέντρου των συνδέσμων της. Δεν εμπεριέχει ολόκληρο το δέντρο,παρά μόνο αυτά τα δύο στοιχεία.** Επίσης κάθε σύνδεσμος εμπεριέχει μόνο τη την τιμή της ιστοσελίδας που αντιπροσωπεύει. Τa avlnodes και τα nodes ως αντικείμενα εμπεριέχουν τις διευθύνσεις του αριστερού και του δεξιού παιδιού τους στο δικό τους δέντρο το κάθε ένα.

Τα επιμέρους κομμάτια του κώδικα :

Η κλάση Node : Από εκεί τα αντικείμενα που κατασκευάζονται λέγονται nodes και αντιπροσωπεύουν τα λινκς των ιστοσελίδων. Έχουν μια τιμή και δείκτες προς το δεξιό και το αριστερό παιδί τους.

Συγκεκριμένα εμπεριέχει :

Ως public :

1) Ένα κενό κατασκευαστή.
2) Ένα κατασκευαστή με τιμή.
3) Ένα καταστροφέα.
4) Getters (της τιμής,των nodes (links) του δεξιού και του αριστερού παιδιού).
5) Setters(της τιμής,των κόμβων (links) του δεξιού και του αριστερού παιδιού).

Ως private :

1) Δείκτες του αριστερού και του δεξιού παιδιού.
2) Την τιμή του Node.

~~~~~~

Η κλάση Avlnode : Από εκεί τα αντικείμενα που κατασκευάζονται λέγονται avlnodes και αντιπροσωπεύουν τις ιστοσελίδες που έχουν συνδέσμους. Έχουν μια τιμή ,δείκτες προς το δεξιό και το αριστερό παιδί τους,και δείκτη προς τη ρίζα-node ,η οποία ονομάζεται root,των συνδέσμων της ιστοσελίδας.

Συγκεκριμένα εμπεριέχει :

Ως public :
~~~~~~

1) Ένα κενό κατασκευαστή.
2) Ένα κατασκευαστή με τιμή.
3) Ένα καταστροφέα.
4) Getters(του πλήθους των συνδέσμων της ιστοσελίδας,της τιμής της ιστοσελίδας δηλαδή το ID της,της διεύθυνσης της ρίζας των συνδέσμων(από εδώ και πέρα θα το λέμε root),των κόμβων (avlnodes) του δεξιού και του αριστερού παιδιού.)
5) Setters(του πλήθους των συνδέσμων της ιστοσελίδας,της τιμής της ιστοσελίδας δηλαδή το ID της,της διεύθυνσης της ρίζας των συνδέσμων(από εδώ και πέρα θα το λέμε root),των κόμβων (avlnodes) του δεξιού και του αριστερού παιδιού.)
6) Μια συνάρτηση **addlink**(unsigned int valueofnode) η οποία όταν καλείται κάνει έλεγχο με την **existsanceofelement**(Node *k,unsigned int element) αν υπάρχει σύνδεσμος που έχει την ίδια τιμή με τη valueofnode και αν όχι προχωράει στην εκχώρηση κόμβου node με τιμή valueofnode με την συνάρτηση **timetoinsertnode**(Node* root,unsigned int valueofnode) ,η οποία επιστρέφει τη ρίζα του ισοζυγισμένου AVL μετά από μια εισαγωγή κόμβου.Έτσι ο δείκτης root θα δείχνει πλέον στη καινούρια ρίζα η οποία μπορεί να είναι η ίδια ή κάποια άλλη.
7) Η **deletelink**() ,η οποία πάλι κάνει έλεγχο αν υπάρχει node με τιμή valueofnode και αν ναι προχωράει στη διαγραφή του με τη **timetodeletenode**(),η οποία και αυτή επιστρέφει τη ρίζα του ισοζυγισμένου AVL δέντρου.
8) Η inordernodestofile η οποία χρησιμοποιείται για να τυπωθούν οι τιμές των συνδέσμων με αύξουσα σειρά σε ένα αρχείο με ένα ρεύμα εξόδου.

Ως private :

1) Δείκτες του root,του αριστερού και του δεξιού παιδιού (avlnode)

2) Τις συναρτήσεις **height()**(υπολογίζει το μέγιστο ύψος ενός κόμβου node,**heightdifference()**(υπολογίζει τη διαφορά υψών του δεξιού και αριστερού υποδέντρου κόμβων node,**balance()**(χειρίζεται τις private συναρτήσεις των rotations προκειμένω να φέρει ισοροπία στο δέντρο των κόμβων node.Αυτές οι συναρτήσεις ευθύνονται για την επικράτηση ενός **ισοζυγισμένου** AVL δέντρου από nodes.
3) Μια συνάρτηση **minimumvaluenode**() η οποία επιστρέφει τον κόμβο node με την μικρότερη τιμή από το δέντρο όπου η ρίζα είναι το όρισμα της συνάρτησης **minimumvaluenode**().

~~~~~~

Η κλάση ClassInvertedIndex : Από εκεί τα αντικείμενα που κατασκευάζονται λέγονται trees και αντιπροσωπεύουν τα δέντρα των avlnodes.Στην εργασία αυτή δημιουργείται μόνο ένα τέτοιο αντικείμενο. Έχουν δείκτη προς το head το οποίο είναι η ρίζα του δέντρου των avlnodes.

Συγκεκριμένα εμπεριέχει :

Ως public :

1) Τον δείκτη head,που είναι η ρίζα του δέντρου των avlnodes.
2) Ένα κενό κατασκευαστή.
3) Ένα κενό καταστροφέα.
4) Μία συνάρτηση **addavlnode** ()(παρόμοια με την **addlink**() απλά αυτή αναφέρεται σε avlnodes και όχι σε node).
5) Μία συνάρτηση **deleteavlnode()**(παρόμοια με την **deletelink()**).
6) Getters(του πλήθους των κόμβων avlnodes,του πλήθους των κόμβων των avlnodes+ των κόμβων nodes που αυτοί περιέχουν).
7) Setters(του πλήθους των κόμβων avlnodes,του πλήθους των κόμβων των avlnodes+ των κόμβων nodes που αυτοί περιέχουν).
~~~~~~

8) Μία συνάρτηση **findavlnode**(),η οποία βρίσκει τον κόμβο avlnode στον οποίο πρέπει να εισάγει ένα κόμβο node για να δημιουργηθεί ένα λινκ(επιστρέφει τον κόμβο avlnode).
9) Μία συνάρτηση **writetofile**() η οποία συνεργάζεται με τη **write**() και με ρεύμα εξόδου παράγει το αρχείο txt που μας ζητήθηκε.

Ως private :

1) Τις συναρτήσεις **height**(),**heightdifference()**,**balance**(),οι οποίες είναι ανάλογες με αυτές που είδαμε στη κλάση avlnode,με τη διαφορά ότι αυτές αναφέρονται σε avlnode αντί για node.
2) Τις συναρτήσεις με τις περιστροφές που εκτελούνται από τη balance().
3) Μία συνάρτηση **minimumvaluenode**(),η οποία επιστρέφει τον κόμβο avlnode με τη μικρότερη τιμή.
4) Τις συναρτήσεις **timetoinsertnode**() kai **timetodeleteavlnode**() οι οποίες συνεργάζονται με τις **addavlnode**() και **deleteavlnode** () αντίστοιχα.Παρόμοιες με αυτές που είχαμε δει στη κλάση Avlnode,απλά αυτές αναφέρονται σε avlnodes αντί για nodes.
5) Μία συνάρτηση **existanceofelement**(),η οποία ελέγχει αν το ID μιας ιστοσελίδας υπάρχει ,δηλαδή αν υπάρχει avlnode με τιμή ίση με αυτή που εμπεριέχεται στο όρισμα της **existanceofelement**().
6) Την συνάρτηση write(),η οποία συνεργάζεται με την **writetofile**().
7) Δείκτη για το μέγεθος του δέντρου των avlnodes.
8) Δείκτη για το συνολικό μέγεθος(το οποίο είναι extra feature και δεν λειτουργεί.Προσθέθηκε σαν ιδέα για περεταίρω εξέλιξη του προγράμματος).Απλώς υπάρχει.Δεσμεύει ελάχιστο χώρο μιας και το δέντρο είναι ένα,και δεν καθυστερεί το πρόγραμμα σχεδόν καθόλου.

Σημείωση : Η διαφορά της κλάσης ClassInvertedIndex με την Avlnode είναι ότι η πρώτη περιέχει συναρτήσεις που επεξεργάζονται avlnodes και

το δέντρο τους,ενώ η δεύτερη περιέχει συναρτήσεις που επεξεργάζονται nodes και το δέντρο τους.

~~~~~~

Ανάλυση του κόστους αναζήτησης,εισαγωγής και διαγραφής κόμβων και επιπλέον παρατηρήσεις:

Η αναζήτηση(search) παίρνει χρόνο log(n) διότι αν θέλει να βρει ένα avlnode το μόνο που έχει να κάνει είναι να διασχίζει το δέντρο. Αν θέλει να βρει ένα λινκ πρέπει να βρει τον κόμβο που το περιέχει (log(n)) και να διασχίσει το δέντρο των λινκς(log(n)) . log(n)+log(n) μας δίνει μια πολυπλοκότητα O(log(n)).

Η εισαγωγή(insertion) είναι λίγο πιο περίπλοκη.Υπάρχουν τρεις περιπτώσεις:

1) Να υπάρχει avlnode kai node-link μέσα σε αυτό. Με άλλα λόγια το λινκ που πάει να εισαχθεί υπάρχει ήδη. Σε αυτή τη περίπτωση καλείται η **existanceofelement**() για το avlnode,η **findavlnode**() για να βρει τον κόμβο που θα εισάγει το λινκ και η **existenceofelement**() για το λινκ-node που πάει να εισάγει. Το συνολικό κόστος είναι 3 searches,2 για το δέντρο των avlnodes και μία για το δέντρο των nodes 2log(n)+1log(m).
~~~~~~

2) Να υπάρχει avlnode,αλλά να μην υπάρχει το link-node μέσα σε αυτό. Σε αυτή τη περίπτωση καλείται η **existanceofelement**() για το avlnode,η **findavlnode**() για να βρεθεί ο κόμβος που θα εισαχθεί το node-link,καλείται ξανά η **existanceofelement**() για το λινκ όμως,βγαίνει false και έπειτα καλείται η **addlink**().Το συνολικό κόστος είναι 3 searches + 1 insertion ,2 searches για το δέντρο των avlnodes,1 search για το δέντρο των nodes και 1 insertion.
Άρα συνολικό κόστος = 2log(n)+1log(m)+insertion σε m στοιχεία.

3) Να μην υπάρχει το avlnode.Άρα δεν υπάρχει ούτε το λινκ-node.Καλείται η **existanceofelement**(),η **addavlnode**(),η **findavlnode**(),η **existanceofelement**() και η **addlink**(). Το συνολικό κόστος είναι 3 searches + 2 insertions,2 searches για το δέντρο των avlnodes,1 search για το δέντρο των nodes,1 insertion για το δέντρο των avlnodes και 1 insertion για το δέντρο των nodes.
Άρα συνολικό κόστος= 2log(n)+1log(m)+insertion σε n στοιχεία +insertion σε m στοιχεία.

Οι εισαγωγές γίνονται με αναδρομικές αναθέσεις τιμών με την **timetoinsertnode**() γι'αυτό και δαπανούν αρκετό χρόνο.Εαν δεν συμπεριλαμβάναμε την **existanceofelement**() το πρόγραμμα θα λειτουργούσε μια χαρά καθώς αν πήγαινε να εισαχθεί μια τιμή που ήδη υπάρχει η **timetoinsertnode**() θα ανέθετε στον εαυτό της τον εαυτό της,και το ίδιο ισχύει για τις προηγούμενες τιμές που είχε προσπεράσει όταν διέσχισε το δέντρο κάνοντας συγκρίσεις για το που πρέπει να εισαχθεί το στοιχείο.Δηλαδή θα έχανε χρόνο με αναδρομικές αναθέσεις τιμών. Με την **existanceofelement**() αποφεύγονται οι αναδρομικές αναθέσεις στη περίπτωση που ένα input αρχείο θέλει να εισάγει πολλά λινκς για μια σελίδα.Γενικότερα η **existanceofelement**() προτιμάται για μεγάλα και ογκώδη inputs.Δεν πρόκειται για μια μικρή διαφορά,διότι χάρη σε αυτή το αποτέλεσμα είναι το πρόγραμμα να τρέχει από 1 ώρα και 10 λεπτά σε 5 μόλις λεπτά.

Αν συμπεριλαμβάναμε και τις σελίδες με 0 λινκς στο δέντρο και δεν είχαμε την **existanceofelement**() το πρόγραμμα θα τελείωνε μετά από 6 ώρες . Στο τελικό πρόγραμμα δεν συμπεριλάβαμε τις σελίδες με 0 συνδέσμους,αλλά συμπεριλάβαμε την **existanceofelement**() , και έτσι το πρόγραμμα τρέχει σε 5 λεπτά.Καταλαβαίνουμε λοιπόν το πόσο σημαντικό είναι να δημιουργηθεί ένας σωστός και ιδανικός αλγόριθμος και μία σωστή δομή προκειμένω να λύσουμε ένα πρόβλημα.

Παρόλα αυτά δίνεται η επιλογή να προσθεθούν οι σελίδες με 0 λινκ. Αυτό γίνεται με τη αφαίρεση των : '//' από τις γραμμές 41 και 54 που βρίσκονται στη **main**()

```
39                  input>>y;
40                  tree.addavlnode(x);
41                  //tree.addavlnode(y); // can be used if you want to make avlnodes from the right column of the input file
42                  Avlnode *temp;
43                  temp=tree.findavlnode(tree.head,x);
```

~~~~~~

```
53                  tree.addavlnode(x);
54                  //tree.addavlnode(y); //if reads command : INSERT_LINK 14  35 then makes 35 an avlnode
55                  Avlnode *temp;
56                  temp=tree.findavlnode(tree.head,x);
```

Τέλος κατανοούμε ότι το συγκεκριμένο πρόγραμμα μπορεί να βελτιωθεί περεταίρω αν αφαιρέσουμε κάποια περιτά searches αλλά και πάλι η διαφορά δεν θα ήταν μεγάλη μιας και το περισσότερο χρόνο τον παίρνει η ανάθεση τιμών και η δημιουργία των δέντρων.

Σας ευχαριστούμε.

"A poor programmer is he who blames his tools."
~~~~~~