ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Δομές Δεδομένων (Εργ.)
Ακ. Έτος 2018-19
Διδάσκων: Ευάγγελος Σπύρου

Εργαστήριο 7 – Δυαδικά Δένδρα Αναζήτησης

1. Στόχος του εργαστηρίου

Στόχος του έβδομου εργαστηρίου είναι η κατανόηση της δομής των δυαδικών δένδρων αναζήτησης (binary search trees), καθώς και η υλοποίηση τους σε γλώσσα C.

2. Δυαδικά Δένδρα

Η δομή δεδομένων δυαδικό δένδρο αναζήτησης αποτελεί ίσως την πιο ενδιαφέρουσα δομή δένδρου. Ένα δυαδικό δένδρο αποτελείται από ένα πεπερασμένο σύνολο κόμβων. Το δένδρο είναι είτε άδειο, είτε αποτελείται από μια ρίζα και δύο άλλα δυαδικά δένδρα που ονομάζονται αριστερό και δεξιό υποδένδρο. Ο ορισμός αυτός είναι αναδρομικός. Πιο απλά μπορούμε να ορίσουμε το δυαδικό δένδρο ως ένα δένδρο κάθε κόμβος του οποίου έχει το πολύ δύο θυγατρικούς.

Μια σπουδαία λειτουργία σε ένα δυαδικό δένδρο είναι η διάσχισή του (traverse). Πρόκειται για την επίσκεψη (visit) όλων των κόμβων του για μια μόνο φορά τον καθέναν. Υπάρχουν τρεις τρόποι για να πραγματοποιηθεί η διάσχιση:

- Προδιατεταγμένη (pre-order)
- Μεταδιατεταγμένη (post-order)
- Ενδοδιατεταγμένη (in-order)

Η **προδιατεταγμένη** (pre-order) διάσχιση απαιτεί τις ακόλουθες διαδικασίες με την εξής συγκεκριμένη σειρά:

1. Επίσκεψη της ρίζας
2. Επίσκεψη του αριστερού υποδένδρου
3. Επίσκεψη του δεξιού υποδένδρου

Τα βήματα 2. και 3. απαιτούν την επίσκεψη ενός υποδένδρου. Τα υποδένδρα αυτά διασχίζονται επαναλαμβάνοντας τις διαδικασίες 1. – 3. με την ίδια σειρά. Όταν ένα υποδένδρο είναι άδειο, τότε το αντίστοιχο βήμα παραλείπεται, όπως π.χ. οι δύο τελευταίες επισκέψεις του βήματος.

Αντίστοιχα η **μεταδιατεταγμένη** (post-order) διάσχιση ακολουθεί τα εξής βήματα:

1. Επίσκεψη του αριστερού υποδένδρου
2. Επίσκεψη του δεξιού υποδένδρου
3. Επίσκεψη της ρίζας

Τέλος, η **ενδοδιατεταγμένη** (in-order) διάσχιση ακολουθεί τα εξής βήματα:

1. Επίσκεψη του αριστερού υποδένδρου
2. Επίσκεψη της ρίζας

3. Επίσκεψη του δεξιού υποδένδρου

3. Δυαδικά Δένδρα Αναζήτησης

Ένα δυαδικό δένδρο αναζήτησης έχει τη δομή ενός δυαδικού δένδρου. Ένα δυαδικό δένδρο μπορεί να αναπαρασταθεί μέσω μιας διασυνδεδεμένης δομής δεδομένων: κάθε κόμβος είναι ένα αντικείμενο. Περιέχει:

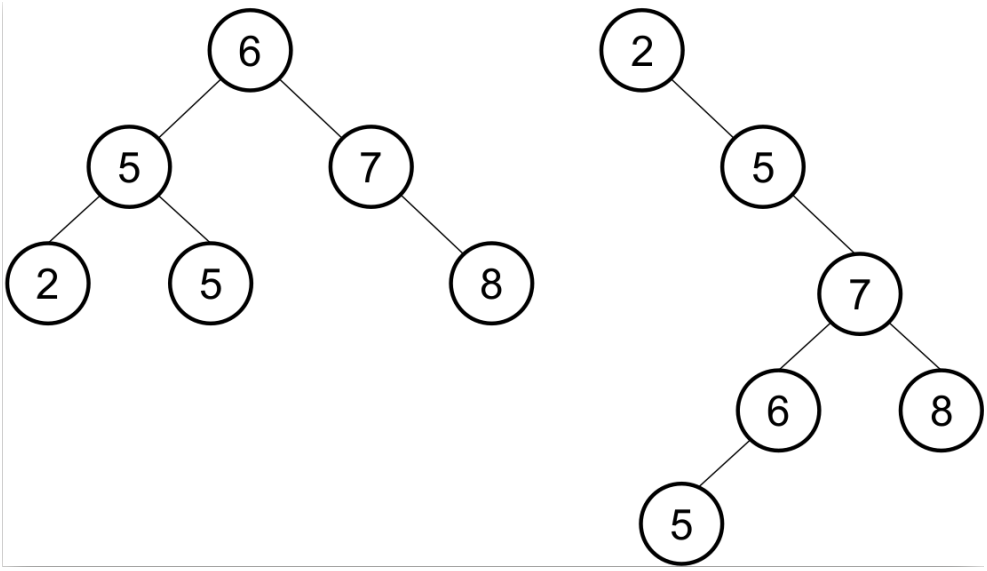
1. το πεδίο κλειδί (*key*) και ενδεχομένως παρελκόμενα δεδομένα
2. πεδία αριστερός, δεξιός, πατρικός (*left*, *right*, *parent*, αντίστοιχα).

Κάποιοι από τους κόμβους δεν υπάρχουν, τα αντίστοιχα πεδία έχουν την τιμή NULL. Η ρίζα είναι ο μόνος κόμβος με πατρικό πεδίο NULL.

Τα κλειδιά σε ένα δυαδικό δένδρο αναζήτησης είναι πάντοτε αποθηκευμένα έτσι ώστε να ικανοποιείται η ιδιότητα:

“Έστω *x* ένας κόμβος σε δυαδικό δένδρο αναζήτησης. Εάν *y* είναι οποιοσδήποτε κόμβος στο αριστερό υποδένδρο του *x*, τότε $y.key \leq x.key$. Εάν *y* είναι οποιοσδήποτε κόμβος στο δεξιό υποδένδρο του *x*, τότε $x.key \leq y.key$.”

Παραδείγματα:



Το αριστερό έχει 6 κόμβους και ύψος 2, το δεξί έχει επίσης 6 κόμβους και ύψος 4. Περιέχουν ακριβώς τα ίδια κλειδιά. Το δεξί είναι λιγότερο αποτελεσματικό, καθώς ο χρόνος εκτέλεσης για τις περισσότερες πράξεις είναι ανάλογος του ύψους του δένδρου.

4. Λειτουργίες δυαδικών δένδρων αναζήτησης

Εισαγωγή

Εάν εισάγουμε ένα κλειδί k_0 σε ένα δυαδικό δένδρο αναζήτησης και σύμφωνα με τη βασική ιδιότητά του:

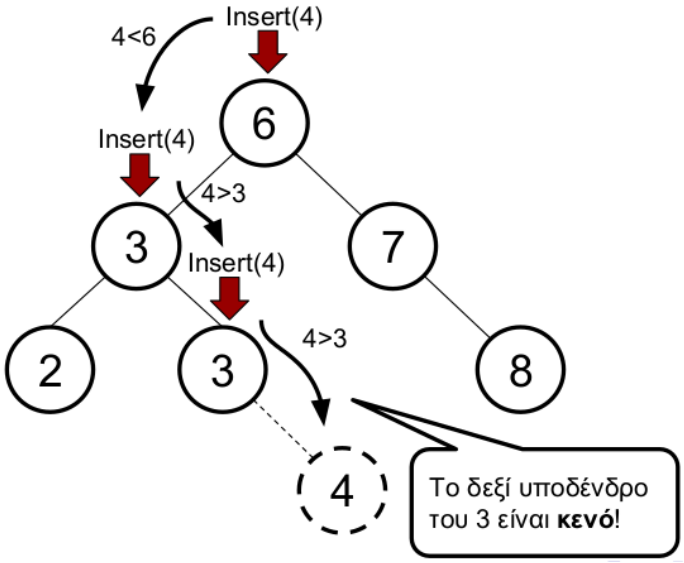
1. Εάν $k_0 < root.key$, αυτό θα εισαχθεί στο αριστερό υποδένδρο της ρίζας.
2. Εάν $k_0 > root.key$, αυτό θα εισαχθεί στο δεξί υποδένδρο της ρίζας.
3. Εάν $k_0 = root.key$, αυτό θα εισαχθεί στο αντίστοιχο υποδένδρο, σύμφωνα με την παραδοχή που έχουμε κάνει (προσοχή: να γίνεται με συνέπεια).

Ας συνεχίσουμε με το ίδιο σκεπτικό και έστω ότι $k_0 < root.key$. Τότε, το k_0 θα εισαχθεί στο αριστερό υποδένδρο και πιο συγκεκριμένα:

1. Εάν $k_0 < root.key$, αυτό θα εισαχθεί στο αριστερό υποδένδρο του αριστερού παιδιού της ρίζας
2. Εάν $k_0 > root.key$, αυτό θα εισαχθεί στο δεξί υποδένδρο του αριστερού παιδιού της ρίζας
3. Εάν $k_0 = root.key$, αυτό θα εισαχθεί στο αντίστοιχο υποδένδρο, σύμφωνα με την παραδοχή που έχουμε κάνει

Συνεπώς, για την υλοποίηση της εισαγωγής μπορούμε να χρησιμοποιήσουμε αναδρομή!

Παράδειγμα:



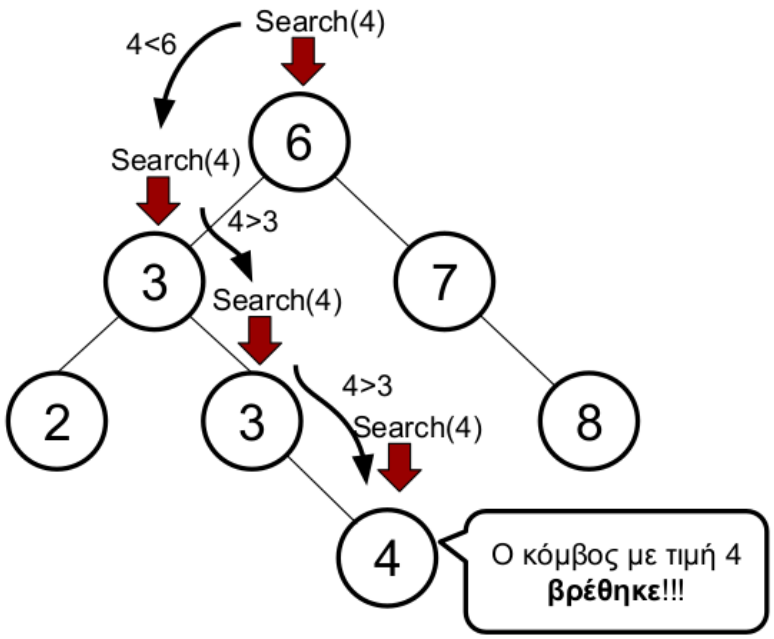
Άρα για να κάνουμε εισαγωγή σε υποδένδρο δυαδικού δένδρου αναζήτησης πρέπει να ακολουθήσουμε τα εξής βήματα:

1. Ελέγχουμε αν το υποδένδρο είναι άδειο. Αν ναι, φτιάχνουμε τον κόμβο και τον αρχικοποιούμε κατάλληλα
2. Εάν δεν είναι άδειο, και το κλειδί είναι μικρότερο της ρίζας, κάνουμε εισαγωγή στο αριστερό υποδένδρο
3. Εάν δεν είναι άδειο, και το κλειδί είναι μεγαλύτερο της ρίζας, κάνουμε εισαγωγή στο δεξί υποδένδρο

Αναζήτηση

Μια συνηθισμένη πράξη σε ένα δυαδικό δένδρο αναζήτησης είναι η αναζήτηση ενός στοιχείου που είναι αποθηκευμένο στο δένδρο. Εκτός από την αναζήτηση, τα δυαδικά δένδρα αναζήτησης υποστηρίζουν τις πράξεις εύρεσης ελαχίστου/μεγίστου, διαδόχου/προκατόχου.

Παράδειγμα:



Ελάχιστο και Μέγιστο

Για να εντοπίσουμε σε κάποιο δυαδικό δένδρο αναζήτησης το κλειδί με την ελάχιστη τιμή, μπορούμε ξεκινώντας από τη ρίζα να ακολουθήσουμε τους διαδοχικούς δείκτες *right* μέχρι να συναντήσουμε NULL. Με τον ίδιο τρόπο θα αναζητήσουμε το κλειδί με την ελάχιστη τιμή.

Διάδοχος και Προκάτοχος

Εάν όλα τα κλειδιά είναι διαφορετικά μεταξύ τους, ο διάδοχος (προκάτοχος) ενός κόμβου *x* είναι ο κόμβος που περιέχει το μικρότερο (μεγαλύτερο) από τα κλειδιά που είναι μεγαλύτερα (μικρότερα) από *x.key*.

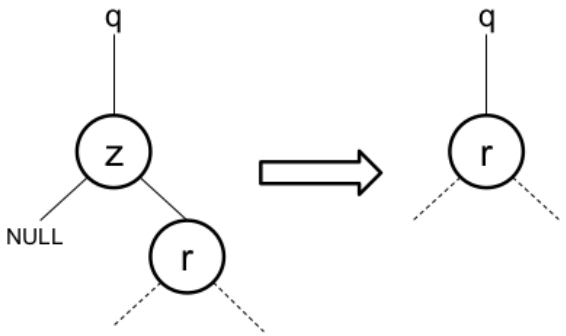
Διαγραφή

Η διαδικασία για τη διαγραφή δεδομένου κόμβου *z* δέχεται ως είσοδο έναν δείκτη προς τον *z*. Υπάρχουν τρεις βασικές περιπτώσεις.

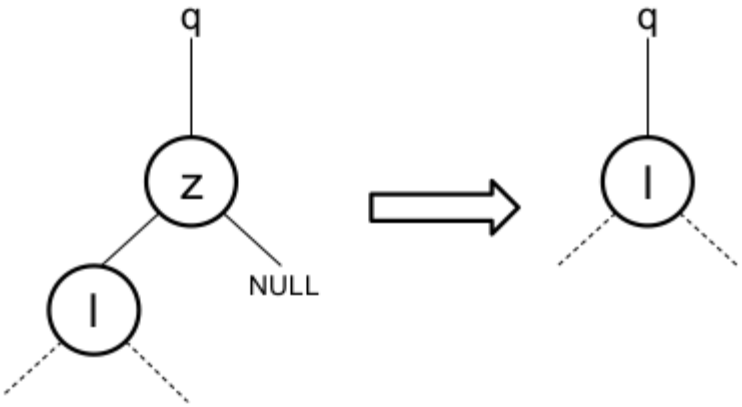
1. Ο *z* δεν έχει θυγατρικούς κόμβους: Τότε απλά τον αφαιρούμε και στον πατρικό του κόμβο τον αντικαθιστούμε με NULL
2. Ο *z* έχει μόνο έναν θυγατρικό κόμβο: Ο κόμβος αυτός παίρνει τη θέση του *z*, δημιουργώντας σύνδεση με τον πατρικό του *z*
3. Ο *z* έχει δύο θυγατρικούς κόμβους: Αρχικά εντοπίζουμε τον διάδοχό του, έστω *y*, ο οποίος πρέπει να βρίσκεται στο δεξιό υποδένδρο του *z*. Έπειτα ο *y* πρέπει να πάρει τη θέση του *z*. Το υπόλοιπο δεξί υποδένδρο του *z* γίνεται το νέο δεξιό υποδένδρο του *y*, αντίστοιχα και το αριστερό. Έχει σημασία εάν ο *y* είναι δεξιός θυγατρικός κόμβος του *z*.

1η Περίπτωση:

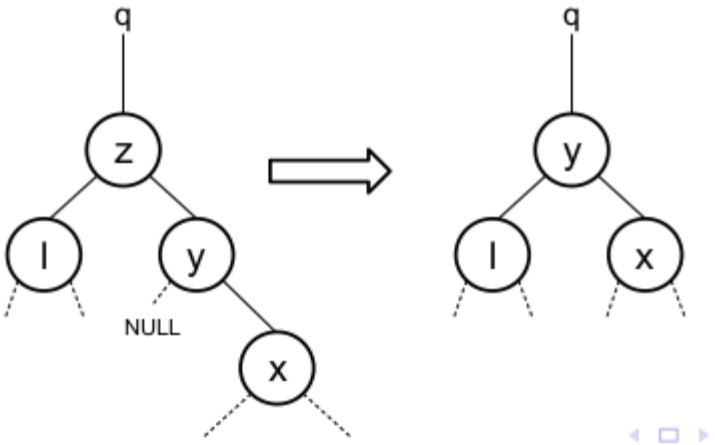
Ο *z* δεν έχει αριστερό θυγατρικό κόμβο. Αντικαθίσταται από τον δεξιό θυγατρικό του κόμβο, ο οποίος μπορεί και να είναι NULL. Όταν είναι NULL είναι ισοδύναμη περίπτωση με το να μην έχει καθόλου θυγατρικούς. Όταν δεν είναι NULL, είναι η περίπτωση να έχει μόνο δεξιό θυγατρικό.



2η Περίπτωση:
Ο *z* έχει μόνο αριστερό θυγατρικό κόμβο. Αντικαθίσταται από αυτόν.

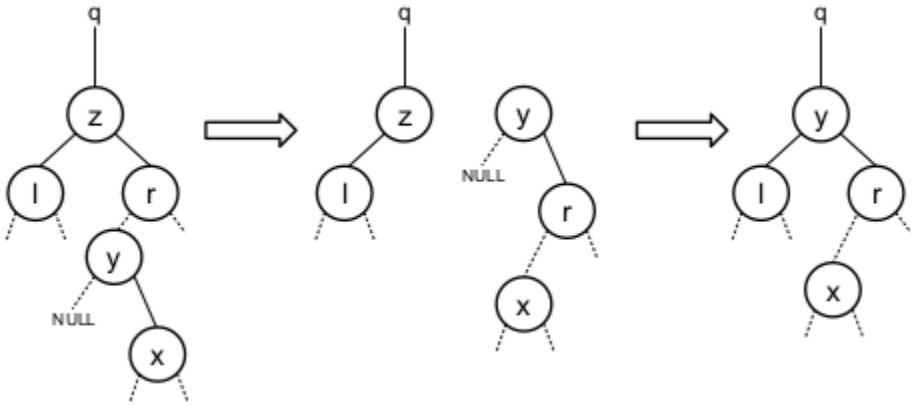


3α Περίπτωση:
Ο *z* έχει και αριστερό και δεξιό θυγατρικό κόμβο. Βρίσκουμε τον διάδοχο του *z*, έστω *y*. Αυτός βρίσκεται στο δεξί υποδένδρο του *z* και δεν έχει αριστερό θυγατρικό κόμβο. Θέλουμε να μετακινήσουμε τον *y* ώστε να αντικαταστήσει τον *z*. Ο *y* είναι ο δεξιός θυγατρικός κόμβος του *z*. Αντικαθιστούμε τον *z* με *y*, αφήνοντας μόνο τον δεξιό θυγατρικό κόμβο του *y*.



3β Περίπτωση:
Ο *z* έχει και αριστερό και δεξιό θυγατρικό κόμβο. Βρίσκουμε τον διάδοχο του *z*, έστω *y*. Αυτός βρίσκεται στο δεξί υποδένδρο του *z* και δεν έχει αριστερό θυγατρικό κόμβο. Θέλουμε να μετακινήσουμε τον *y* ώστε να αντικαταστήσει τον *z*. Αν ο *y* βρίσκεται στο δεξιό υποδένδρο του *z*,

χωρίς να είναι θυγατρικός του, πρώτα αντικαθιστούμε τον *y* από τον θυγατρικό του και έπειτα τον *z* από τον *y*.



5. Υλοποίηση δυαδικού δένδρου αναζήτησης

Να υλοποιηθεί σε γλώσσα C ένα δυαδικό δένδρο αναζήτησης που να υποστηρίζει όλες τις παραπάνω λειτουργίες.