ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δομές δεδομένων

Ενότητα 4η: Σύνολα - Λεξικά

Παναγιώτα Φατούρου
Τμήμα Επιστήμης Υπολογιστών

ΕΝΟΤΗΤΑ 4
ΣΥΝΟΛΑ - ΛΕΞΙΚΑ

Σύνολα (Sets)

- Τα μέλη ενός συνόλου προέρχονται από κάποιο χώρο U (universe) αντικειμένων/στοιχείων (π.χ., σύνολα αριθμών, λέξεων, ζευγών σύνολα αποτελούμενα από έναν αριθμό και μια λέξη, κ.ο.κ.).
- Αν S είναι ένα υποσύνολο του U και x είναι ένα αντικείμενο του U, είτε $x \in S$ ή $x \notin S$.
- Ένα σύνολο δεν περιέχει το ίδιο στοιχείο 2 ή περισσότερες φορές.
- Ένα πολυσύνολο (multi-sets) μπορεί να περιέχει πολλαπλά στιγμιότυπα του ίδιου στοιχείου.

Παραδείγματα Εφαρμογών

- Υπάρχει αυτός ο εργαζόμενος στη βάση δεδομένων των εργαζομένων;
- Υπάρχει αυτό το τηλέφωνο στον ηλεκτρονικό τηλεφωνικό κατάλογο;
- Υπάρχει αυτή η κράτηση στη βάση δεδομένων μιας αεροπορικής ή ακτοπλοϊκής εταιρείας;

Ενδεικτικές Λειτουργίες επί Συνόλων

- **MakeEmptySet()**: Επιστρέφει το κενό σύνολο $\varnothing$.
- **IsEmptySet(S)**: Επιστρέφει true αν S = $\varnothing$ και false διαφορετικά.
- **Insert(x,S)**: Προσθέτει το στοιχείο x στο σύνολο S ή δεν πραγματοποιεί καμία ενέργεια αν x $\in$ S.
- **Delete(x,S)**: Διαγράφει το στοιχείο x από το S ή δεν πραγματοποιεί καμία ενέργεια αν x $\notin$ S.
- **Member(x,S)**: Επιστρέφει true αν το x είναι μέλος του συνόλου S και false διαφορετικά.
- **Size(S)**: Επιστρέφει |S|, το πλήθος των στοιχείων του S.
- **Union(S,T)**: επιστρέφει S $\cup$ T, το σύνολο που αποτελείται από τα στοιχεία εκείνα που είναι μέλη είτε του S ή του T.

Ενδεικτικές Λειτουργίες επί Συνόλων

- **Intersection(S,T):** επιστρέφει S ∩ T, το σύνολο που αποτελείται από τα στοιχεία εκείνα που είναι μέλη και του S και του T.
- **Difference(S,T):** Επιστρέφει S \ T, το σύνολο των στοιχείων που ανήκουν στο S αλλά δεν ανήκουν στο T.
- **Equal(S,T):** Επιστρέφει true αν S = T και false διαφορετικά.
- **Iterate(S,F):** Εφαρμόζει τη λειτουργία F σε κάθε στοιχείο του S.
- Για χώρους στοιχείων στους οποίους ορίζεται η ιδιότητα της γραμμικής διάταξης:
  - **Min(S) (Max(S)):** επιστρέφει το μικρότερο (μεγαλύτερο) στοιχείο του S.

Λεξικά

Θεωρούμε ότι κάθε στοιχείο του συνόλου είναι ένα ζεύγος <K,I> όπου K είναι το κλειδί που χαρακτηρίζει μοναδικά το στοιχείο, και I είναι δεδομένα τύπου Type που έχουν συσχετισθεί με αυτό.

Ο αφηρημένος τύπος δεδομένων που υποστηρίζει τις λειτουργίες:

- MakeEmptySet(),
- IsEmptySet(),
- Insert(K,I,S) ,
- Delete(K,S),
- LookUp(K,S): Δεδομένου ενός κλειδιού K, επιστρέφει τα δεδομένα I, τέτοια ώστε <K,I> $\in$ S και σε περίπτωση που δεν υπάρχει στοιχείο με κλειδί K στο S, επιστρέφει nill,

λέγεται **λεξικό**.

Υλοποίηση Λεξικών με Λίστες

Αποθήκευση των στοιχείων του συνόλου σε μια λίστα:

- Στατική λίστα (χρήση πίνακα)
- Συνδεδεμένη Λίστα (απλά ή διπλά συνδεδεμένη).

Υλοποίηση Λειτουργιών

Συνδεδεμένη Λίστα

Χρονική Πολυπλοκότητα LookUp(): Θ(n)
Στη χειρότερη περίπτωση το στοιχείο είναι το τελευταίο στη λίστα!

Χρονική Πολυπλοκότητα Insert(): Θ(n)
Αναζήτηση και αν το στοιχείο υπάρχει δεν εισάγεται στη λίστα -> Η αναζήτηση κοστίζει O(n)

Χρονική Πολυπλοκότητα Delete(): Θ(n)
Αναζήτηση του στοιχείου και διαγραφή του -> Η αναζήτηση κοστίζει O(n)

Στατική Λίστα

Το μέγιστο πλήθος στοιχείων του συνόλου πρέπει να είναι γνωστό εξ αρχής.
Ποια είναι η χρονική πολυπλοκότητα των LookUp, Insert, Delete;

Υλοποίηση Λεξικών με Λίστες - Αναμενόμενο Κόστος

Υποθέτουμε πως η πιθανότητα p_j η LookUp() να ψάχνει για το j-οστό στοιχείο είναι η ίδια για κάθε j, $1 \leq j \leq n$. Έτσι, αν έχουμε n στοιχεία, η πιθανότητα είναι 1/n.
Έστω ότι c_j είναι το κόστος που πληρώνουμε για να βρούμε το j-οστό στοιχείο $\Rightarrow c_j = j$.

Αναμενόμενο Κόστος $= 1*1/n+2*1/n+\ldots+n*1/n$
$= (1+2+\ldots+n)/n$
$= n(n+1)/(2n)$
$= (n+1)/2$
$= \Theta(n)$.

Υλοποίηση Λεξικών με Λίστες - Αναμενόμενο Κόστος

Διαφορετικές Πιθανότητες για διαφορετικά κλειδιά

$K_1, \ldots, K_n$: τα κλειδιά στο σύνολο σε φθίνουσα διάταξη ως προς τη συχνότητα με την οποία αναζητούνται μέσω της LookUp().

$p_1 \geq p_2 \geq \ldots \geq p_n$: πιθανότητα μια LookUp() να ψάχνει για τα $K_1, K_2, \ldots, K_n$, αντίστοιχα.

Ο αναμενόμενος χρόνος αναζήτησης ελαχιστοποιείται όταν τα στοιχεία έχουν τη διάταξη $K_1, K_2, \ldots, K_n$ στη λίστα:

$$C_{opt} = \sum_{k=1}^{n} k\, p_k$$

Γιατί αυτό είναι βέλτιστο;

Αναμενόμενο Κόστος

Ας υποθέσουμε, για να καταλήξουμε σε άτοπο, ότι η διάταξη των κλειδιών που οδηγεί στο ελάχιστο αναμενόμενο πλήθος συγκρίσεων είναι $K_{m_1}, K_{m_2}, \ldots, K_{m_n}$, όπου η ακολουθία $m_1, \ldots, m_n$, αποτελεί μετάθεση της 1, …, n και ότι $p_{m_i} < p_{m_j}$ για κάποιο i < j. Το κόστος C τότε είναι:

$$C = (\sum_{\substack{k=1 \\ k \neq i \\ k \neq j}}^{n} k p_{m_k}) + i p_{m_i} + j p_{m_j}$$

Τότε, ανταλλάσσοντας τις θέσεις των K_{m_i} και K_{m_j} στην ακολουθία θα προέκυπτε μια άλλη ακολουθία, της οποίας το κόστος C′ θα ήταν:

$$C' = (\sum_{\substack{k=1 \\ k \neq i \\ k \neq j}}^{n} k p_{m_k}) + i p_{m_j} + j p_{m_i}$$

Έχουμε $C - C' = ip_{m_i} + jp_{m_j} - ip_{m_j} - jp_{m_i} = (j-i)(p_{m_j} - p_{m_i}) > 0$

Άρα, η διάταξη $K_{m_1}, K_{m_2}, \ldots, K_{m_n}$ δεν οδηγεί στο ελάχιστο αναμενόμενο πλήθος συγκρίσεων, το οποίο αντιτίθεται στην υπόθεση.

Ευριστικό «Μετακίνησης στην Αρχή» (Move-To-Front)

- Η πραγματική κατανομή πιθανότητας συνήθως δεν είναι γνωστή.
- Το λεξικό μπορεί να αλλάζει μέγεθος κατά τη χρήση του.
- Η μελέτη πιθανοτικών μοντέλων κάποιες φορές απέχει αρκετά από το τι γίνεται στην πράξη!

Ευριστικό «Μετακίνησης στην Αρχή»

«Μετά από κάθε επιτυχημένη αναζήτηση, το στοιχείο που βρέθηκε μετακινείται στην αρχή της λίστας.»

Χρονική Πολυπλοκότητα;

- Συνδεδεμένη Λίστα;
- Στατική Λίστα;

Ευριστικό «Αλληλομετάθεσης» (Transpose)

Ευριστικό «Αλληλομετάθεσης» (Transpose)

«Μετά από κάθε επιτυχημένη αναζήτηση, το στοιχείο που βρέθηκε μετακινείται μια θέση προς την αρχή της λίστας (δηλαδή ανταλλάσσεται με το προηγούμενό του στη λίστα)».

Σύγκριση μεταξύ των δύο Ευριστικών

- Το ευριστικό Transpose αποδεικνύεται ότι έχει καλύτερο αναμενόμενο κόστος από το MoveToFront.
- Ωστόσο, το Transpose σταθεροποιείται σε μια σταθερή "καλή" κατάσταση πιο αργά από το MoveToFront.

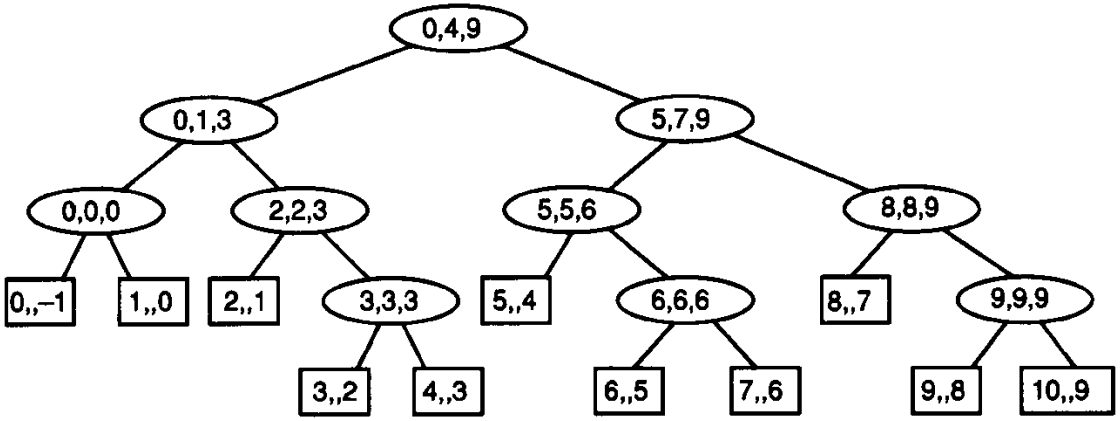
Ταξινομημένες Στατικές Λίστες

- Χρήση πίνακα για αποθήκευση των στοιχείων του συνόλου.
- Κάθε στοιχείο του πίνακα είναι ένα struct με πεδία Key και data.
- Τα στοιχεία του πίνακα είναι ταξινομημένα βάσει του κλειδιού τους.

Type BinarySearchLookUp(**key** K, **table** T[0..n-1])
/* Return information stored with key K in T, or nill if K is not in T */

```
left = 0;
right = n-1;
repeat forever
        if (right < left) then
                return nill;
        else
                middle = ⌊(left+right)/2⌋;
                if (K == T[middle]->Key) then
                        return T(middle]->data;
                else if (K < T[middle]->Key) then
                        right = middle-1;
                else left = middle+1;
```

Δυνατές Εκτελέσεις BinarySearchLookUp



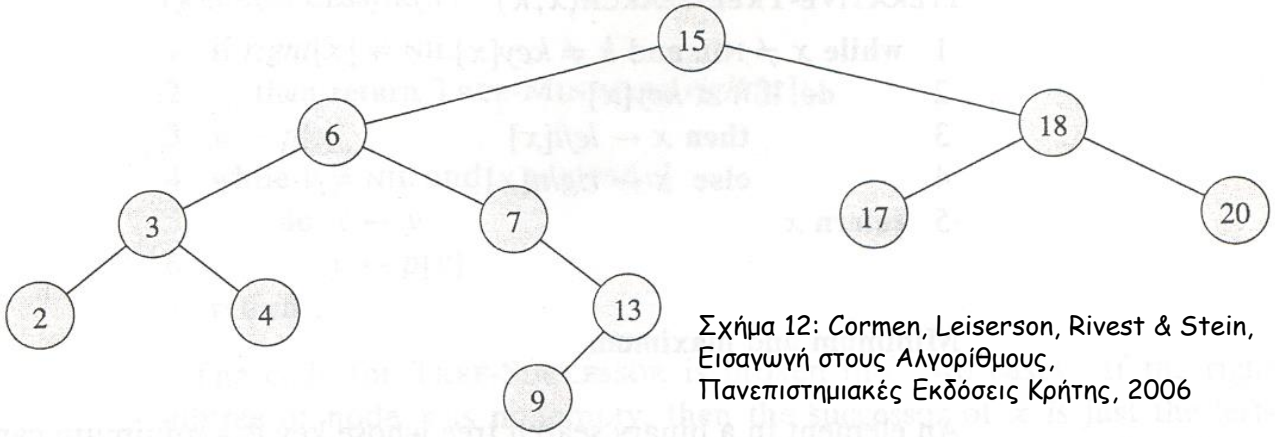
Σχήμα 6.1: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

- Το δένδρο περιγράφει όλες τις δυνατές εκτελέσεις της BinarySearch() σε ένα πίνακα 10 στοιχείων.

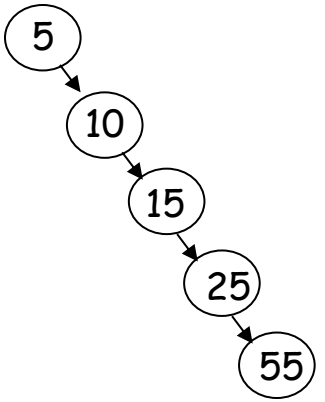
Θεώρημα 1

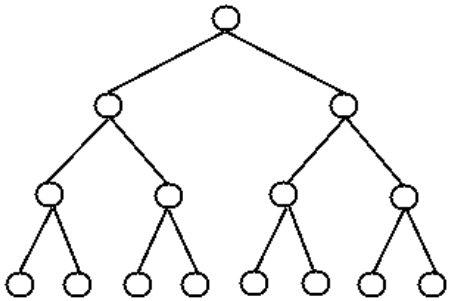
Ο αλγόριθμος δυαδικής αναζήτησης εκτελεί O(logn) συγκρίσεις για κάθε αναζήτηση στοιχείου σε πίνακα με n στοιχεία.

Ταξινομημένα Δυαδικά Δένδρα (Δένδρα Δυαδικής Αναζήτησης)



Σχήμα 12: Cormen, Leiserson, Rivest & Stein, Εισαγωγή στους Αλγορίθμους, Πανεπιστημιακές Εκδόσεις Κρήτης, 2006





Είναι δυαδικά δένδρα για τα οποία ισχύει η ακόλουθη ιδιότητα:
Για κάθε κόμβο η τιμή του κλειδιού του είναι μεγαλύτερη από όλες τις τιμές των κόμβων του αριστερού υποδένδρου του και μικρότερη από όλες τις τιμές των κόμβων του δεξιού υποδένδρου του.

Κάθε κόμβος είναι ένα struct με πεδία key, data, LC, RC.

- Το μέγιστο ύψος δυαδικού δένδρου με n κόμβους είναι n-1. Γιατί;
- Το ελάχιστο ύψος δυαδικού δένδρου με n κόμβους είναι logn. Γιατί;

Ποια η σχέση των ταξινομημένων δυαδικών δένδρων και της ενδοδιατεταγμένης διάσχισης?

Ταξινομημένα Δένδρα

Αναδρομική Έκδοση

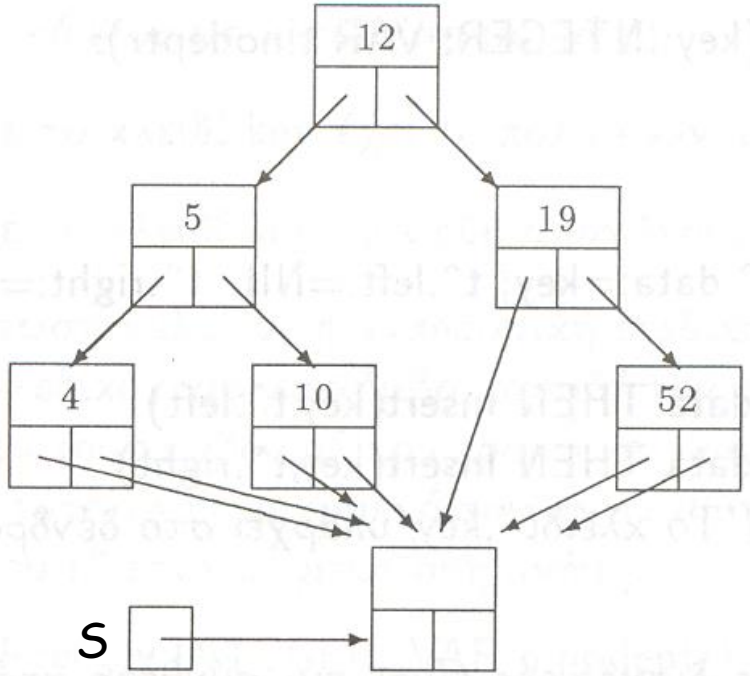
```
function BinarySearchTreeLookUp(
                    key K, pointer R):Type
/* Εύρεση του κλειδιού K στο δένδρο με ρίζα P
και επιστροφή του  πεδίου data ή nill αν το κλειδί
δεν υπάρχει στο δένδρο */
    if (R == NULL) return nill;
    else if (K == R->key) return R->data;
    else if (K < R->key)
        return(BinaryTreeLookUp(K, R->LC));
    else
        return(BinaryTreeLookUp(K, R->RC));
```

Χρονική πολυπλοκότητα;

Μη-Αναδρομική Έκδοση

```
function BinarySearchTreeLookUp (
                    key K, pointer R):Type
/* Εύρεση του κλειδιού K στο δένδρο με ρίζα P
και επιστροφή του  πεδίου data ή nill αν το
κλειδί δεν υπάρχει στο δένδρο */
    P = R;
    while (P != NULL && K != P->key) {
        if (K < P->key) P = P->LC;
        else P = P->RC;
    }
    if (P != NULL)
            return(P->data);
    else return nill;
}
```

Ταξινομημένα Δένδρα με Κόμβο Φρουρό



Σχήμα 4.16: Ι. Μανωλόπουλος, Δομές Δεδομένων, Μια προσέγγιση με την Pascal, Εκδόσεις Art of Text.

Ποια η χρησιμότητα του κόμβου φρουρού;

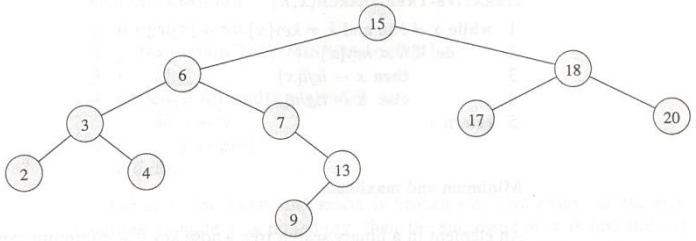
```
function BinarySearchTreeLookUp(
                        key K, pointer R):Type
    P = R;
    while (P != NULL && K != P->key) {
        if (K < P->key) P = P->LC;
        else P = P->RC;
    }
    if (P != NULL)
        return(P->data);
    else return nil;
}
```

```
function BinarySearchTreeGuardLookUp(
                        key K, pointer R):Type
    G->Key = K;
    P = R;
    while (K != P->Key) {
        if (K < P->key) P = P->LC;
        else P = P->RC;
    }
    if (P != G) return(P->data);
    else return nil;
}
```

Ταξινομημένα Δένδρα - Ελάχιστο και Μέγιστο Στοιχείο

function *BinarySearchTreeMinimum(*
pointer R): **info**
/* ο R είναι δείκτης στη ρίζα του δένδρου */

```
pointer P;

P = R;
if (P == NULL) return error;
while (P->LC != NULL) P = P->LC;
return(P->data);
```



Πολυπλοκότητα;

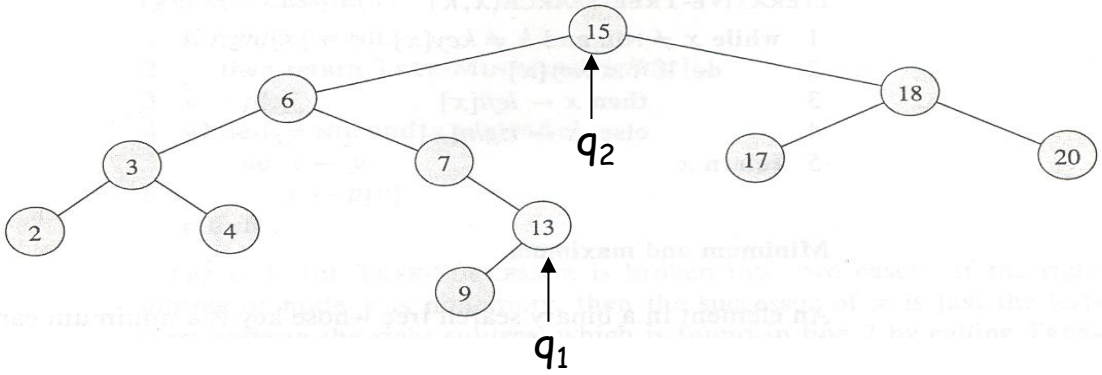
function *BinarySearchTreeMaximum(*
pointer R): **info**
/* ο R είναι δείκτης στη ρίζα του δένδρου */

```
pointer P;

P = R;
if (P == NULL) return error;
while (P->RC != NULL) P = P->RC;
return(P->data);
```

Ταξινομημένα Δένδρα – Επόμενος στην Ενδοδιατεταγμένη Διάσχιση

Στοιχείο με αμέσως μεγαλύτερο και αμέσως μικρότερο κλειδί από το κλειδί ενός κόμβου v

Το πρόβλημα είναι ίδιο με την εύρεση του επόμενου και προηγούμενου κόμβου του v στην ενδοδιατεταγμένη διάσχιση του δένδρου.

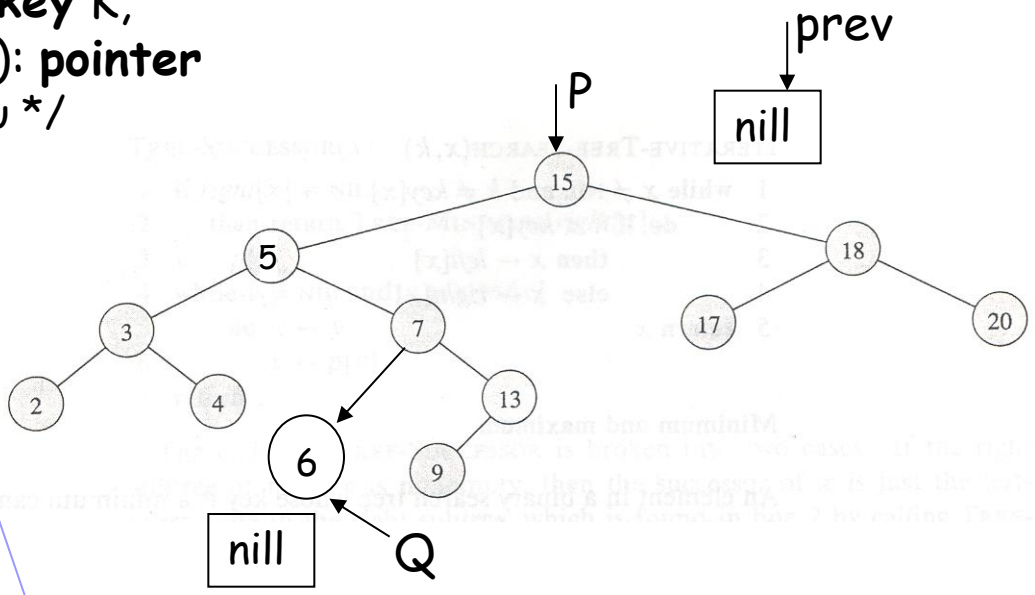


- Πως θα βρούμε τον επόμενο του κόμβου στον οποίο δείχνει ο q_1, στην ενδοδιατεταγμένη διάσχιση;
- Πως θα βρούμε τον επόμενο του κόμβου στον οποίο δείχνει ο q_2, στην ενδοδιατεταγμένη διάσχιση;

Πολυπλοκότητα;

Ταξινομημένα Δένδρα - Εισαγωγή

```
function BinarySearchTreeInsert(key K,
                                Type I, pointer R): pointer
/* ο R είναι δείκτης στη ρίζα του δένδρου */
   pointer P, Q, Prev = NULL;
   P = R;
   while (P != NULL) {
       if (P->Key == K) {
              P->data = I;
              return R;
       }
       Prev = P;
       if (K < P->Key) P = P->LC;
       else P = P->RC;
   }
   /* Δημιουργία & προσθήκη νέου κόμβου */
   Q = NewCell(Node);
   Q->Key = K; Q->data = I; Q->LC = Q->RC = NULL;
   if (Prev == NULL) return Q;
   else if (K < Prev->Key) Prev->LC = Q;
   else Prev->RC = Q;
   return R;
```



Αναζήτηση κόμβου με κλειδί ίσο με K. Αν τέτοιος κόμβος δεν βρεθεί, ο δείκτης prev, μετά το πέρας της ανακύκλωσης, θα δείχνει στον κόμβο γονέα του προς εισαγωγή κόμβου!

Παράδειγμα

Εισαγωγή κόμβου με κλειδί 6

Ταξινομημένα Δένδρα - Διαγραφή

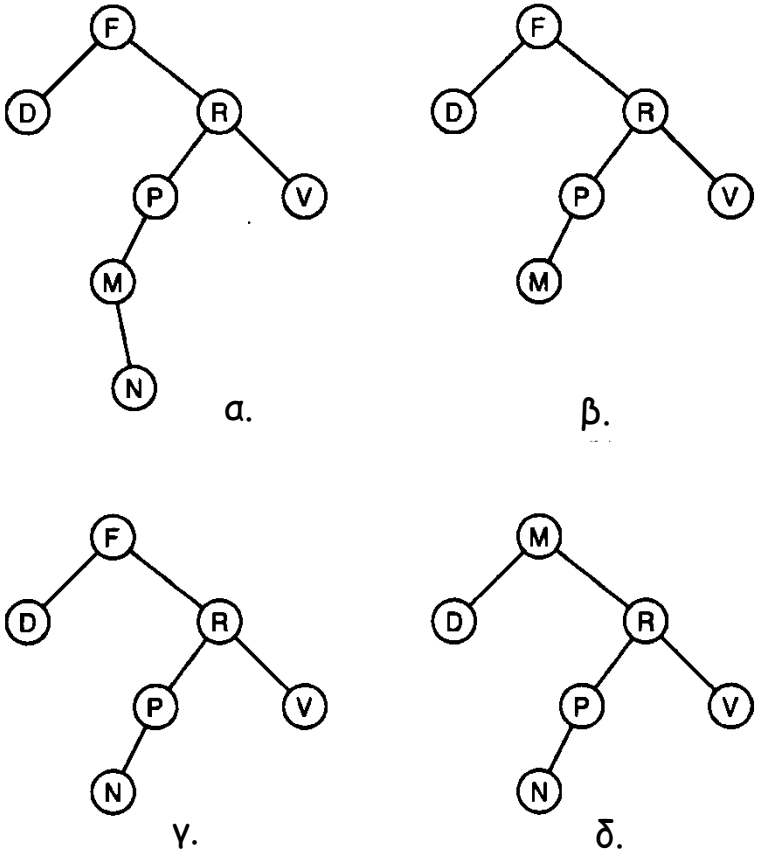
Σχήμα 6.4: Lewis & Denenberg, Data Structures & Their Algorithms, Addison-Wesley, 1991

Περιπτώσεις

1) **Ο προς διαγραφή κόμβος είναι φύλλο.** Διαγράφουμε τον κόμβο χωρίς να εκτελέσουμε κάποια επιπρόσθετη ενέργεια.

2) **Ο προς διαγραφή κόμβος είναι εσωτερικός αλλά έχει μόνο ένα παιδί.** Αντικαθιστούμε τον κόμβο με το μοναδικό παιδί του.

3) **Ο προς διαγραφή κόμβος είναι εσωτερικός με δύο παιδιά.** Αντικαθιστούμε τον κόμβο με τον επόμενο ή τον προηγούμενό του στην ενδο-διατεταγμένη διάσχιση. Αυτός είναι κόμβος με το πολύ ένα παιδί. Γιατί;

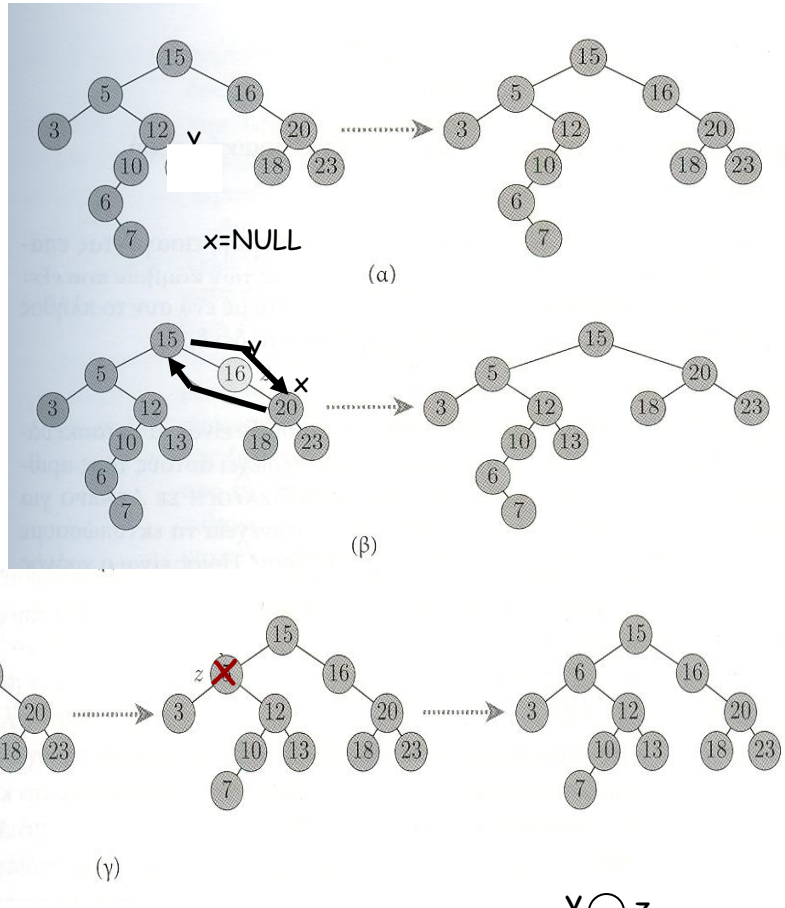


α. Αρχικό Δένδρο, β. Διαγραφή Ν από αρχικό δένδρο, γ. Διαγραφή Μ από αρχικό δένδρο, δ. Διαγραφή F από αρχικό δένδρο

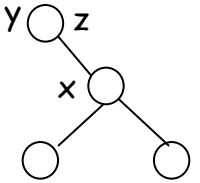
Ταξινομημένα Δένδρα - Διαγραφή

Υποθέτουμε πως το δένδρο είναι διπλά συνδεδεμένο, δηλαδή κάθε κόμβος έχει ένα δείκτη p που δείχνει στο γονικό κόμβο.

```
pointer BinaryTreeDelete(pointer R, Key K) {
/* Ο R είναι η διεύθυνση ενός δείκτη στη ρίζα του δένδρου */
/* Το K είναι το κλειδί του προς διαγραφή κόμβου */

   if (z->LC == NULL || z->RC == NULL)
       y = z;
   else y = TreeSuccessor(z);

   if (y->LC != NULL) x = y->LC;
   else x = y->RC;

   if (x != NULL) x->p = y->p;
   if (y->p == NULL) return x;
                              // διαγραφή ρίζας
   else if (y == y->p->LC) y->p->LC = x;
   else y->p->RC = x;

   if (y != z) z->Key = y->Key;
   if y has other fields copy them two;
   return R;
}
```



Σχήμα 13.4: Cormen, Leiserson & Rivest, Introduction to Algorithms, MIT Press, 1990



Ταξινομημένα Δένδρα

Θεώρημα

Η χρονική πολυπλοκότητα των λειτουργιών Insert(), Delete() και LookUp() σε ταξινομημένα δυαδικά δένδρα είναι O(h), όπου h το ύψος του δένδρου.

Ποιο πρόβλημα μπορεί να προκύψει με συνεχή αντικατάσταση του κόμβου με τον επόμενό του στην ενδο-διατεταγμένη διάσχιση?

Στατικά Ταξινομημένα Δυαδικά Δένδρα

- Υπάρχουν κλειδιά που αναζητούνται πιο συχνά και άλλα που αναζητούνται πιο σπάνια.
- Σε μια λίστα, κλειδιά που αναζητούνται συχνά κρατούνται όσο το δυνατόν πιο κοντά στην αρχή της λίστας.

Ποιο είναι το ανάλογο του ευριστικού "Μετακίνησης στην Αρχή» σε ένα ταξινομημένο δυαδικό δένδρο?

- Κάθε φορά που αναζητείται ένα κλειδί μεταφέρεται στη ρίζα.
- Ωστόσο, αυτό θα πρέπει να γίνει προσεκτικά ώστε να μην καταστρέφεται η ιδιότητα ταξινόμησης του δένδρου.

Αυτό το επιτυγχάνουν τα δένδρα Splay.

Αναφορές

Το υλικό της ενότητας αυτής περιέχεται στα ακόλουθα βιβλία:

- Harry Lewis and Larry Denenberg, *Data Structures and Their Algorithms*, Harper Collins Publishers, Inc., New York, 1991
  - Chapter 6: List and Tree Implementations of Sets
- Cormen, Leiserson, Rivest & Stein, *Εισαγωγή στους Αλγορίθμους*, Πανεπιστημαικές Εκδόσεις Κρήτης, 2006.
  - Κεφάλαιο 12: Δυαδικά Δένδρα Αναζήτησης
- Cormen, Leiserson & Rivest, Introduction to Algorithms, MIT Press, 1990.
  - Chapter 13: Binary Search Trees
- Ι. Μανωλόπουλος, Δομές Δεδομένων, Μια προσέγγιση με την Pascal, Εκδόσεις Art of Text.
  - Κεφάλαιο 4: Δένδρα

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

•Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα αδειοδότησης

.

Σημείωμα Αναφοράς