Δομές Δεδομένων

14η Διάλεξη

Δέντρα Δυαδικής Αναζήτησης

Ε. Μαρκάκης

Περίληψη

- Δέντρα Δυαδικής Αναζήτησης
- Υλοποιήσεις εισαγωγής και αναζήτησης
- Χαρακτηριστικά επιδόσεων ΔΔΑ
- Εισαγωγή στη ρίζα ΔΔΑ
- Υλοποιήσεις άλλων λειτουργιών:
  - Επιλογή, διαμέριση, αφαίρεση, ένωση

Δέντρα δυαδικής αναζήτησης

- Ανακεφαλαίωση από προηγούμενο μάθημα:
- Και στην ακολουθιακή και στη δυαδική αναζήτηση κάποιες λειτουργίες θα έχουν κόστος O(N)
- Χρήση πίνακα ή διατεταγμένης λίστας δεν είναι η καλύτερη λύση
- Βελτιώσεις:
  - Χρήση ρητής δομής δέντρου για την υλοποίηση του πίνακα συμβόλων
  - Αναδρομικές κλήσεις στα υποδέντρα
  - Πολυπλοκότητα εξαρτάται από το ύψος του δέντρου
  - Άρα O(logN) αν το δέντρο είναι «ισοζυγισμένο»

Δέντρα δυαδικής αναζήτησης

- Ορισμός δέντρου Δυαδικής Αναζήτησης (ΔΔΑ)
  - Δυαδικό δέντρο με κλειδιά σε κάθε εσωτερικό κόμβο
  - Κάθε εσωτερικός κόμβος έχει ακριβώς 2 παιδιά
  - Για κάθε εσωτερικό κόμβο
    - Οι κόμβοι του αριστερού υποδέντρου έχουν μικρότερα κλειδιά
    - Οι κόμβοι του δεξιού υποδέντρου έχουν μεγαλύτερα κλειδιά
- Κλάση ΔΔΑ
  - Ορίζει δομή κόμβων και ρίζα, αρχικά κενή

```
class ST {
  private class Node {
       ITEM item;
       Node l, r; //αναφορές σε αριστερό και δεξί
  υποδέντρο
       Node(ITEM x) { item = x; l=null; r=null; } }
  private Node head;
  ST(int maxN) { head = null; } …
```

Δέντρα δυαδικής αναζήτησης

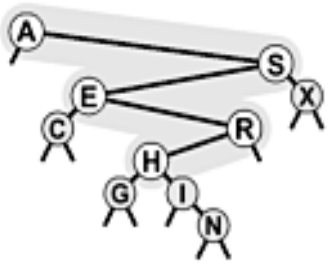
- Υλοποίηση αναζήτησης:
- Δες αν το κλειδί που ψάχνουμε είναι ίσο με το κλειδί της ρίζας
- Αν όχι αναδρομική κλήση στο αριστερό ή στο δεξί υποδέντρο.
- Εγγύηση ότι αν το κλειδί είναι μικρότερο από το κλειδί της ρίζας, αρκεί να ψάξουμε στο αριστερό υποδέντρο και αντίστροφα
- Υλοποίηση εισαγωγής:
- Προχωράμε στο δέντρο με παρόμοιο τρόπο μέχρι να βρουμε τη σωστή θέση για να διατηρηθεί η ιδιότητα του ΔΔΑ
- Εισαγωγή στη θέση κάποιου εξωτερικού κόμβου
  - Μπορούμε να χρησιμοποιήσουμε και άλλες προσεγγίσεις

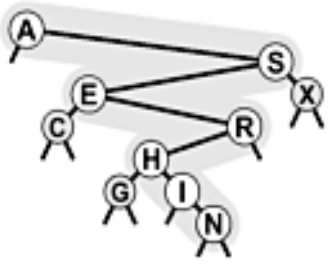
Δέντρα δυαδικής αναζήτησης

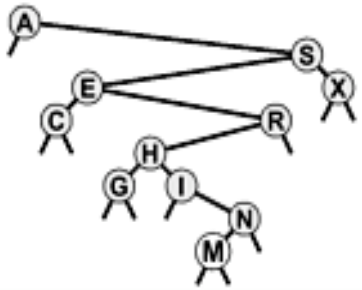
- Τάξη ΔΔΑ
  - Αναδρομική εισαγωγή και αναζήτηση

```
…private Node insertR(Node h, ITEM x) {
  if (h == null) return new Node(x);
  if (less(x.key(), h.item.key()))
       h.l = insertR(h.l, x);
  else h.r = insertR(h.r, x);
  return h; }
void insert(ITEM x) { head = insertR(head, x); }
private ITEM searchR(Node h, KEY v) {
  if (h == null) return null;
  if (equals(v, h.item.key())) return h.item;
  if (less(v, h.item.key())) return searchR(h.l, v);
       else return searchR(h.r, v); }
ITEM search(KEY key) { return searchR(head, key); }
```

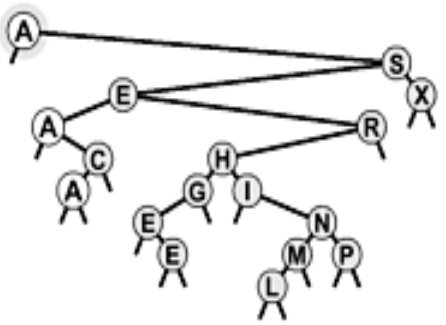
Δέντρα δυαδικής αναζήτησης

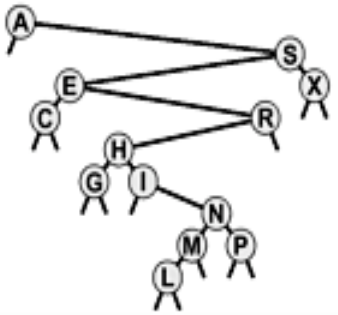






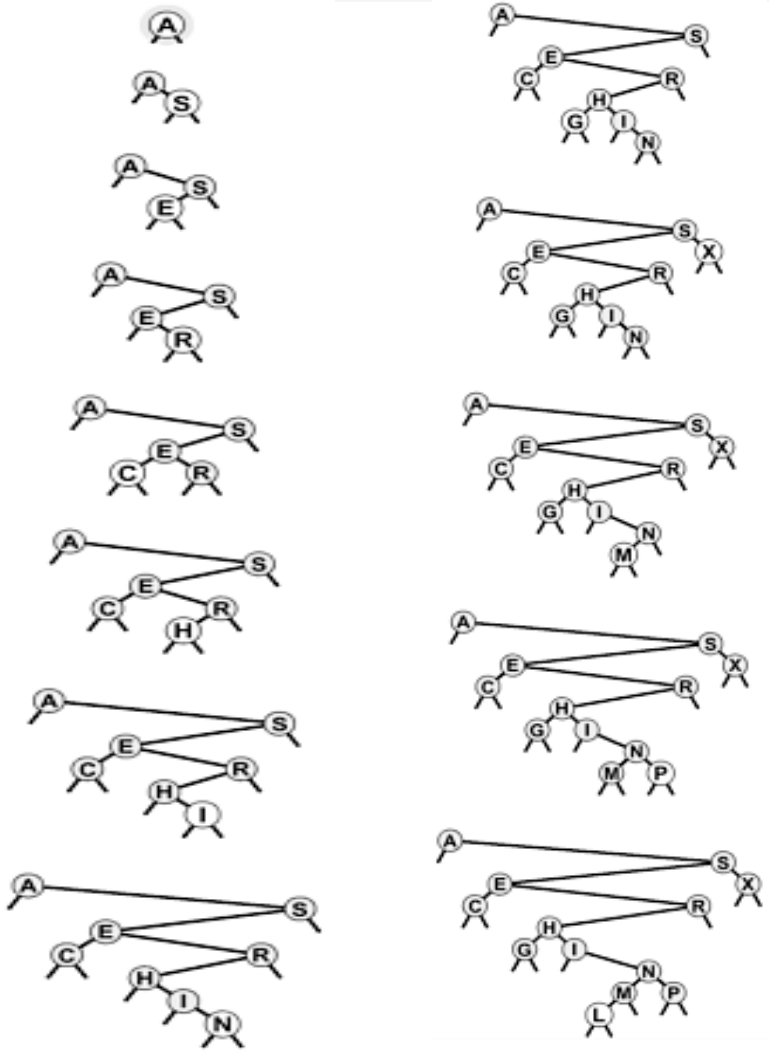
- Παραδείγματα αναζήτησης
  - Η: καταλήγουμε σε εσωτερικό κόμβο
  - Μ: καταλήγουμε σε εξωτερικό κόμβο (null)
- Παράδειγμα εισαγωγής
  - Μ: ανεπιτυχής αναζήτηση
  - Εισαγωγή νέου κλειδιού
- Χειρισμός διπλών κλειδιών
  - Δεν εισάγονται συνεχόμενα
    - Παράδειγμα: 3 εισαγωγές του Α
  - Συνεχίζουμε την αναζήτηση
  - Σταματάμε μόνο όταν βρούμε null
  - Μπορούμε να εντοπίζουμε όλα τα διπλά κλειδιά





Δέντρα δυαδικής αναζήτησης

Παράδειγμα κατασκευής Δυαδικού Δέντρου Αναζήτησης με διαδοχικές εισαγωγές



Δέντρα δυαδικής αναζήτησης

- Απλή επεξεργασία ΔΔΑ με διάσχιση
  - Παράδειγμα 1: καταμέτρηση του αριθμού των εσωτερικών κόμβων
  - Αρκεί μία διάσχιση του δέντρου, π.χ. προδιατεταγμένη
  - Ή θα μπορούσαμε να έχουμε έναν μετρητή που να αυξάνεται σε κάθε εισαγωγή

```
private int countR(Node h) {
  if (h == null) return 0;
  return 1 + countR(h.l) + countR(h.r); }
int count() { return countR(head); }
```

Δέντρα δυαδικής αναζήτησης

- Απλή επεξεργασία ΔΔΑ με διάσχιση
  - Παράδειγμα 2: ταξινομημένη εκτύπωση
  - Αρκεί να κάνουμε μία ενδοδιατεταγμένη διάσχιση

```
private String toStringR(Node h) {
  if (h == null) return "";
  String s = toStringR(h.l);
  s += h.item.toString() + "\n";
  s += toStringR(h.r);
  return s; }
public String toString() { return toStringR(head); }
```

Δέντρα δυαδικής αναζήτησης

- Μη αναδρομικοί αλγόριθμοι ΔΔΑ
  - Χρήση βρόχου αντί αναδρομής για εισαγωγή
  - Αρχικά έχουμε αναζήτηση και μετά εισαγωγή

```
public void insert(ITEM x) {
  KEY key = x.key();
  if (head == null) {
     head = new Node(x);
     return; }
  Node p = head, q = p;
  while (q != null)
     if (less(key, q.item.key())) { p = q; q = q.l; }
     else { p = q; q = q.r; }
  if (less(key, p.item.key())) p.l = new Node(x);
  else p.r = new Node(x); }
```

Χαρακτηριστικά επιδόσεων ΔΔΑ

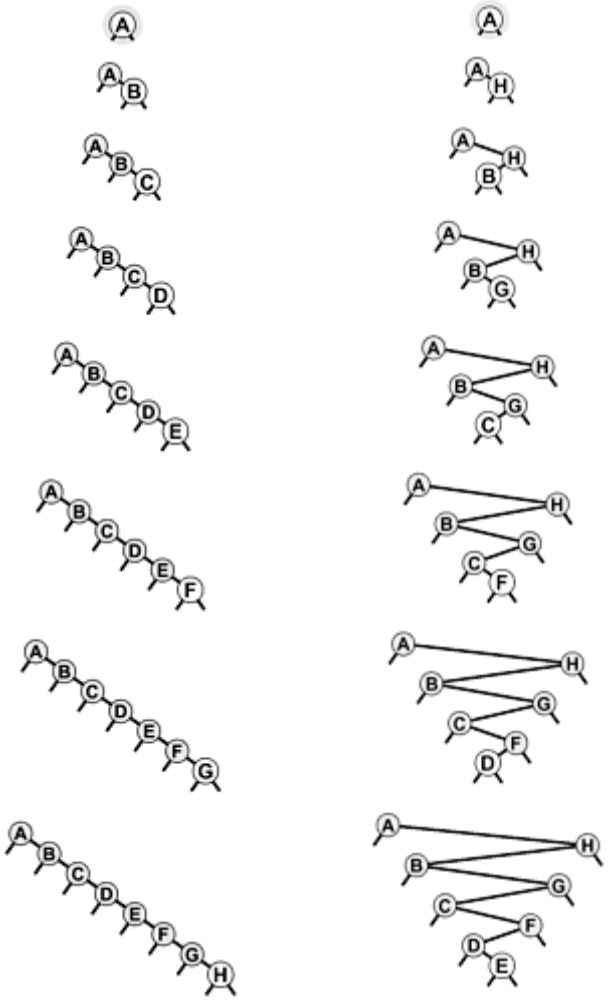
- Τα ΔΔΑ έχουν δομή παρόμοια με τη λειτουργία της Quicksort
- Η ρίζα παίζει το ρόλο του pivot της Quicksort
- Όλα τα στοιχεία στο αριστερό υποδέντρο είναι μικρότερα της ρίζας
- Όλα τα στοιχεία στο δεξιό υποδέντρο είναι μεγαλύτερα της ρίζας

- Στη Δυαδική αναζήτηση με ταξινομημένο πίνακα:
- Αναζήτηση: O(logN)
- Εισαγωγή: O(N) χειρότερη και μέση περίπτωση

- Οι επιδόσεις στα ΔΔΑ εξαρτώνται από το σχήμα του δέντρου (συγκεκριμένα από το ύψος)
  - Το ύψος είναι από logN έως N

Χαρακτηριστικά επιδόσεων ΔΔΑ

- Χειρότερη περίπτωση: N συγκρίσεις!
  - Δέντρα που εκφυλίζονται σε λίστες
  - Εμφανίζονται όταν τα κλειδιά είναι ήδη ταξινομημένα (όπως και στην Quicksort)
  - Εμφανίζονται και σε άλλες διατάξεις των κλειδιών



Χαρακτηριστικά επιδόσεων ΔΔΑ

- Όμως κατά μέσο όρο με τυχαία κλειδιά:
  1. Η επιτυχής αναζήτηση απαιτεί 1.39logN = O(logN) συγκρίσεις
  - Ανάλυση παρόμοια με αυτή της quicksort
  - Βασισμένη στον υπολογισμό της εσωτερικής διαδρομής του δέντρου
  - Υπενθύμιση: μήκος εσωτερικής διαδρομής: άθροισμά των επιπέδων ολων των εσωτερικών κόμβων
  - Ομοίως ορίζεται το μήκος εξωτερικής διαδρομής
  - Από Κεφ. 5: μήκος εξωτερικής = μήκος εσωτερικής + 2N
  2. Η ανεπιτυχής αναζήτηση και η εισαγωγή έχουν επίσης ίδιο μέσο κόστος 1.39logN = O(logN)
  - Αρκεί να υπολογίσουμε το μέσο μήκος εξωτερικής διαδρομής
  - Έχουμε N+1 ισοπίθανους εξωτερικούς κόμβους
  - Κόστος = 1/(N+1) *(μέσο μήκος εξωτερικής διαδρομής)

  - Κόστος αναζήτησης 39% υψηλότερο από τη δυαδική αναζήτηση
    - Κερδίζουμε όμως σημαντικά στο κόστος της εισαγωγής!

Χαρακτηριστικά επιδόσεων ΔΔΑ

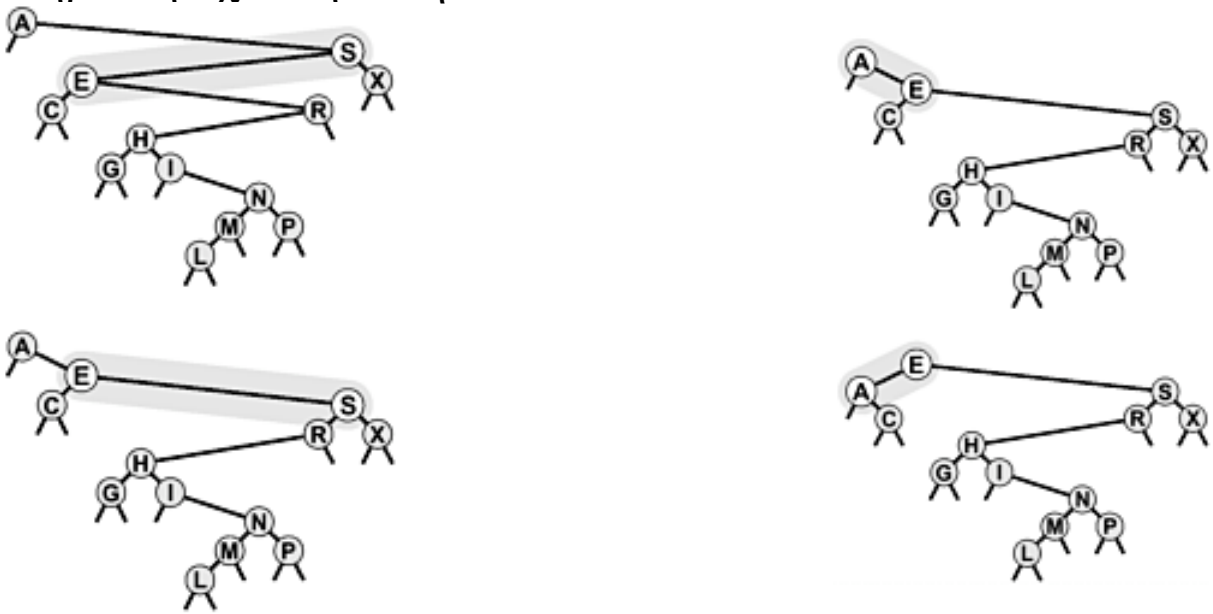
	Κατασκευή				Αναζήτηση			
N	ΑΠ	ΤΛ	ΔΑ	ΔΔΑ	ΑΠ	ΤΛ	ΔΑ	ΔΔΑ
1250	0	81	90	4	123	250	5	6
2500	0	291	356	8	457	977	9	11
5000	1	1260	1445	19	1853	4077	18	24
12500	2	10848	10684	53	12749	34723	54	69
25000				169				174
50000				407				431
100000				900				995
200000				2343				2453

Εισαγωγή στη ρίζα ΔΔΑ

- Η εισαγωγή που είδαμε γίνεται σε εξωτερικό κόμβο
- Πολλές φορές θέλουμε η εισαγωγή να γίνεται στη ρίζα
- Παραδείγματα:
  - Μία καινούρια σελίδα μπορεί να γίνει γρήγορα δημοφιλής, οι μηχανές αναζήτησης θα πρέπει να την ανακτούν σχετικά γρήγορα
  - Εμπορικές συναλλαγές στο Internet: καλύτερα να είναι πιο πάνω στο δέντρο οι πιο πρόσφατες συναλλαγές ή οι ενεργές συναλλαγές
- Υλοποίηση εισαγωγής στη ρίζα αντί στα φύλλα
  - Ο νέος κόμβος γίνεται ρίζα
  - Συγκρίνουμε τον νέο κόμβο με την παλιά ρίζα
  - Μεγαλύτερος: η παλιά ρίζα θα μπει αριστερά του
  - Μικρότερος: η παλιά ρίζα θα μπει δεξιά του
  - Ο νέος κόμβος όμως δεν είναι στο σωστό σημείο!
    - Μεγαλύτερος: ίσως υπάρχουν μικρότεροι στο δεξί υποδέντρο
    - Μικρότερος: ίσως υπάρχουν μεγαλύτεροι στο αριστερό υποδέντρο
  - Υλοποίηση με διαδοχικές περιστροφές

Εισαγωγή στη ρίζα ΔΔΑ

- Περιστροφή (Rotation) σε ΔΔΑ
  - Αντιμετάθεση της ρίζας με ένα από τα παιδιά της
  - Δεξιά περιστροφή: αριστερό παιδί γίνεται ρίζα, η ρίζα πάει στο δεξιό υποδέντρο
  - Δεξιό υποδέντρο του αριστερού παιδιού: πρέπει να γίνει αριστερό υποδέντρο της παλιάς ρίζας
  - Αριστερή περιστροφή: δεξί παιδί γίνεται ρίζα, η ρίζα πάει στο αριστερό υποδέντρο
  - Διατηρεί τη σχετική θέση των κλειδιών των παιδιών



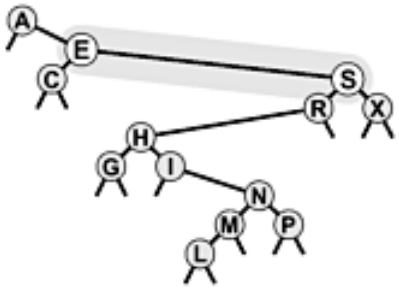
Εισαγωγή στη ρίζα ΔΔΑ

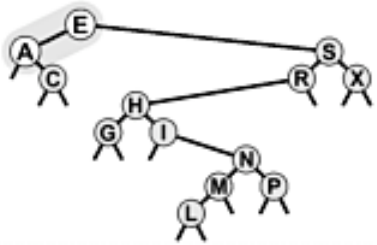
- Δεξιά περιστροφή

```
private Node rotR(Node h) {
  Node x = h.l; h.l = x.r; x.r = h; return x; }
```

- Αριστερή περιστροφή

```
private Node rotL(Node h) {
  Node x = h.r; h.r = x.l; x.l = h; return x; }
```





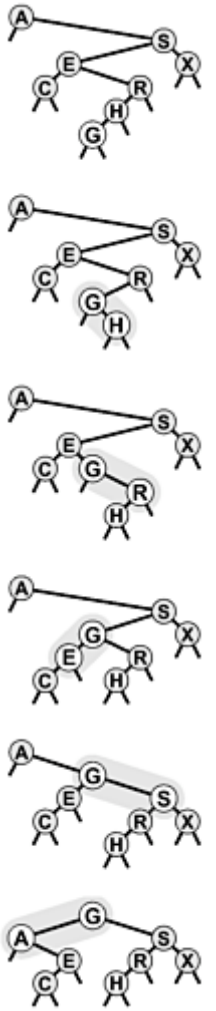
Εισαγωγή στη ρίζα ΔΔΑ

- Υλοποίηση εισαγωγής στη ρίζα
  - Εισάγουμε τον νέο κόμβο στο κατάλληλο υποδέντρο
  - Ολοκληρώνουμε με την κατάλληλη περιστροφή

```
private Node insertT(Node h, ITEM x) {
  if (h == null) return new Node(x);
  if (less(x.key(), h.item.key())) {
       h.l = insertT(h.l, x);
       h = rotR(h); }
  else {
       h.r = insertT(h.r, x);
       h = rotL(h); }
  return h; }
public void insert(ITEM x) {
  head = insertT(head, x); }
```

Εισαγωγή στη ρίζα ΔΔΑ

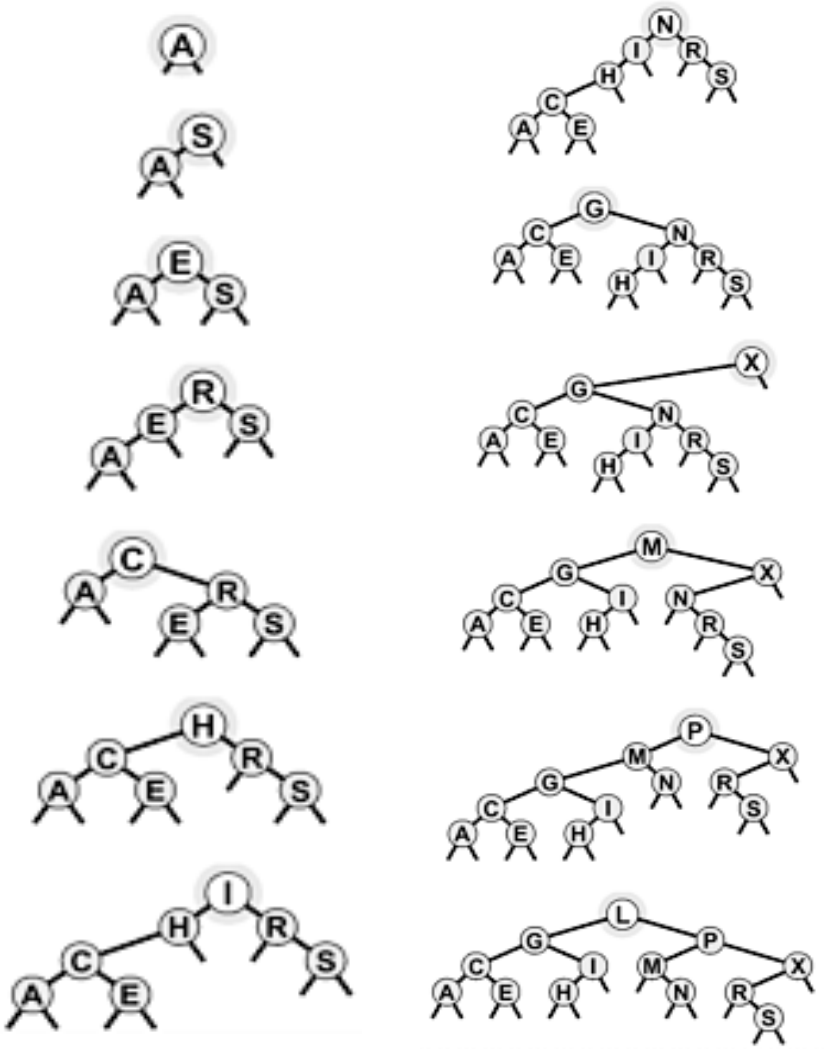
- Πώς λειτουργεί η εισαγωγή στη ρίζα;
  - Αρχικά κατεβαίνουμε μέχρι τα φύλλα
  - Εισάγουμε το κλειδί στο σωστό σημείο
  - Κάθε περιστροφή το κάνει ρίζα υποδέντρου
    - Αν μπήκε στο αριστερό υποδέντρο, δεξιά περιστροφή
    - Αν μπήκε στο δεξί υποδέντρο, αριστερή περιστροφή
  - Τελικά το κλειδί φτάνει μέχρι τη ρίζα
  - Παράδειγμα: εισαγωγή του G
- Ιδιότητες εισαγωγής στη ρίζα
  - Οι πιο πρόσφατες εισαγωγές είναι κοντά στη ρίζα
  - Μπορούμε να κάνουμε το ίδιο και στις αναζητήσεις
    - Εντοπίζουμε το κλειδί και μετά το φέρνουμε στη ρίζα
    - Οι πιο πρόσφατες αναζητήσεις είναι κοντά στη ρίζα



Εισαγωγή στη ρίζα ΔΔΑ

Παράδειγμα κατασκευής Δυαδικού Δέντρου Αναζήτησης με διαδοχικές εισαγωγές στη ρίζα:

Εισαγωγή των **A S E R C H I N G X M P L**



Υλοποιήσεις άλλων λειτουργιών

- Επιλογή k-οστού μικρότερου κόμβου
  - Μετράμε τους κόμβους ανά υποδέντρο
  - Απαιτεί ύπαρξη μετρητή σε κάθε κόμβο: μετρά πόσους κόμβους έχει το υποδέντρο με ρίζα τον κόμβο
    - Η εισαγωγή και η αφαίρεση πρέπει να ενημερώνουν τον μετρητή!
  - Έστω ότι το αριστερό υποδέντρο έχει t κλειδιά
    - Αν k<t ψάξε το k-οστό κλειδί στα αριστερά
    - Αν k>t ψάξε το (k-t-1)-οστό κλειδί στα δεξιά

```
private ITEM selectR(Node h, int k) {
//για το k-οστό το όρισμα πρέπει να είναι k-1
  if (h == null) return null;
  int t = (h.l == null) ? 0 : h.l.N; /*πεδίο N σε κάθε
  κόμβο δηλώνει το μέγεθος του υποδέντρου με ρίζα τον
  κόμβο*/
  if (t > k) return selectR(h.l, k);
  if (t < k) return selectR(h.r, k-t-1);
  return h.item; }
ITEM select(int k) { return selectR(head, k); }
```