Δομές Δεδομένων

17η Διάλεξη

Ισορροπημένα δέντρα

Ε. Μαρκάκης

Περίληψη

- Εισαγωγή
- Τυχαιοποιημένα ΔΔΑ (Randomized Binary Search trees)
- Στρεβλά ΔΔΑ (Splay trees)
- Καθοδικά δέντρα 2-3-4 (Top-Down 2-3-4 trees)

Εισαγωγή

- Μειονεκτήματα δέντρων Δυαδικής Αναζήτησης (ΔΔΑ)
  - Κακή επίδοση χειρότερης περίπτωσης
    - Εμφανίζεται σε αρκετές πρακτικές περιπτώσεις
    - Παράδειγμα: εισαγωγή ήδη ταξινομημένων κλειδιών
  - Λύση: διατήρηση των δέντρων σε ισορροπημένη μορφή
    - Όλοι οι εξωτερικοί κόμβοι στο ίδιο ή σε δύο διαδοχικά επίπεδα
  - Πρόβλημα: διατήρηση ισορροπίας μετά από αλλαγές
    - Περιοδική πλήρης ανακατασκευή

```
private Node balanceR(Node h) {
  if ((h == null) || (h.N == 1)) return h;
  h = partR(h, h.N/2);
  h.l = balanceR(h.l); h.r = balanceR(h.r);
  fixN(h.l); fixN(h.r); fixN(h);
  return h; }
```

Εισαγωγή

- Διατήρηση ΔΔΑ σε ισορροπημένη μορφή
  - Η περιοδική ανακατασκευή μπορεί να έχει μεγάλο κόστος
  - Χρειαζόμαστε μεθόδους τοπικής ανακατασκευής
- Προσεγγίσεις αποφυγής χειρότερης περίπτωσης
  - Τυχαιοποίηση: στατιστική αποφυγή της χειρότερης περίπτωσης
    - Εισάγουμε κάποια τυχαία απόφαση στον αλγόριθμο
    - Παράδειγμα: τυχαιοποιημένα ΔΔΑ
  - Απόσβεση: εκτέλεση επιπλέον εργασίας ορισμένες φορές
    - Το κόστος αποσβένεται μετά από πολλές λειτουργίες
    - Παράδειγμα: στρεβλά ΔΔΑ
  - Βελτιστοποίηση: εγγυήσεις επίδοσης σε κάθε λειτουργία
    - Χρήση πρόσθετων δομικών πληροφοριών μέσα στα ΔΔΑ
    - Παράδειγμα: δέντρα 2-3-4, δέντρα κόκκινου-μαύρου

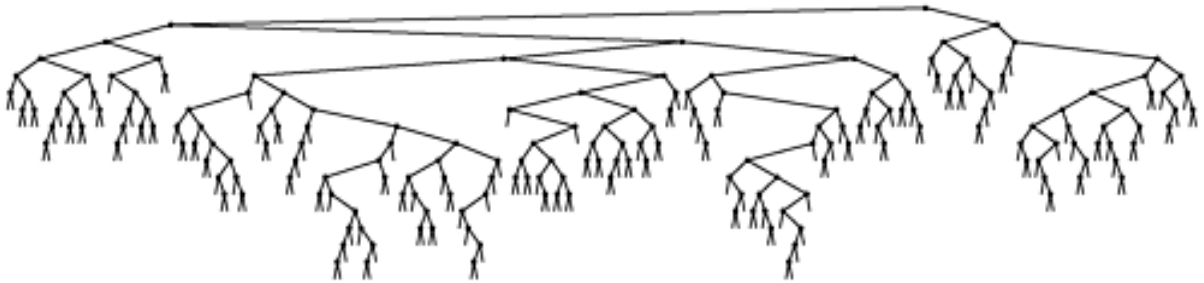
Τυχαιοποιημένα ΔΔΑ

- Απλή παραλλαγή των ΔΔΑ
  - Έστω ένα ΔΔΑ με N κόμβους
  - Το επόμενο στοιχείο εισάγεται στη ρίζα με πιθανότητα 1/(N+1)
    - Αυτή είναι η πιθανότητα να είναι η ρίζα ενός τυχαίου ΔΔΑ
  - Αναδρομική εισαγωγή σε υποδέντρο με τον ίδιο τρόπο

```
private Node insertR(Node h, ITEM x) {
  if (h == null) return new Node(x);
  if (Math.random()*h.N < 1.0) return insertT(h, x);
  if (less(x.key(), h.item.key()))
       h.l = insertR(h.l, x);
  else
       h.r = insertR(h.r, x);
  h.N++; return h; }
void insert(ITEM x) { head = insertR(head, x); }
```

Τυχαιοποιημένα ΔΔΑ

- Απόδοση τυχαιοποιημένων ΔΔΑ
  - Είναι ισοδύναμα με ΔΔΑ όπου Ν κλειδιά εισάγονται τυχαία
    - Παρά το ότι τα κλειδιά δεν εισάγονται τυχαία
  - Κατασκευή: περίπου 2NlnN συγκρίσεις
  - Αναζήτηση: περίπου 2lnN συγκρίσεις
- Ιδιότητες τυχαιοποιημένων ΔΔΑ
  - Το ΔΔΑ που προκύπτει εξαρτάται από τους τυχαίους αριθμούς
  - Αμελητέα πιθανότητα το δέντρο να είναι μη ισορροπημένο
    - Παράδειγμα: ΔΔΑ που προκύπτει από κλειδιά σε αύξουσα σειρά



Τυχαιοποιημένα ΔΔΑ

- Μειονεκτήματα τυχαιοποιημένων ΔΔΑ
  - Οι καλές γεννήτριες τυχαίων αριθμών είναι αργές
    - Χρήση γρήγορων αλλά όχι τόσο τυχαίων γεννητριών
  - Απαιτείται πεδίο πλήθους κόμβων σε κάθε κόμβο
    - Μπορεί όμως να χρησιμοποιείται και για άλλους λόγους
- Αφαίρεση: παρόμοια με τα απλά ΔΔΑ
  - Όταν ενώνουμε τα υποδέντρα παίρνουμε μια τυχαία απόφαση

```
private Node joinLR(Node a, Node b) {
  int N = a.N + b.N;
  if (a == null) return b;
  if (b == null) return a;
  if (Math.random()*N < 1.0*a.N) {
       a.r = joinLR(a.r, b); return a; }
  else { b.l = joinLR(a, b.l); return b; } }
```

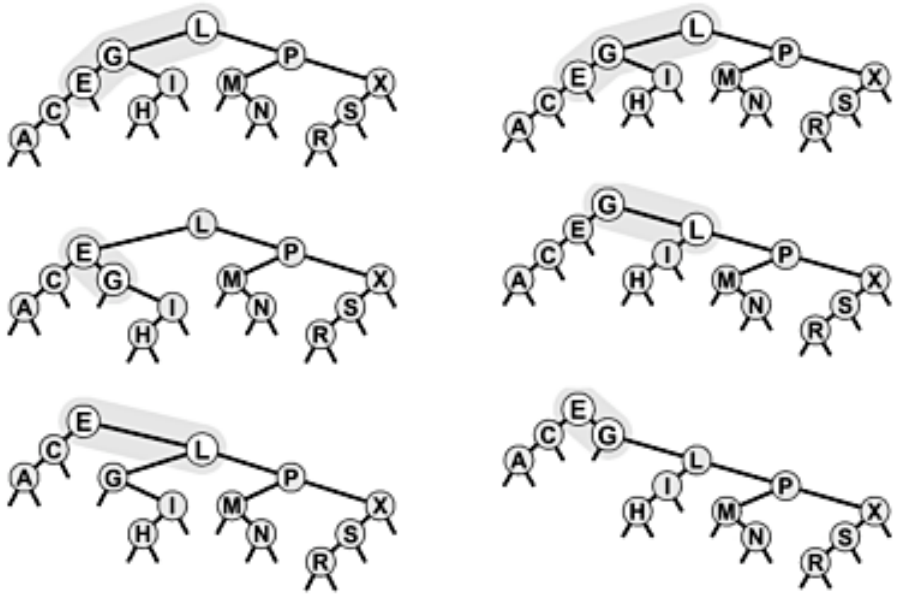
Τυχαιοποιημένα ΔΔΑ

- Υλοποίηση ένωσης: παρόμοια με τα απλά ΔΔΑ
  - Τυχαία απόφαση για το ποια ρίζα θα μείνει στην κορυφή

```
private Node joinR(Node a, Node b) {
  if (b == null) return a;
  if (a == null) return b;
  insertT(b, a.item);
  b.l = joinR(a.l, b.l);
  b.r = joinR(a.r, b.r);
  return b; }
public void join(ST b) {
  int N = head.N;
  if (Math.random()*(N+b.count()) < 1.0*N)
       head = joinR(head, b.head);
  else head = joinR(b.head, head); }
```

Στρεβλά ΔΔΑ

- Παραλλαγή των ΔΔΑ με εισαγωγή στη ρίζα
  - Λαμβάνουμε υπόψη δύο διαδοχικές περιστροφές
  - Αν είναι προς αντίθετες κατευθύνσεις, καμία αλλαγή
  - Αν είναι προς την ίδια κατεύθυνση, διαφοροποιούμαστε
    - Κάνουμε πρώτα την περιστροφή στη ρίζα και μετά στο παιδί



Στρεβλά ΔΔΑ

- Υλοποίηση στρεβλής εισαγωγής
  - Διαφοροποιείται μόνο σε ζεύγη όμοιων περιστροφών
  - Εξετάζουμε πάντα δύο επίπεδα

```
private Node splay(Node h, ITEM x) {
  if (h == null) return new Node(x);
  if (less(x.key(), h.item.key())) {
     if (h.l == null) {
        h.l = new Node(x); return rotR(h); }
     if (less(x.key(), h.l.item.key())) {
        h.l.l = splay(h.l.l, x); h = rotR(h); }
     else {
        h.l.r = splay(h.l.r, x); h.l = rotL(h.l);}
     return rotR(h); }
  else {…
```

Στρεβλά ΔΔΑ

- Υλοποίηση στρεβλής εισαγωγής

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

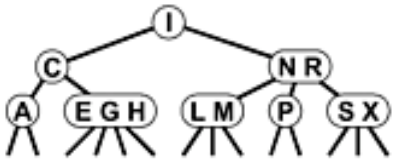
- Τι κερδίζουμε με τη στρεβλή εισαγωγή;
  - Σε κάθε εισαγωγή μικραίνει η διαδρομή εισαγωγής
    - Το μήκος της διαδρομής μειώνεται στο μισό
    - Μπορεί βέβαια να αυξάνεται το μήκος άλλων διαδρομών

Στρεβλά ΔΔΑ

- Απόδοση στρεβλών ΔΔΑ
  - Ν εισαγωγές σε άδειο ΔΔΑ: O(NlogN)
  - Ν εισαγωγές / αναζητήσεις σε ΔΔΑ Μ κόμβων: O((N+M)log(N+M))
    - Γενική περίπτωση, εξειδικεύεται στο O(NlogN) όταν M=0
  - Μέσο κόστος όλων των λειτουργιών
    - Κάποιες λειτουργίες μπορεί να είναι πολύ αργές
    - Το κόστος τους αποσβένεται σε πολλές λειτουργίες
  - Αυτή είναι η χειρότερη, όχι η μέση περίπτωση!
    - Τα κόστη είναι μέσα κόστη στη χειρότερη περίπτωση
- Βελτίωση απόδοσης στρεβλών ΔΔΑ
  - Μπορούμε να κάνουμε εξισορρόπηση και στις αναζητήσεις
  - Ιδιαίτερα αποδοτική για ανομοιόμορφες προσπελάσεις
    - Τα κλειδιά που χρησιμοποιούνται συχνά πλησιάζουν τη ρίζα

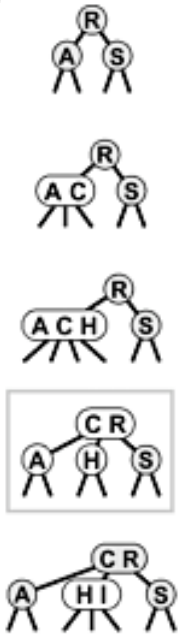
Καθοδικά δέντρα 2-3-4

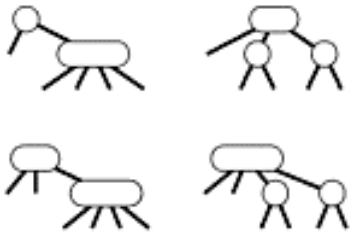
- Τα δέντρα 2-3-4 έχουν τρία είδη κόμβων
  - 2-κόμβοι: 1 κλειδί και 2 σύνδεσμοι (όπως στα ΔΔΑ)
  - 3-κόμβοι: 2 κλειδιά και 3 σύνδεσμοι
  - 4-κόμβοι: 3 κλειδιά και 4 σύνδεσμοι
  - Ισορροπημένο δέντρο 2-3-4
    - Όλοι οι κενοί σύνδεσμοι ισαπέχουν από τη ρίζα
  - Η αναζήτηση είναι γενίκευση αυτής των ΔΔΑ
  - Η εισαγωγή είναι λίγο πιο περίπλοκη
    - Δεν αρκεί αναζήτηση και μετά εισαγωγή
    - Θέλουμε να διατηρήσουμε την ισορροπία
    - Τι κάνουμε όταν θέλουμε εισαγωγή σε 4-κόμβο;



Καθοδικά δέντρα 2-3-4

- Ανοδική εισαγωγή
  - Εντοπίζουμε τον κόμβο στο τελευταίο επίπεδο
  - Αν είναι 4-κόμβος τον σπάμε στα δύο
  - Το μεσαίο κλειδί πηγαίνει στον πατέρα
  - Το νέο κλειδί εισάγεται στο κατάλληλο παιδί
  - Συνεχίζουμε αναδρομικά στον πατέρα
- Καθοδική εισαγωγή
  - Όποτε βρίσκουμε έναν 4-κόμβο τον σπάμε
    - Αυτό γίνεται σε όλα τα επίπεδα κατεβαίνοντας
  - Εισάγουμε το μεσαίο κλειδί στον πατέρα
    - Αν η ρίζα γίνει 4-κόμβος τη διασπάμε άμεσα
    - Δεν περιμένουμε την επόμενη εισαγωγή
  - Εισάγουμε το νέο κλειδί στο τελευταίο επίπεδο





Καθοδικά δέντρα 2-3-4

- Απόδοση καθοδικών δέντρων 2-3-4
  - Αναζήτηση: το πολύ logN+1 διασχίσεις κόμβων
    - Όλοι οι κόμβοι ισαπέχουν από τη ρίζα
    - Μόνο η διάσπαση της ρίζας αυξάνει το ύψος
    - Όλοι οι κόμβοι είναι τουλάχιστον 2-κόμβοι
  - Εισαγωγή: το πολύ logN+1 διαιρέσεις κόμβων
    - Διαιρέσεις σε όλη τη διαδρομή
  - Τα ανοδικά δέντρα θέλουν περισσότερες διαιρέσεις
    - Είναι όμως πιο απλά στην υλοποίηση
- Υλοποίηση δέντρων 2-3-4
  - Η χρήση 3 ειδών κόμβων έχει αρκετό κόστος
  - Μπορεί η επιβάρυνση να υπερβαίνει την ωφέλεια
  - Εναλλακτική υλοποίηση: δέντρα κόκκινου-μαύρου