Δομές Δεδομένων

11η Διάλεξη

Ταξινόμηση Quicksort

και

Ιδιότητες Δέντρων

Ε. Μαρκάκης

Περίληψη

- Quicksort
- Χαρακτηριστικά επιδόσεων
- Μη αναδρομική υλοποίηση
- Δέντρα
- Μαθηματικές ιδιότητες

Mergesort - Ανακεφαλαίωση

- Τρέχει σε χρόνο O(NlogN)
- Average case επίσης O(NlogN)
- Πολύ καλύτερη από τις στοιχειώδεις μεθόδους

- Αλλά:
- Θέλει πρόσθετο χώρο O(N) (βοηθητικό πίνακα)
- Χρόνος εκτέλεσης ανεξάρτητος από τα δεδομένα εισόδου

Ιδέες για βελτιώσεις

- Γιατί να χωρίζουμε τον πίνακα ακριβώς στη μέση σε κάθε βήμα?
- Μπορούμε να κάνουμε διαμερίσεις σε υποπίνακες που να εξαρτώνται και από τα δεδομένα
- **Quicksort:** προτάθηκε από τον Hoare (1960). Υλοποιεί τέτοιες διαμερίσεις με βάση ένα από τα στοιχεία του πίνακα
- Αποτελεί την καλύτερη μέθοδο ταξινόμησης ως σήμερα (καθιερωμένη ταξινόμηση της βιβλιοθήκης της Java)

Quicksort

- Ιδέα: ξανά διαίρει και βασίλευε
- Επιλέγουμε κάποιο στοιχείο του πίνακα ως “pivot”
- Αναδιατάσσουμε τον πίνακα σε 2 υποπίνακες έτσι ώστε
  - Πρώτος υποπίνακας έχει τα στοιχεία που είναι ≤ pivot
  - Δεύτερος υποπίνακας έχει τα στοιχεία που είναι ≥ pivot
  - Το pivot μπαίνει στην τελική του θέση
- Αναδρομή στους υποπίνακες
- Δεν χρειάζεται συγχώνευση στο τέλος!

Quicksort

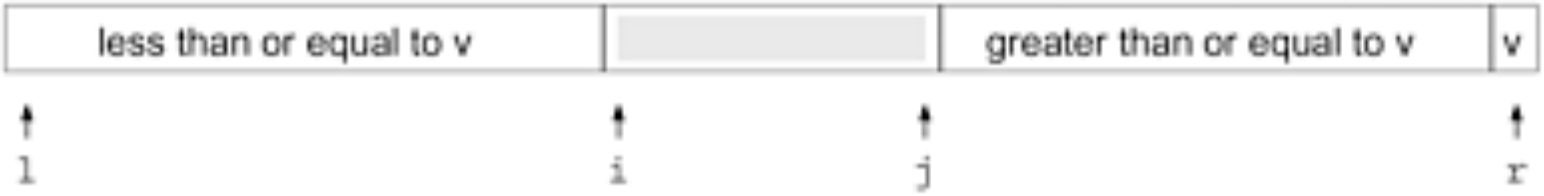
```
static void quicksort(ITEM[] a, int p, int
  r)
{ if (r <= p) return;
  int i = partition(a, p, r);/* διαμερίζει τον
  πίνακα και βάζει στη θέση a[i]  το pivot
  element */
  quicksort(a, p, i-1);
  quicksort(a, i+1, r);
}
```

Υλοποίηση της διαμέρισης

- Η partition πρέπει να τοποθετεί το pivot στοιχείο στη θέση a[i] και να φροντίζει ότι
- Τα στοιχεία a[p],…,a[i-1] είναι ≤ a[i]
- Τα στοιχεία a[i+1], …, a[r] είναι > a[i]
- Ποιο στοιχείο να επιλέξουμε για pivot?
- Για τυχαία δεδομένα δεν παίζει σπουδαίο ρόλο ποιο θα επιλέξουμε

Υλοποίηση της διαμέρισης

- Υλοποίηση με pivot = a[r]
- Έστω v = a[r]
- Αναδιάταξη:
- Σαρώνουμε τον πίνακα από τα αριστερά μέχρι να βρούμε στοιχείο ≥ v
- Σαρώνουμε από δεξιά μέχρι να βρούμε στοιχείο ≤ v
- Αντιμεταθέτουμε και συνεχίζουμε μέχρι να συναντηθούν οι αριθμοδείκτες ή να περάσει ο j τον i
- Στο τέλος βάζουμε το v εκεί που έχει σταματήσει ο i



Υλοποίηση της διαμέρισης

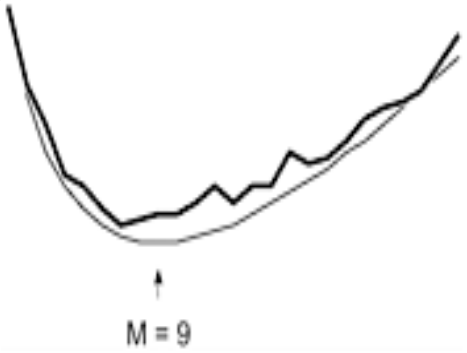
Διαμέριση με `pivot = a[r]`

```
static int partition(ITEM a[], int p, int r)
  { int i = p-1, j = r; ITEM v = a[r];
    for (;;)
    { while (less(a[++i], v)) ;//i δεν βγαίνει εκτός
      while (less(v, a[--j]))
         if (j == p) break;
      if (i >= j) break;
      exch(a, i, j);
    }
    exch(a, i, r);
    return i; }
```

Χαρακτηριστικά Επιδόσεων

- Δεν χρειάζεται πρόσθετο χώρο μνήμης σε αντίθεση με την Mergesort
- Χρόνος εκτέλεσης: Worst case όταν ο πίνακας είναι ήδη ταξινομημένος. Θα χρειαστούμε N + N-1+...+1 = N(N+1)/2 = $O(N^2)$ συγκρίσεις
- Average case: O(NlogN) (απόδειξη πιο δύσκολη)
- Στην πράξη είναι η πιο γρήγορη και ευρέως χρησιμοποιούμενη μέθοδος σήμερα
  - Δεν συναντώνται συχνά δεδομένα που να είναι ήδη ταξινομημένα ή σχεδόν ταξινομημένα
  - Αντιθέτως το worst case των Insertion sort, Bubble sort, Selection sort, είναι πιο πιθανό

Βελτιώσεις και Παραλλαγές



- Όπως και με την Mergesort, η αναδρομή δεν λειτουργεί καλά για μικρά υποαρχεία
- Χρήση Insertion sort σε αρχεία με μέγεθος ≤ M
- Αλλαγή 1ης γραμμής σε

```
if (r-p <= M) insertionsort(a, p, r)
```

- Ή ακόμα καλυτερα σε: `if (r-p <= M) return;`
- Χρήση Insertion sort μόνο στο τέλος:

```
static void sort(ITEM a[], int p, int r) {
  quicksort(a, p, r);
  insertionsort(a, p, r); }
```

Quicksort ή Shellsort?

N	Shellsort	Quicksort	Quicksort με M=3	Quicksort με M=10	Quicksort με M=20
12500	31	23	22	10	9
25000	42	20	19	19	19
50000	94	45	41	39	40
100000	214	95	88	85	90
200000	516	201	194	189	188
400000	1159	449	420	417	404
800000	2678	927	908	870	876

Quicksort ή Mergesort?

- Α: Αναδρομικός Mergesort
- Σ: συνθετικός Mergesort

N	Quicksort	Α	Σ
12500	23	27	30
25000	20	43	42
50000	45	91	92
100000	95	199	204
200000	201	421	455
400000	449	904	992
800000	927	1910	2106

Το βάθρο...

Quicksort

Mergesort

Shellsort

Μη αναδρομική υλοποίηση

- Με χρήση στοίβας
- Μιμείται τις αναδρομικές κλήσεις
- Κάνουμε pop για να επεξεργαστούμε έναν υποπίνακα
- Ωθούμε στη στοίβα τους αριθμοδείκτες των 2 υποπινάκων που προκύπτουν μετά τη διαμέριση
- Βάζουμε πρώτα τον υποπίνακα με το μεγαλύτερο μέγεθος
- Επεξεργαζόμαστε πρώτα τον υποπίνακα με το μικρότερο μέγεθος
- Εξασφαλίζεται έτσι ότι το μέγεθος της στοίβας δεν μεγαλώνει πολύ

Μη αναδρομική υλοποίηση

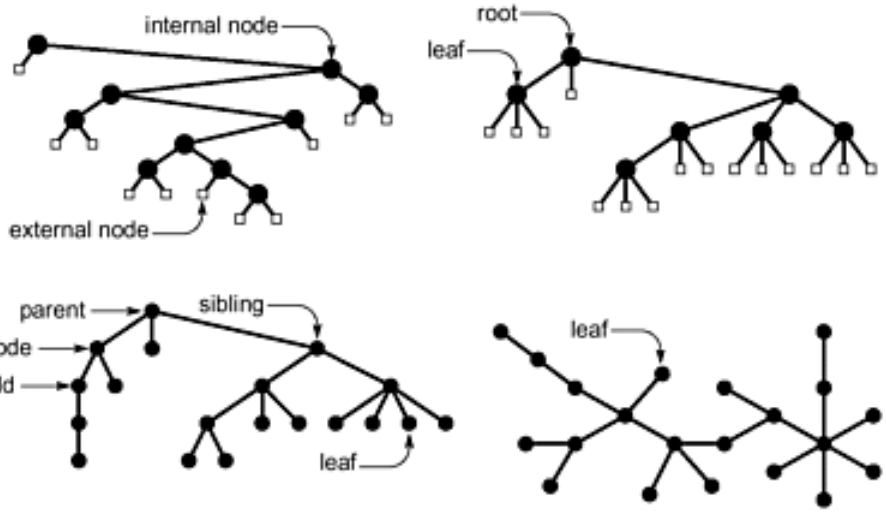
```
static void quicksort(ITEM[] a, int p, int r) {
   intStack S = new intStack(2*N);//N = r-p+1
   S.push(p); S.push(r);
   while (!S.isEmpty())
   {
      r = S.pop(); p = S.pop();
      if (r <= p) continue;
      int i = partition(a, p, r);
      if (i-p > r-i) { S.push(p); S.push(i-1);}
      S.push(i+1); S.push(r);
      if (r-i >= i-p) { S.push(p); S.push(i-1); }
   }
}
```

Δέντρα

- Γράφημα ή γράφος
  - Οποιοδήποτε σύνολο κορυφών (κόμβων) και πλευρών (ακμών)
  - Κάθε πλευρά συνδέει δύο κορυφές
  - Η ακμή μπορεί να είναι κατευθυνόμενη ή όχι
  - *Μονοπάτι*: ακολουθία από διαδοχικές πλευρές
- *Δέντρο*: ειδική μορφή γραφήματος
  - Ανάμεσα σε δύο κόμβους υπάρχει ένα ακριβώς μονοπάτι
  - Μας βοηθά στην αναπαράσταση της εκτέλεσης αλγορίθμων (union-find, divide and conquer,…)
- *Δέντρο με ρίζα*: ειδική μορφή δέντρου
  - Δέντρο όπου ένας κόμβος ορίζεται ως ρίζα
    - Συνήθως θεωρούμε ότι οι ακμές κατευθύνονται προς ή από τη ρίζα
  - Σχεδόν πάντα αναφερόμαστε σε δέντρα με ρίζα
  - *Υποδέντρο*: δέντρο που ξεκινά από κόμβο διαφερετικό από τη ρίζα

Δέντρα

- Ορολογία δένδρων
  - Γονέας ενός κόμβου: ο αμέσως από πάνω κόμβος (με κατεύθυνση προς τη ρίζα)
    - Η ρίζα είναι ο μόνος κόμβος που δεν έχει γονέα
  - Παιδιά ενός κόμβου: οι αμέσως από κάτω κόμβοι
  - Αδέλφια: παιδιά του ίδιου γονέα
  - Μη τερματικός κόμβος: έχει ένα τουλάχιστον παιδί
  - Φύλλο ή τερματικός κόμβος: δεν έχει κανένα παιδί
  - Δάσος: συλλογή από δέντρα ξένα μεταξύ τους



Δέντρα

- Διατεταγμένο δέντρο
  - Ένα δέντρο στο οποίο η σειρά των παιδιών έχει σημασία
  - Παράδειγμα: τα παιδιά έχουν αύξουσα διάταξη κλειδιών
- Μ-αδικό δέντρο
  - Κάθε μη τερματικός κόμβος έχει Μ παιδιά
  - Συχνά θεωρούμε ότι υπάρχουν εξωτερικοί κόμβοι
    - Χρησιμεύουν στο να έχουν όλοι οι κόμβοι ακριβώς Μ παιδιά
- Δυαδικό δέντρο
  - Ειδική περίπτωση διατεταγμένου Μ-αδικού δέντρου με Μ=2
  - Ονομάζουμε τα παιδιά αριστερό και δεξιό
- Αναδρομικοί ορισμοί: χρήσιμοι στην επαγωγή
  - Δυαδικό δέντρο: Είτε ένας κόμβος χωρίς παιδιά,
  - Είτε ένας κόμβος με παιδιά δύο δυαδικά δέντρα

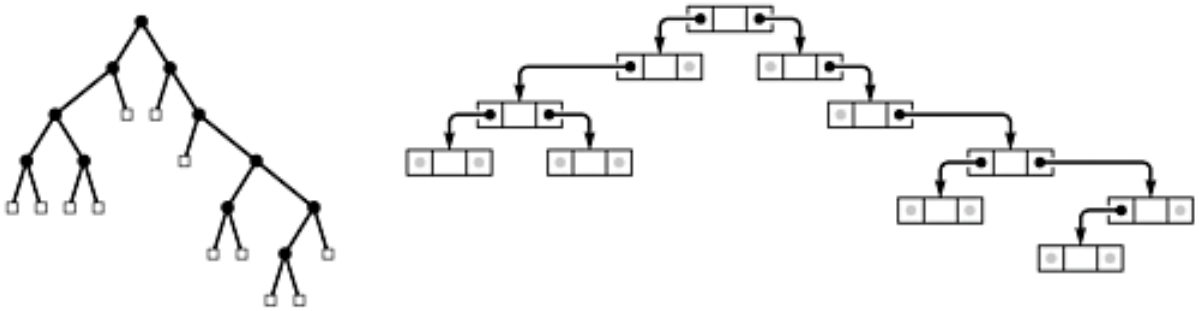
Δέντρα

- Αναπαράσταση δέντρων
- Μπορεί να γίνει με πολλούς τρόπους ανάλογα με την εφαρμογή

1. Πίνακες ή λίστες γειτνίασης (ισχύει για κάθε γράφημα)
2. Πίνακας parent

- parent[i] δείχνει τον γονέα του i
- Για τη ρίζα κάνουμε τη σύμβαση ότι parent[i] = i

Δέντρα

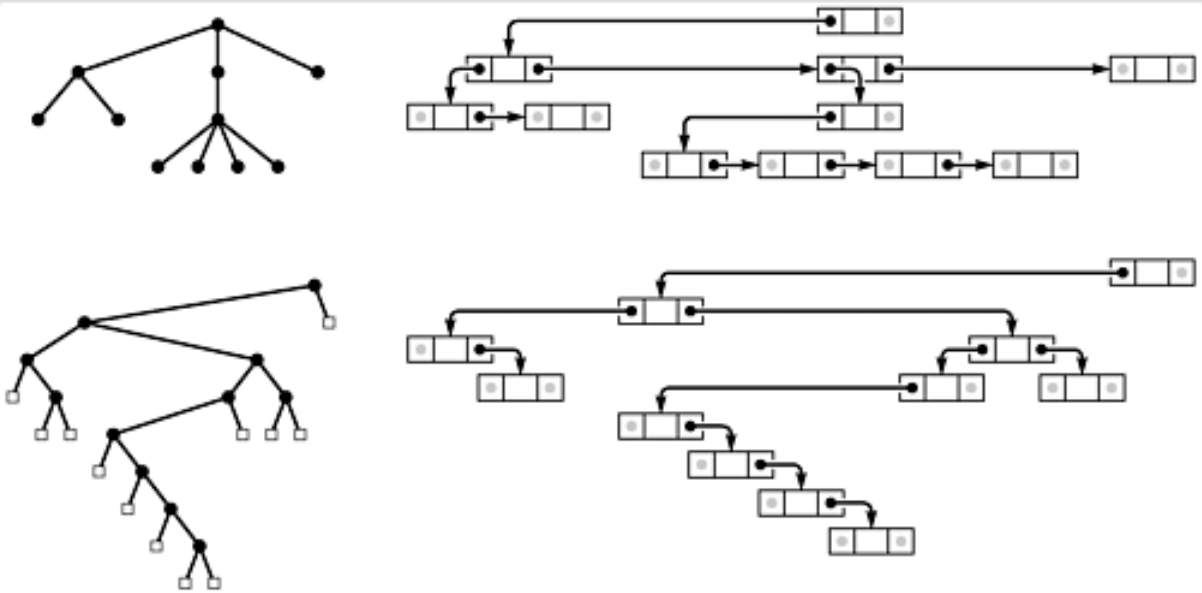


- Αναπαράσταση δυαδικών δέντρων
  - Ορίζουμε κόμβους με στοιχείο και δύο συνδέσμους προς αριστερό και δεξιό υποδέντρο
  - Οι κόμβοι χωρίς παιδιά έχουν συνδέσμους null

```
Class Node {
  Item item; Node l; Node r;
  Node(Item v, Node l, Node r) {
       this.item = v; this.l = l; this.r = r; }}
```

  - Κατάλληλη για κίνηση προς τα κάτω
  - Για κίνηση προς τα πάνω προσθέτουμε σύνδεσμο στο γονέα

Δέντρα



- Αναπαράσταση γενικών δέντρων
  - Στα Μ-αδικά δέντρα γενικεύουμε τα δυαδικά δέντρα
    - Είτε Μ επώνυμοι σύνδεσμοι, είτε πίνακας με Μ συνδέσμους
  - Στα γενικά δέντρα χρήση συνδεδεμένης λίστας παιδιών
    - Αριστερός δείκτης: λίστα παιδιών, δεξιός δείκτης: επόμενος αδελφός
    - Υπάρχει ισοδύναμη αναπαράσταση με δυαδικό δέντρο

Μαθηματικές ιδιότητες

- Ικανές και αναγκαίες συνθήκες
- Ένας γράφος G με N κορυφές είναι δέντρο αν και μόνο αν ισχύει μία από τις συνθήκες:
- Ο G έχει N-1 πλευρές και κανένα κύκλο
- Ο G έχει N-1 πλευρές και είναι συνδεδεμένος (για κάθε ζεύγος κορυφών υπάρχει μονοπάτι που τις συνδέει)
- Κάθε ζεύγος κορυφών συνδέεται με ένα μόνο μονοπάτι
- Ο G είναι συνδεδεμένος αλλά παύει να είναι αν αφαιρέσουμε οποιαδήποτε πλευρά
- Οι συνθήκες αυτές είναι ισοδύναμες μεταξύ τους

Μαθηματικές ιδιότητες

- Ιδιότητες δυαδικών δένδρων
  - Συνήθως αποδεικνύονται με επαγωγή
- Πλήθος κόμβων
  - Έστω δυαδικό δέντρο με N εσωτερικούς κόμβους
  - Το δέντρο έχει N+1 τερματικούς κόμβους
- Πλήθος ακμών
  - Έστω δυαδικό δέντρο με N εσωτερικούς κόμβους
  - Το δέντρο έχει 2N συνδέσμους
    - N-1 σύνδεσμοι μεταξύ εσωτερικών κόμβων
    - N+1 σύνδεσμοι από εξωτερικούς κόμβους
- Επίπεδο κόμβου
  - Η ρίζα έχει επίπεδο 0
  - Άλλοι κόμβοι: επίπεδο ίσο με 1 παραπάνω από τον γονέα τους

Μαθηματικές ιδιότητες

- Άλλοι ορισμοί δένδρων
  - Ύψος δέντρου: μέγιστο επίπεδο κόμβων
  - Μήκος διαδρομής: άθροισμα επιπέδων όλων των κόμβων
  - Μήκος εσωτερικής και εξωτερικής διαδρομής
  - Πλήρες δυαδικό δέντρο (complete binary tree): όλα τα επίπεδα γεμάτα εκτός τελευταίου
    - Στο τελευταίο επίπεδο είναι γεμάτοι οι αριστερότεροι κόμβοι
- Μήκη διαδρομών
  - Έστω δυαδικό δέντρο με N εσωτερικούς κόμβους
  - Μήκος εξωτερικής = μήκος εσωτερικής διαδρομής + 2N
  - Μήκος εσωτερικής διαδρομής: Nlg(N/4) έως N(N-1)/2

Μαθηματικές ιδιότητες - Παραδείγματα

- Ύψος δέντρου
  - Έστω δυαδικό δέντρο με N εσωτερικούς κόμβους
  - Το ύψος είναι τουλάχιστον lgN και το πολύ N-1