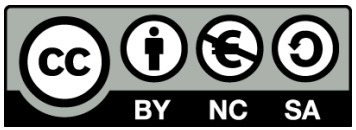
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

ΑΛΓΟΡΙΘΜΟΙ

Ενότητα 6: Μετασχηματίσε και Κυριάρχησε

Ιωάννης Μανωλόπουλος, Καθηγητής
Αναστάσιος Γούναρης, Επίκουρος Καθηγητής
Τμήμα Πληροφορικής ΑΠΘ



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» 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.

Μετασχημάτισε και Κυριάρχησε

Απλοποίηση στιγμιότυπου, αλλαγή αναπαράστασης, αναγωγή προβλήματος

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

Μετασχημάτισε και Κυριάρχησε

Μπορούμε να επιλύσουμε ένα πρόβλημα με μετασχηματισμό σε:

- Ένα περισσότερο βολικό στιγμιότυπο του ιδίου προβλήματος (απλοποίηση στιγμιότυπου)
  - Προταξινόμηση, Gaussian απαλοιφή, αντιστροφή πίνακα, υπολογισμός ορίζουσας
- Μία διαφορετική αναπαράσταση του ίδιου στιγμιότυπου (*αλλαγή αναπαράστασης*)
  - Ισοζυγισμένα δυαδικά δένδρα, σωροί και ταξινόμηση με σωρό, υπολογισμός πολυωνύμου με τον κανόνα του Horner, γρήγορος μετασχηματισμός Fourier
- ένα τελείως διαφορετικό πρόβλημα (*αναγωγή προβλήματος*)
  - Μείωση σε προβλήματα γράφων, γραμμικός προγραμματισμός

Απλοποίηση στιγμιότυπου - Προταξινόμηση

Ένα στιγμιότυπο ενός προβλήματος επιλύεται μετα-σχηματίζοντάς το σε ένα απλούστερο/ευκολότερο στιγμιότυπο του ιδίου προβλήματος

<u>Προταξινόμηση:</u>

Πολλά προβλήματα με λίστες επιλύονται ευκολότερα αν οι λίστες είναι ταξινομημένες

- Μοναδικότητα στοιχείου
- Υπολογισμός της τυπικής τιμής (mode)
- Υπολογισμός επαναλαμβανόμενων στοιχείων
- Αναζήτηση
- Υπολογισμός του μεσαίου

Εύρεση επαναλαμβανόμενων στοιχείων

- Αλγόριθμος ωμής βίας: $\Theta(n^2)$
- Αλγόριθμος με προταξινόμηση:
  - Ταξινόμηση με συγχώνευση: $\Theta(n \lg n)$
  - Σάρωσε τον πίνακα για την εύρεση ίδιων διαδοχικών στοιχείων: $\Theta(n)$
  - Συνολικά: $\Theta(n \lg n)$
- Συμπέρασμα: η προταξινόμηση βελτιώνει σημαντικά

Έλεγχος της τυπικής τιμής

- *Τυπική τιμή (mode)* είναι το συχνότερο στοιχείο
- Ωμή βία: Σάρωσε τη λίστα, υπολόγισε τις συχνότητες, βρες τη μεγαλύτερη
- ***Algorithm* PresortedMode**

```
Sort the array A
i←0; modefrequency←0;
while i ≤ n-1 do
      runlength←1; runvalue←A[i];
      while i+runlength≤n-1 and A[i+runlength]=runvalue
            runlength←runlength+1
      if runlength>modefrequency
            modefrequency←runlength, modevalue←runvalue
      i←i+runlength
return modevalue
```

- Συμπέρασμα: η προταξινόμηση και πάλι βοηθά την επίδοση

Έλεγχος μοναδικότητας στοιχείου

- Αλγόριθμος ωμής βίας: $\Theta(n^2)$
- ***Algorithm*** **PresortedElementUniqueness**

```
Sort the array A
for i ← 0 to n-2 do
    if A[i]=A[i+1] return false
    else return true
```

- Συμπέρασμα: η προταξινόμηση βελτιώνει την επίδοση
- Ίδια βελτίωση για την εύρεση της τυπικής τιμής

Το πρόβλημα της επιλογής

Βρες το *k*-οστό μικρότερο στοιχείο μεταξύ των A[1],…,A[*n*]

- *Το ελάχιστο: k* = 1, *το μέγιστο: k = n, το μεσαίο:* $k = \lceil n/2 \rceil$

- Αλγόριθμοι με προταξινόμηση
  - Ταξινόμησε τη λίστα
  - Επίστρεψε το A[*k*]
- Αλγόριθμοι με διαμερισμό (μείωσε και βασίλευε):
  - Με διαμερισμό τοποθέτησε τον pivot στη θέση A[*s*]
  - Αν *s=k* , τότε επίστρεψε A[*s*]
  - Αλλιώς αν *s<k, τότε επανάλαβε με την υπολίστα* A[*s*+1],…,A[*n*]
  - *Αλλιώς αν s>k, τότε επανάλαβε με την υπολίστα* A[1],…,A[*s*-1]

Σημειώσεις στο πρόβλημα της επιλογής

- Αλγόριθμοι με προταξινόμηση:
  $\Omega(n\lg n) + \Theta(1) = \Omega(n\lg n)$
- Αλγόριθμοι με διαμερισμό (μείωσε και βασίλευε):
  - Χειρότερη περίπτωση: $T(n) = T(n-1) + (n+1) \rightarrow \Theta(n^2)$
  - Καλύτερη περίπτωση: $\Theta(n)$
  - Μέση περίπτωση: $T(n) = T(n/2) + (n+1) \rightarrow \Theta(n)$
  - Bonus: βρίσκει επίσης τα k μικρότερα στοιχεία
- Ειδικές περιπτώσεις max, min: καλύτεροι και απλούστεροι γραμμικοί αλγόριθμοι (ωμή βία)
- Συμπέρασμα: η προταξινόμηση δεν βοηθά την περίπτωση αυτή

Gaussian απαλοιφή

- Δίνεται σύστημα δύο εξισώσεων με δύο αγνώστους

  $a_{11}x + a_{12}y = b_1$

  $a_{21}x + a_{22}y = b_2$
- Έχει μία μοναδική λύση εκτός αν οι συντελεστές είναι ανάλογοι
- Εκφράζουμε τη μία μεταβλητή ως συνάρτηση της άλλης και αντικαθιστούμε για να επιλύσουμε μία εξίσωση
- Τι συμβαίνει σε σύστημα με *n* εξισώσεις και *n* αγνώστους;

Gaussian απαλοιφή (2)

- Μετασχηματίζουμε τη Ax=b σε A'x=b', όπου ο πίνακας A' είναι άνω τριγωνικός
- Η λύση είναι εφικτή με αντικατάσταση προς τα πίσω
- Βασικές πράξεις
  - Ανταλλαγή εξισώσεων
  - Αντικατάσταση μίας εξίσωσης με ένα πολλαπλάσιο (μη μηδενικό)
  - Αντικατάσταση μίας εξίσωσης με το άθροισμα ή τη διαφορά της εξίσωσης αυτής με το πολλαπλάσιο μίας άλλης
- Παράδειγμα

$$2x_1 - x_2 + x_3 = 1$$

$$4x_1 + x_2 - x_3 = 5$$

$$x_1 + x_2 + x_3 = 0$$

Gaussian απαλοιφή (3)

Algorithm GaussElimination

```
for i ← 1 to n do A[i,n+1]←b[i]
for i← 1 to n-1 do
    for j←i+1 to n do
        for k←i to n+1 do
            A[j,k]←A[j,k]-
                A[i,k]*A[j,i]/A[i,i]
```

- Προβληματική λύση αν το A[i,i] είναι 0 ή πολύ μικρό

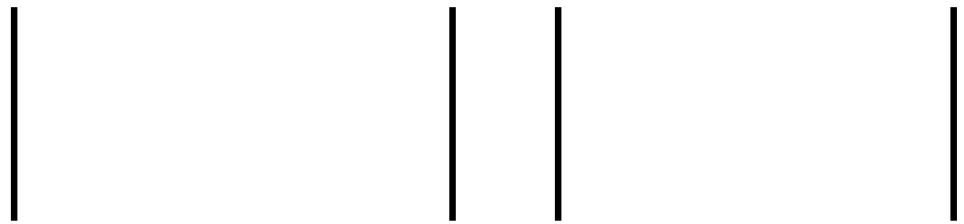
Gaussian απαλοιφή (μερική περιστροφή)

Algorithm GaussElimination2

```
for i ← 1 to n do A[i,n+1]←b[i]
for i← 1 to n-1 do
    pivotrow←i
    for j←i+1 to n do
        if |A[j,i]|>|A[pivot,i]| pivotrow←j
    for k←i to n+1 do
        swap(A[i,k],A[pivotrow,k])
    for j←i+1 to n do
        temp←A[j,i]/A[i,i]
        for k←I to n+1 do
            A[j,k]←A[j,k]-A[I,k]*temp
```

LU αποσύνθεση

- Υποπροϊόν της Gaussian απαλοιφής
- Παράδειγμα A=LU



- LUx=b. Συμβολίζουμε y=Ux → Ly=b
- Επιλύουμε το Ly=b, και μετά το Ux=y
- Επιλύουμε όσες φορές χρειάζεται με διαφορετικά b
- Δεν χρειάζεται έξτρα χώρος

Αντιστροφή πίνακα

- $AA^{-1}=I$
- Ο *ιδιάζων (singular)* πίνακας δεν έχει αντίστροφο
- Ένας πίνακας είναι ιδιάζων αν και μόνο αν κάποια από τις γραμμές είναι γραμμικός συνδυασμός άλλων γραμμών
- Εφαρμογή Gaussian απαλοιφής. Αν προκύψει άνω-τριγωνικός πίνακας χωρίς κάποιο μηδενικό στη διαγώνιο, τότε ο πίνακας δεν είναι ιδιάζων
- Πως θα βρούμε τον αντίστροφο ενός πίνακα;
- $Ax^j=e^j$

Υπολογισμός της ορίζουσας

- Ο γνωστός αναδρομικός τύπος
- Τι συμβαίνει αν το *n* είναι μεγάλο; Απόδοση;
- Εφαρμογή Gaussian απαλοιφής
- Η ορίζουσα ενός άνω-τριγωνικού πίνακα ισούται με το γινόμενο των στοιχείων της διαγωνίου
- Απόδοση;

- Ο κανόνας του Cramer

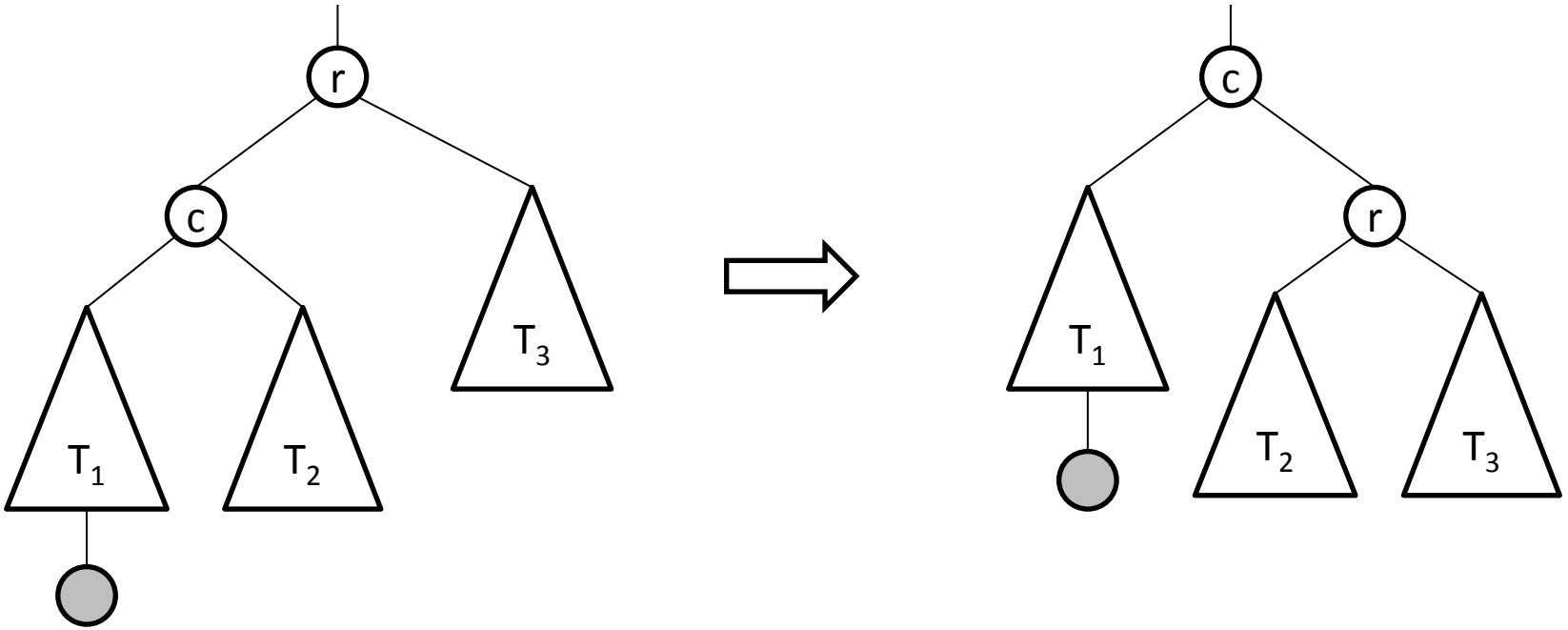
Ταξινόμηση αλγορίθμων αναζήτησης

- Στοιχειώδεις αλγόριθμοι αναζήτησης
  - Σειριακή αναζήτηση
  - Δυαδική αναζήτηση
  - Αναζήτηση δυαδικού δένδρου
- Αναζήτηση ισοζυγισμένου δένδρου
  - Δένδρα AVL
  - Κόκκινα-μαύρα δένδρα
  - Πολυκατευθυνόμενα ισοζυγισμένα δένδρα (δένδρα 2-3, δένδρα 2-3-4, B-δένδρα)
- Κατακερματισμός
  - Ξεχωριστές αλυσίδες
  - Ανοικτές διευθύνσεις

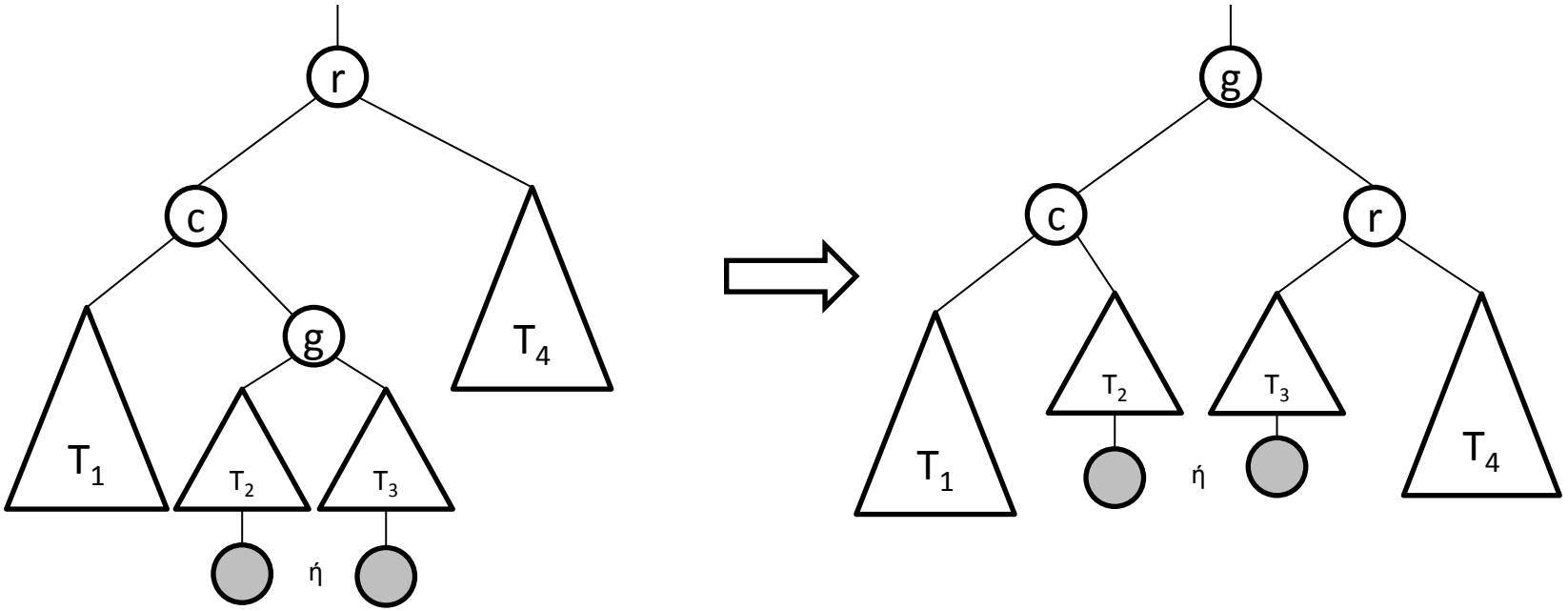
Ισοζυγισμένα δένδρα: δένδρα AVL

- Σε κάθε κόμβο, η διαφορά των υψών του αριστερού και του δεξιού υποδένδρου είναι το πολύ 1
- Η ιδιότητα AVL διατηρείται με περιστροφές, κάθε φορά που το δένδρο δεν είναι ισοζυγισμένο
- $\lg n \leq h \leq 1.4404 \lg (n + 2) - 1.3277$
  στη μέση περίπτωση: $1.01 \lg n + 0.1$ για μεγάλα n
- Μειονέκτημα: θέλει έξτρα χώρο για την διατήρηση του παράγοντα ισοζυγισμού των κόμβων
- Μία παρόμοια ιδέα: τα κόκκινα-μαύρα δένδρα (τα ύψη των υποδένδρων μπορούν να διαφέρουν μέχρι 2 φορές)

Γενική περίπτωση: απλή R-περιστροφή

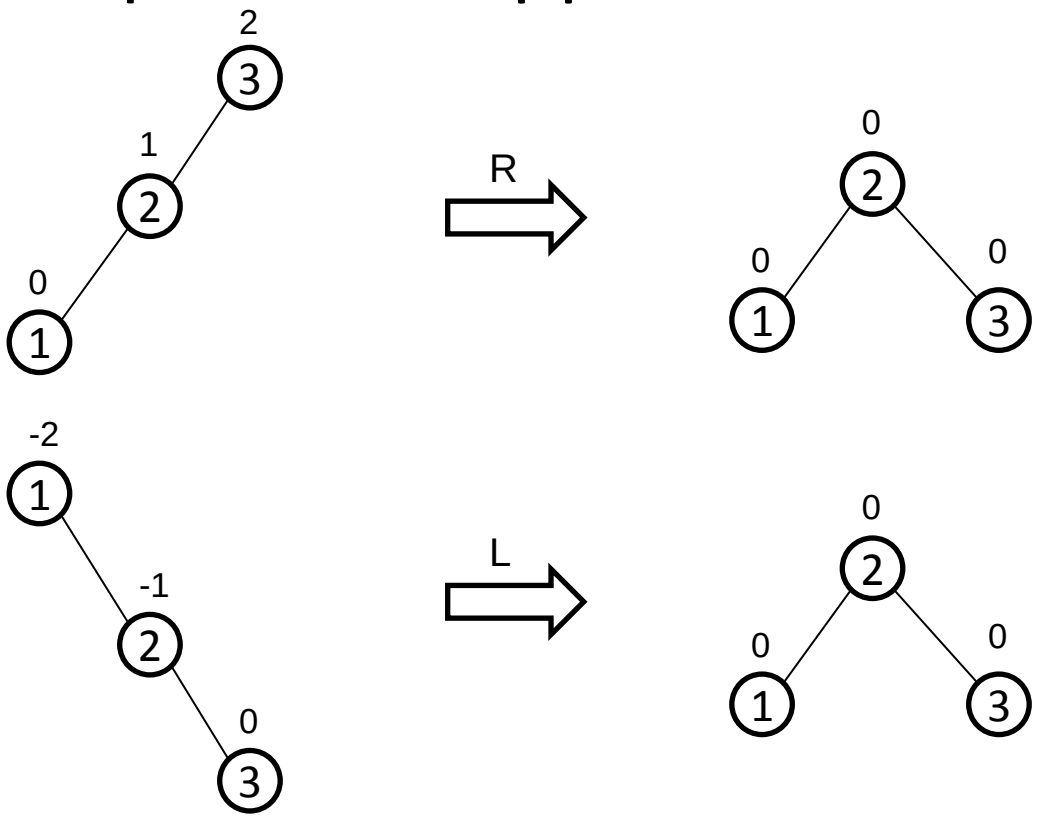


Διπλή LR-περιστροφή



Παράγοντας ισοζυγισμού

Ο αλγόριθμος διατηρεί έναν παράγοντα ισοζυγισμού για κάθε κόμβο

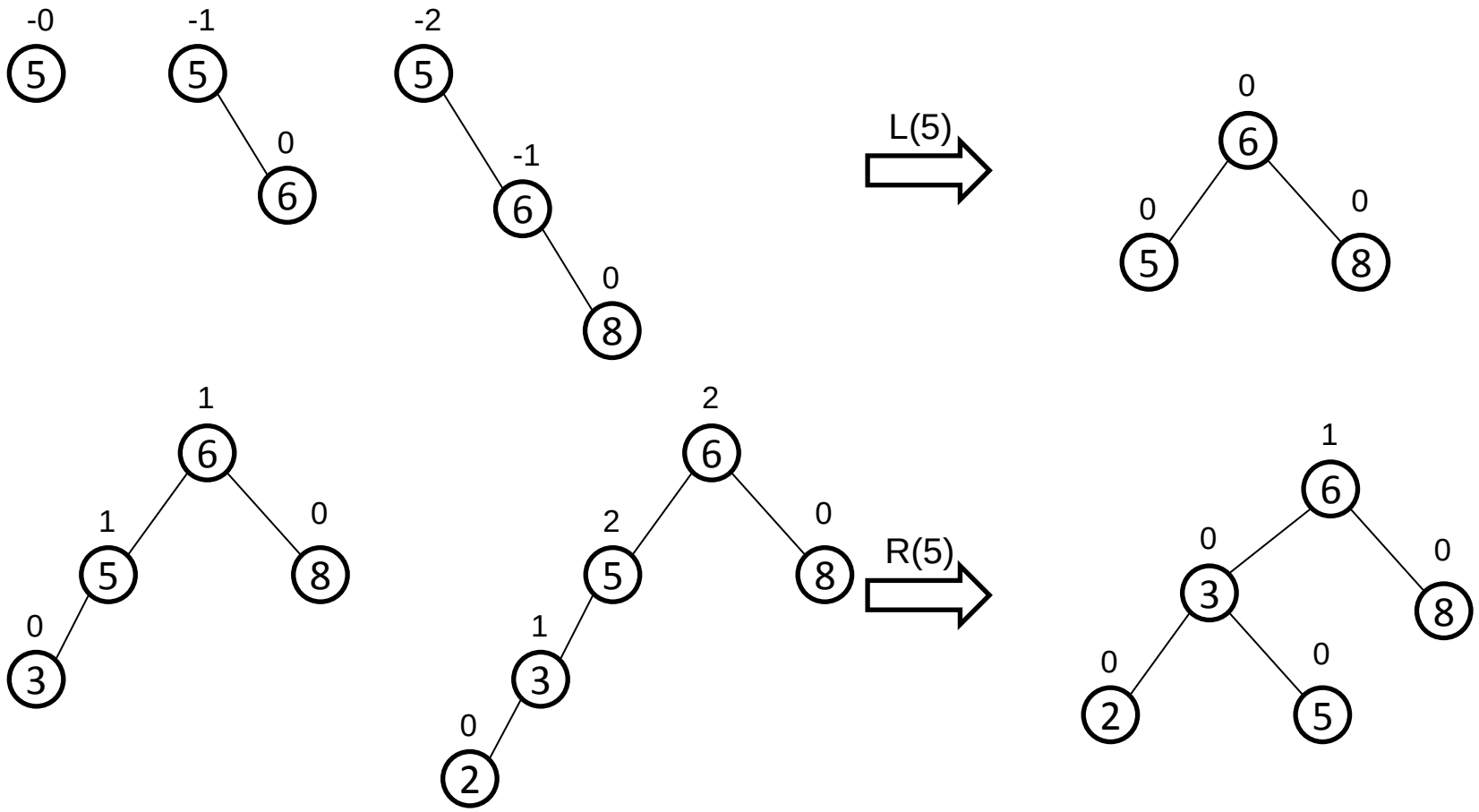


AVL περιστροφές

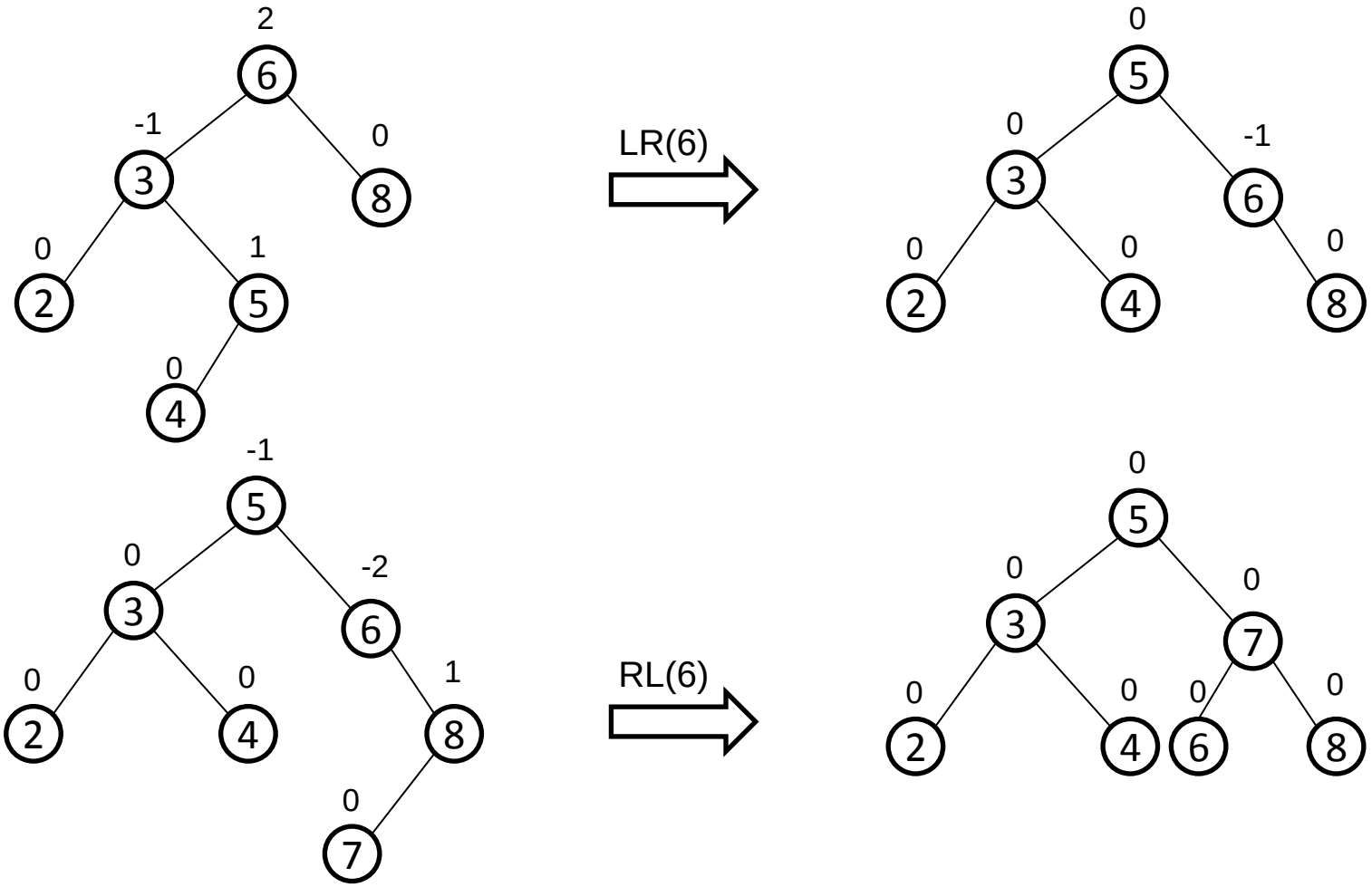
- Μικρά παραδείγματα:
  - 1, 2, 3
  - 3, 2, 1
  - 1, 3, 2
  - 3, 1, 2
- Μεγαλύτερο παράδειγμα: 4, 5, 7, 2, 1, 3, 6
- Στα προηγούμενα σχήματα δίνονται οι γενικές περιπτώσεις των περιστροφών

Επιπλεόν Παράδειγμα

- Κατασκευή ενός AVL δέντρου για τη λίστα: 5, 6, 8, 3, 2, 4, 7



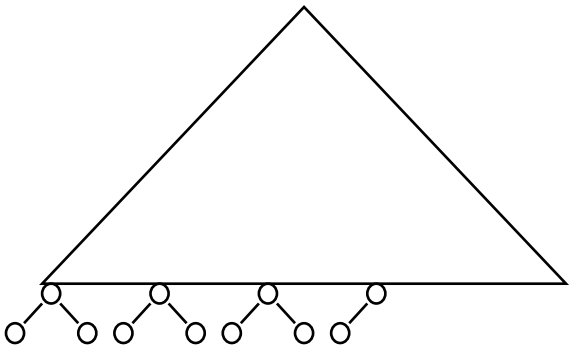
Συνέχεια Παραδείγματος



Σωρός

Ορισμός: Ο σωρός (*heap)* είναι ένα δυαδικό δένδρο με τις εξής προϋποθέσεις:

- Είναι σχεδόν πλήρες:



- Το κλειδί κάθε κόμβου είναι ≥ από τα κλειδιά των παιδιών του

Συνέπειες ορισμού

- Δεδομένου του *n,* υπάρχει ένα μοναδικό δυαδικό δένδρο με *n* κόμβους που είναι σχεδόν πλήρες με $h = \lfloor \lg n \rfloor$
- Η ρίζα έχει το μεγαλύτερο κλειδί
- Καθένα από τα δύο υποδένδρα είναι επίσης σωροί

Ταξινόμηση με σωρό

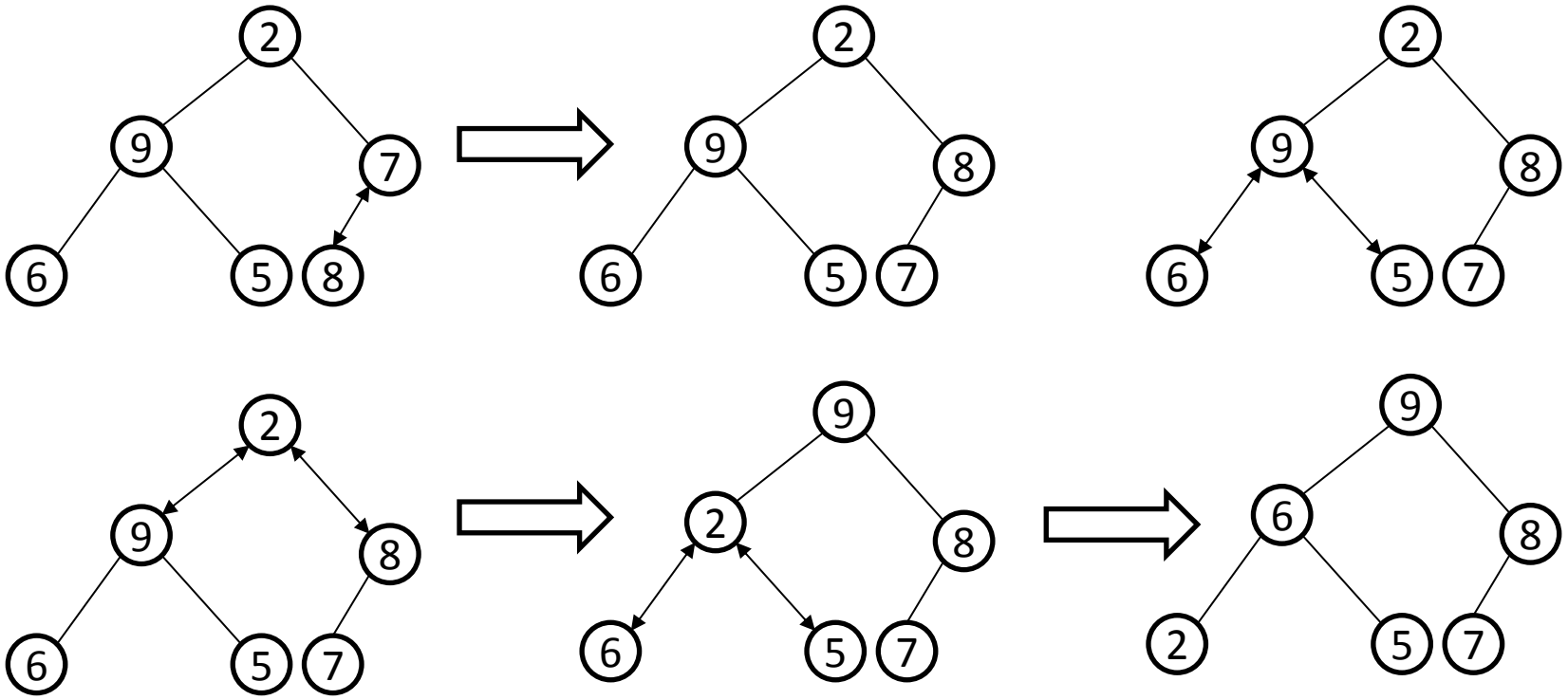
1. Κτίζουμε το σωρό
2. Απομακρύνουμε τη ρίζα – ανταλλάσσουμε με το τελευταίο (δεξιότερο) φύλλο
3. Φτιάχνουμε και πάλι το σωρό (πλην του τελευταίου φύλλου)
4. Επαναλαμβάνουμε τα βήματα 2,3 μέχρι ο σωρός να περιέχει ένα μόνο στοιχείο

Κτίσιμο του σωρού

- Εισάγουμε τα στοιχεία με τη δεδομένη σειρά στο δυαδικό δένδρο με bfs
- Ξεκινώντας από τον τελευταίο (δεξιότερο) κόμβο που είναι πατέρας, τακτοποιούμε τον αντίστοιχο σωρό, αν δεν ικανοποιεί τις συνθήκες του ορισμού:
  1. Ανταλλάσσουμε με το μεγαλύτερο παιδί
  2. Τακτοποιούμε το υποδένδρο που βρίσκεται στη θέση του μεγαλύτερου παιδιού

Παράδειγμα: Κτίσιμο σωρού

Παράδειγμα: 2, 9, 7, 6, 5, 8



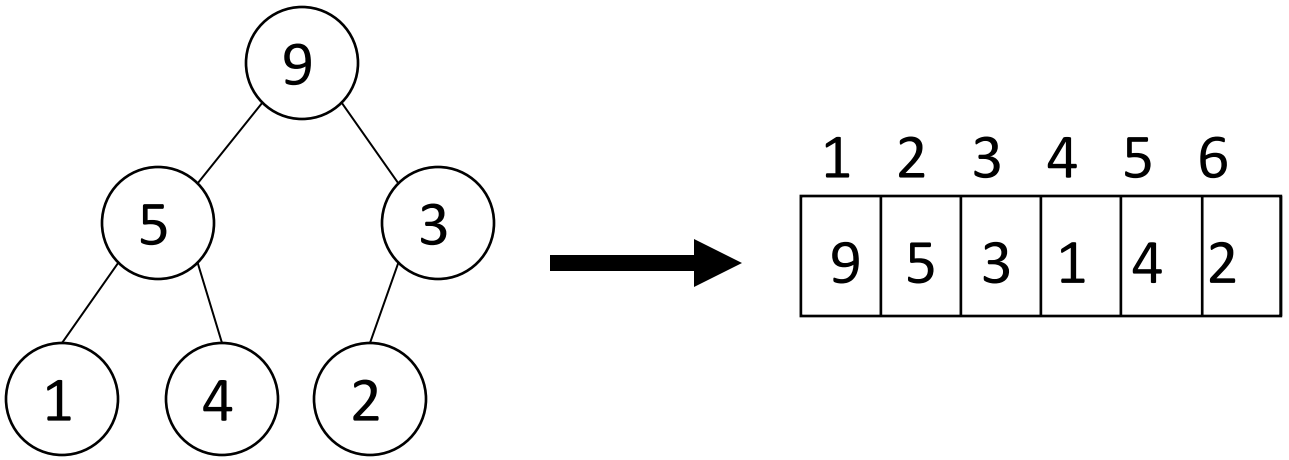
Διαγραφή ρίζας

Η ρίζα του σωρού εξάγεται και ο σωρός τακτοποιείται ως εξής:

- Η ρίζα ανταλλάσσεται με το τελευταίο φύλλο
- Η νέα ρίζα συγκρίνεται με κάθε παιδί της και αν ένα από αυτά είναι μεγαλύτερο από τη ρίζα, τότε το ανταλλάσσουμε με αυτή
- Συνεχίζουμε τις συγκρίσεις/ανταλλαγές με τα παιδιά της νέας ρίζας μέχρι να φθάσει σε ένα επίπεδο του δένδρου όπου θα είναι μεγαλύτερη από τα αντίστοιχα παιδιά

Αναπαράσταση

- Χρησιμοποιούμε ένα πίνακα για να αποθηκεύσουμε το σωρό με bfs
- Παράδειγμα:



- Το αριστερό παιδί του κόμβου j είναι το $2j$
- Το δεξιό παιδί του κόμβου j είναι το $2j+1$
- Ο πατέρας του κόμβου j είναι ο $\lfloor j/2 \rfloor$
- Οι πατρικοί κόμβοι βρίσκονται στις πρώτες $\lfloor n/2 \rfloor$ θέσεις

Αλγόριθμος κατασκευής σωρού

Algorithm HeapBottomUp (H[1 . . . n])

```
//Constructs a heap from the elements of a given array
//by the bottom-up algorithm
//Input: An array  H[1 . . .  n] of orderable items
//Output: A heap H[1 . . . n]
for i ← n/2 downto 1 do
        k ← i; v ← H[k]; heap ← false
        while not heap and 2 * k ≤ n do
                j ← 2 * k
                if  j < n //there are two children
                    if   H[j] < H[j+1]: j ← j + 1
                        if   v ≥ H[j]: heap ← true
                        else  H[k] ← H[j]; k ← j
        H[k] ← v
```

Ανάλυση της ταξινόμησης με σωρό

Με βάση την προηγούμενη διαφάνεια

- Η τακτοποίηση του σωρού στο ύψος j απαιτεί $2j$ συγκρίσεις
- Για το υποδένδρο στο επίπεδο i εκτελούνται $2(h-i)$ συγκρίσεις
- Συνολικά: h-1

$$\sum_{i=0} 2(h-i)\, 2^i = 2\,(n - \lg(n+1)) = \Theta(n)$$

nodes at level i

Ανάλυση της ταξινόμησης με σωρό (2)

Συνολικά για τον αλγόριθμο:

1. Κτίσιμο σωρού: Θ(n)
2. Εξαγωγή ρίζας – ανταλλαγή με το τελευταίο φύλλο
3. Τακτοποίηση σωρού (εκτός του τελευταίου φύλλου): Θ(logn)
4. Επανάληψη (n-1) φορές των 2, 3 μέχρι ο σωρός να περιέχει ένα φύλλο

Χειρότερη περίπτωση: $\Theta(n) + \Theta(n \log n) = \Theta(n \log n)$

Μέση περίπτωση επίσης $\Theta(n \log n)$

Παράδειγμα Heapsort

Να ταξινομηθεί η λίστα 2, 9, 7, 6, 5, 8

Κτίσιμο σωρού

1 9 7 6 5 8

2 9 8 6 5 7

2 9 8 6 5 7

9 2 8 6 5 7

9 6 8 2 5 7

Διαγραφή ρίζας

9 6 8 2 5 7

7 6 8 2 5 | 9

8 6 7 2 5 | 9

5 6 7 2 | 8 9

7 6 5 2 | 8 9

2 6 5 | 7 8 9

6 2 5 | 7 8 9

5 2 | 6 7 8 9

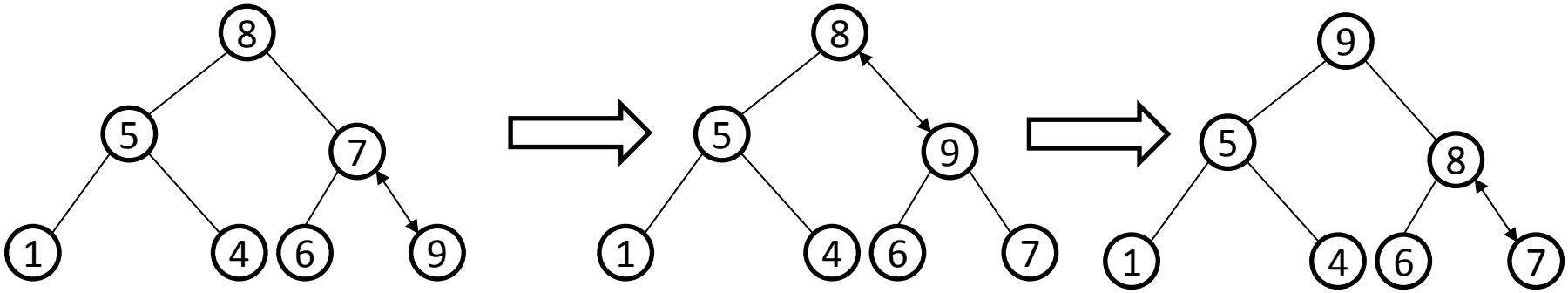
5 2 | 6 7 8 9

2 | 5 6 7 8 9

Εισαγωγή σε σωρό

- Εισαγωγή στοιχείου στην τελευταία θέση του σωρού
- Σύγκριση με τον πατέρα, αν παραβιάζει ο ορισμός, τότε τους ανταλλάσουμε
- Συνεχίζουμε τις συγκρίσεις του νέου στοιχείου με τους κόμβους που βρίσκονται στο μονοπάτι προς τη ρίζα μέχρι να ισχύει ο ορισμός του σωρού

Παράδειγμα: εισαγωγή του στοιχείου 9



Αποτελεσματικότητα: O(log *n*)

Κατασκευή σωρού

- Από επάνω προς τα κάτω: Ο σωρός μπορεί να κατασκευασθεί με διαδοχικές εισαγωγές σε ένα αρχικά κενό σωρό
- Από κάτω προς τα επάνω: Βάζουμε όλα τα στοιχεία στον πίνακα και τα τακτοποιούμε

- Ποιο είναι καλύτερο;

Ουρά προτεραιότητας

- Η ουρά προτεραιότητας είναι ένας αφηρημένος τύπος δεδομένων (ADT) ενός αταξινόμητου συνόλου με τις εξής πράξεις:
  - Βρες το στοιχείο με τη μεγαλύτερη προτεραιότητα
  - Διάγραψε το στοιχείο με τη μεγαλύτερη προτεραιότητα
  - Εισάγαγε ένα στοιχείο με τυχαίο προτεραιότητα
- Ο σωρός είναι ιδανική δομή για την υλοποίηση ουρών προτεραιότητας

Ο κανόνας του Horner

- Ανακοινώθηκε στις αρχές του 19ου αιώνα
- Κατά τον Knuth, η μέθοδος χρησιμοποιήθηκε από το Νεύτωνα
- Υπολογισμός του πολυωνύμου στο σημείο x

  $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

  $p(x) = (\ldots (a_n x + a_{n-1}) x + \ldots) x + a_0$
- Παράδειγμα: υπολογισμός του $p(x)=2x^4-x^3+3x^2+x-5$ στο $x=3$

  $p(x) = x (x (x (2x-1) + 3) + 1) - 5$
- Οπτικοποίηση με πίνακα

Ο κανόνας του Horner (2)

- **Algorithm Horner(P[0..n],x)**

```
// Evaluate polynomial at a given point
// Input: an array P[0..n] of coefficients and a number x
//Output: the value of polynomial at point x
p ← P[n]
for i ← n-1 down to 0 do
    p ← x*p + P[i]
return p
```

- Αποτελεσματικότητα ?
- Παραπροϊόν: συντελεστές του πηλίκου της διαίρεσης του p(x) δια $(x-x_0)$

Δυαδική εκθετικοποίηση

- Ο κανόνας του Horner δεν είναι αποτελεσματικός για τον υπολογισμό του $p(x)=x^n$ at x=a. Εκφυλίζεται σε ωμή βία
- Έστω η δυαδική αναπαράσταση $n=b_l b_{l-1} \ldots b_i \ldots b_1 b_0$
- $n = p(x) = b_l x^l + b_{l-1} x^{l-1} + \ldots + b_1 x + b_0$ για x=2
- **Algorithm LeftRightBinaryExponentiation**

```
product ← a
for i ← l-1 down to 0 do
    product ← product * product
    if b_i ←1 then product ← product*a
return product
```

- Παράδειγμα: υπολογισμός a^{13}, n=13=1101
- Αποδοτικότητα

Δυαδική εκθετικοποίηση (2)

- Υπολογισμός a^n . Θεωρούμε $n = b_l 2^l + b_{l-1} 2^{l-1} + \ldots + b_1 2 + b_0$ και πολλαπλασιάζουμε ανεξαρτήτους όρους
- **Algorithm RightLeftBinaryExponentiation**

```
term ← a
if b0=1 then product ← a
else product ← 1
for i← 1 to l do
    term ← term * term
    if bi = 1 then product ← product * term
return product
```

- Παράδειγμα: υπολογισμός a^{13}, n=13=1101
- Αποδοτικότητα

Ελάχιστο κοινό πολλαπλάσιο

- lcm(24,60)=120, lcm(11,5)=55
- Παράδειγμα: 24 = 2 x 2 x 2 x 3

  60 = 2 x 2 x 3 x 5

  lcm(24,60) = (2x2x3) x 2 x 5
- Αποτελεσματικότητα (απαιτείται λίστα πρώτων)
- lcm(m,n) = mn / gcd(m,n)

Μέτρηση μονοπατιών σε γράφο

- Το πλήθος των διαφορετικών μονοπατιών μήκους $k>0$ από τον κόμβο *i* στον κόμβο *j* ισούται με το στοιχείο (i,j) του πίνακα A^k, όπου A ο πίνακας γειτνίασης
- Παράδειγμα
- Αποδοτικότητα

Μείωση σε προβλήματα γράφων

- Εφαρμόζεται σε πλήθος προβλημάτων παιχνιδιών και γρίφων
- Κτίζουμε το γράφο του χώρου καταστάσεων

- Παράδειγμα: χωρικός, λύκος, κατσίκα, λάχανο
- Διασχίζουμε το γράφο εφαρμόζοντας … ;

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: Ανδρέας Κοσματόπουλος
Θεσσαλονίκη, Αύγουστος 2015

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

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.00.

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.