Βασικές Δομές Δεδομένων

Σύντομη επανάληψη (ΕΠΛ 035).

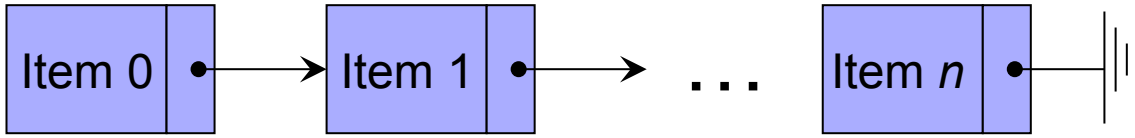
Περίληψη

- Γραμμικές Δομές Δεδομένων
  - Πίνακες
  - Λίστες
- Στοίβες
- Ουρές
- Γράφοι
- Δέντρα

Γραμμικές Δομές

- Πίνακας (array)

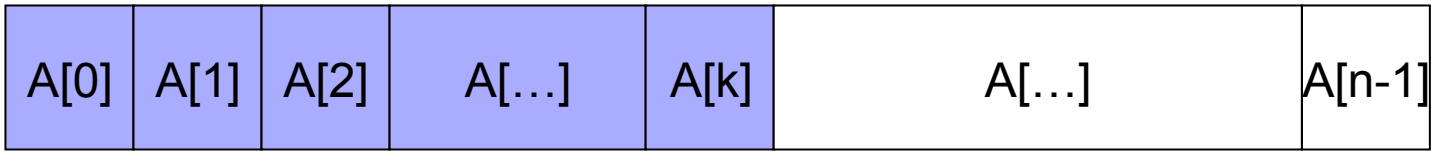
  - Προκαθορισμένη χωρητικότητα (n)
  - Ο χρόνος πρόσβασης (access time) σε οποιοδήποτε στοιχείο του πίνακα είναι σταθερός, ανεξάρτητα από τη θέση του
- Συνδεδεμένη Λίστα (Linked List)



  - Μη προκαθορισμένη χωρητικότητα
  - Ο χρόνος πρόσβασης σε οποιοδήποτε στοιχείο του πίνακα πιθανόν να εξαρτάτε από τη θέση του
- Πώς προσθέτουμε και διαγράφουμε στοιχεία στις πιο πάνω δομές;

Στοίβα

- Βασικές Λειτουργίες
  - Push
  - Pop
- Υλοποίηση με πίνακα



- `A[], top.`
- `push(item)`

```
1. top= top+1;
2. A[top]= item;
```

- `A[], top.`
- `pop()`

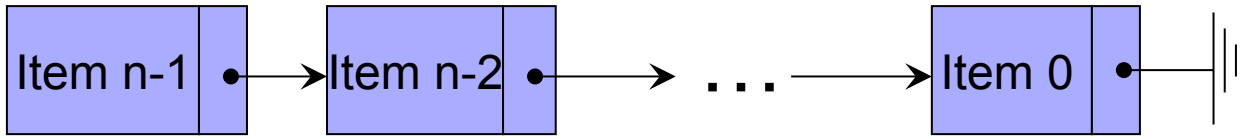
```
1. top= top-1;
2. Return A[top+1];
```

- Έλεγχος για να μην ξεπεραστούν τα όρια του πίνακα.

Στοίβα

- Υλοποίηση με συνδεδεμένη λίστα

top



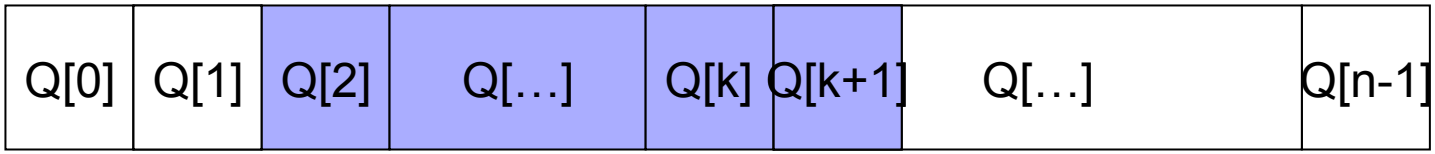
- top.
- push(item)
  1. item.next=top;
  2. top= item;

- top.
- pop()
  1. item= top;
  2. top= item.next;
  3. item.next=null;
  4. Return item;

- Έλεγχος για μη ορισμένους (null) δείκτες.

Ουρά (FIFO Queue)

- Βασικές Λειτουργίες
  - enQueue
  - deQueue
- Υλοποίηση με πίνακα



- `Q[], head, tail.`
- `enQueue(item)`

```
1. tail= (top+1)mod n;
2. Q[tail]= item;
```

- Απαραίτητοι έλεγχοι;

- `Q[], head, tail`
- `deQueue()`

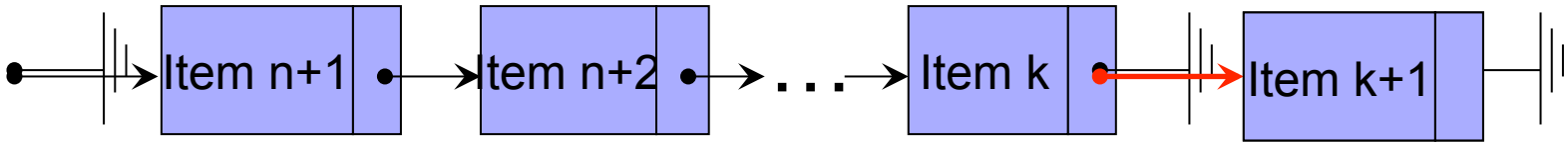
```
1. item= Q[head];
2. head= (head+1)mod n
3. Return item;
```

Ουρά (FIFO Queue)

- Υλοποίηση με συνδεδεμένη λίστα

head

tail



- `head, tail.`
- `enQueue(item)`

```
1. tail.next=item;
2. tail= item;
```

- `head, tail.`
- `deQueue()`

```
1. item= head;
2. head= head.next;
3. item.next=null;
4. Return item;
```

- Απαραίτητοι έλεγχοι;

Πρόβλημα

Υποθέστε πώς

- χρειάζεται να αποθηκεύσετε περίπου 300 στοιχεία σε μία λίστα
- χρειάζεται να έχετε άμεση πρόσβαση στα δεδομένα
- Το «κλειδί» κάθε στοιχείου μπορεί να πάρει τιμές στο διάστημα από 1 μέχρι 2^{32}
- Ποιά δομή θα χρησιμοποιούσατε για την αποθήκευση των δεδομένων;

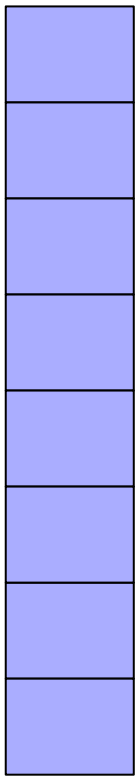
- **Πίνακας (array)**
  - Ο χρόνος πρόσβασης (access time) σε οποιοδήποτε στοιχείο του πίνακα είναι μικρό και σταθερός, ανεξάρτητα από τη θέση του
  - Χρειάζεται πίνακας μεγέθους της τάξης του 2^{32}
- **Συνδεδεμένη λίστα (linked list)**
  - Ο χρόνος πρόσβασης (access time) εξαρτάτε από τη θέση του κάθε στοιχείου και μπορεί να είναι μεγάλος.
  - Χρειάζεται μνήμη **μόνο** για 300 στοιχεία

Hash Table

- Χρησιμοποιούμε πίνακα μεγέθους «λίγο» μεγαλύτερου από ότι χρειάζεται (π.χ. 400)
- Μετατρέπουμε το κλειδί έτσι ώστε να πέρνει τιμές μόνο μεταξύ 1 και του μεγέθους του πίνακα (π.χ. 1 έως 400)
  - Χρησιμοποιώντας ένα «καλό» Hash function
- Σε περίπτωση σύγκρουσης, χρησιμοποιούμε συνδεδεμένη λίστα.

- **Hash Function**
  - Προκαθορισμένο και γνωστό
  - Θα πρέπει να αντιστοιχεί κάθε ένα από τα αναμενόμενα κλειδιά σε «διαφορετική» θέση (με μεγάλη πιθανότητα).

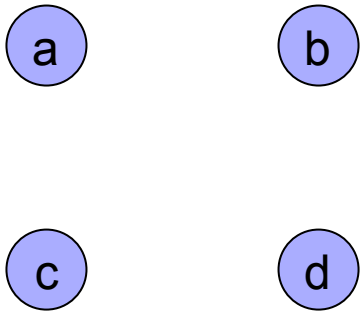
Hash Table



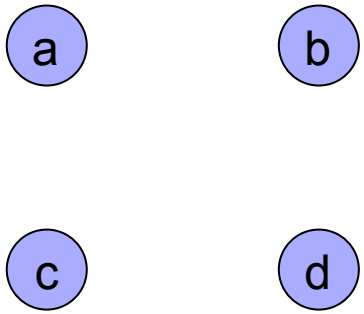
Πίνακας με n θέσεις

Γράφοι (Graphs)

- G=(V, E)
  - V: σύνολο κόμβων (κορυφών – vertices/nodes)
  - E: σύνολο των ακμών (τόξων – edges/arcs)
    - Κατευθυνόμενες και μη κατευθυνόμενες ακμές (directed and undirected edges)
- Παράδειγμα
  - V={a,b,c,d}
  - E={(a,b), (a,c), (b,c), (b,d)}



Μη Κατευθυνόμενες ακμές



Κατευθυνόμενες ακμές

Γράφοι (Graphs)

- **Γειτονικοί κόμβοι** (neighboring/adjacent): Κόμβοι συνδεδεμένοι με μια ακμή
- **Μονοπάτι** (path) μεταξύ 2 κόμβων: Ακολουθία από γειτονικούς κόμβους που ξεκινά από τον ένα και καταλήγει στον άλλο.
- **Απλό (simple) μονοπάτι**: Μονοπάτι στο οποίο όλοι οι κόμβοι εμφανίζονται μόνο μια φορά με πιθανή εξαίρεση τον πρώτο και τελευταίο κόμβο.
- **Κυκλική διαδρομή (cycle):** Απλό μονοπάτι στο οποίο ο πρώτος και τελευταίος κόμβος είναι ο ίδιος
- **Απόσταση (distance) μεταξύ κόμβων**: Το μήκος του ελάχιστου μονοπατιού μεταξύ των κόμβων

Γράφοι (Graphs)

- **Συνδεδεμένος (connected) γράφος**: Γράφος στον οποίο υπάρχει μονοπάτι μεταξύ οποιωνδήποτε δύο κόμβων.
- **Άκυκλος** (acyclic) γράφος: Γράφοι στους οποίους δεν υπάρχουν μονοπάτια που να επιστρέφουν στον κόμβο από όπου ξεκίνησαν.
- **Γράφος με βάρη** (weighted graph): Γράφος στον οποίο κάθε ακμή συσχετίζεται με κάποιο βάρος (ή κόστος).
- **Αραιός** (sparse) γράφος: γράφος με σχετικά λίγες ακμές
- **Πυκνός** (dense) γράφος: μη αραιός γράφος
- **Μέγιστος αριθμός ακμών** (δεν επιτρέπονται self-loops)

Μη Κατευθυνόμενος

Κατευθυνόμενος

Απεικόνιση Γράφων (Graph Representation)

- **Πίνακας Γειτνίασης (Adjacency Matrix)**
  - **A**: Πίνακας |V|x|V|
  - $A[i,j]= 1$ Εάν υπάρχει ακμή μεταξύ του κόμβου i και j
  - $A[i,j]= 0$ Εάν **δεν** υπάρχει ακμή μεταξύ του κόμβου i και j
- **Εάν πρόκειται για γράφο με βάρη**
  - $A[i,j]= w_{ij}$ όπου w_{ij} είναι το βάρος της ακμής μεταξύ του κόμβου i και j
  - $A[i,j]= \infty$ Εάν **δεν** υπάρχει ακμή μεταξύ του κόμβου i και j
- Πόσος ο ελάχιστος χώρος που χρειάζεται ο πίνακας (δεν επιτρέπονται self-loops);

$$|A| = \frac{1}{2}|V| \times (|V| - 1)$$

Μη Κατευθυνόμενος

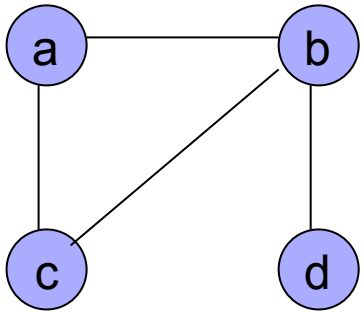
$$|A| = |V| \times (|V| - 1)$$

Κατευθυνόμενος

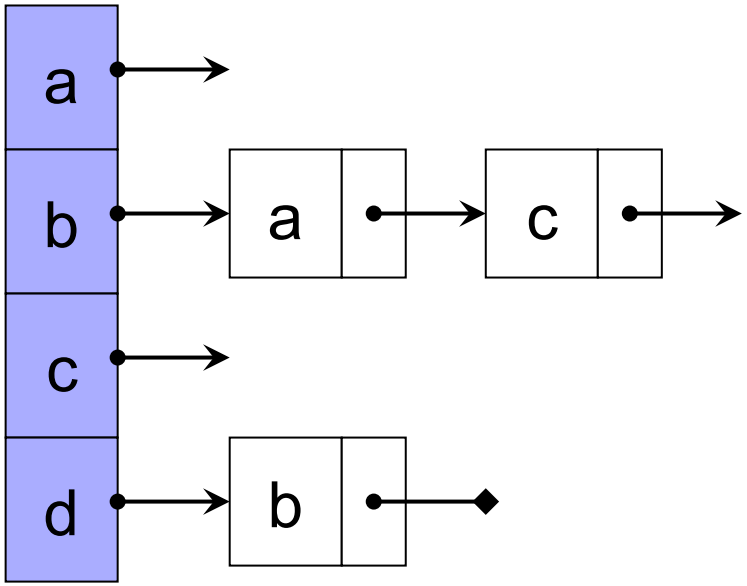
Απεικόνιση Γράφων (Graph Representation)

- **Λίστα Γειτνίασης (Adjacency List)**
  - Ένα διάνυσμα (μονοδιάστατος πίνακας) μεγέθους $|V|$.
  - Από το στοιχείο i του πίνακα ξεκινά μια συνδεδεμένη λίστα
- **Εάν πρόκειται για γράφο με βάρη**
  - Σε κάθε στοιχείο της λίστας υπάρχει επίσης και το βάρος της ακμής που συνδέει τους κόμβους
- Χώρος που χρειάζεται στη μνήμη:
  - Στη χειρότερη περίπτωση: $|V|$x($|V|$-1)
  - Στην περίπτωση αραιών γράφων: $|V|+|E|$

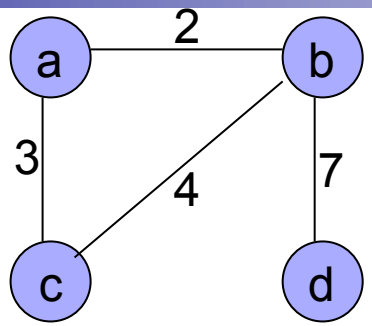
Παράδειγμα



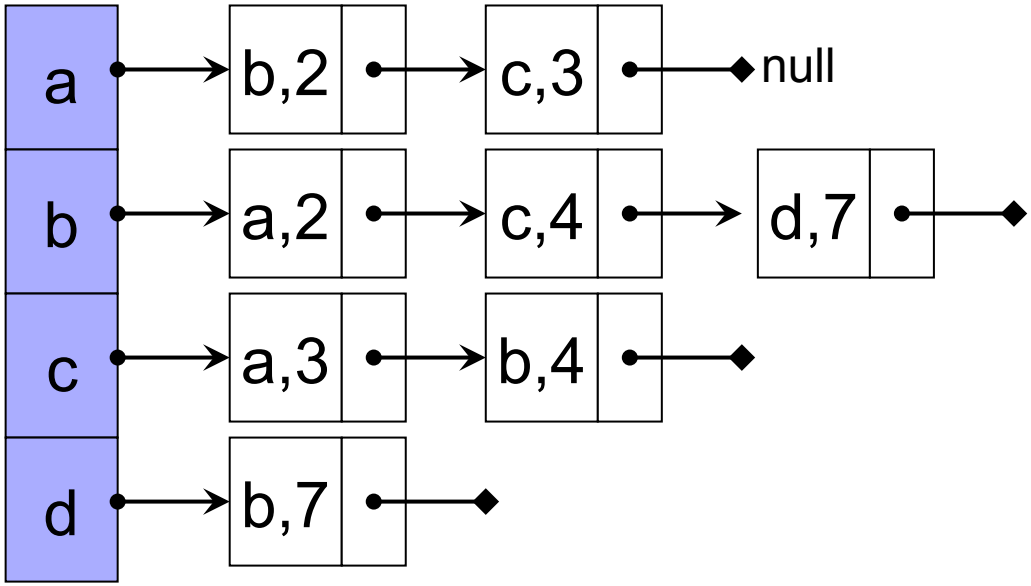
$$A = \begin{bmatrix} & & 0 \\ 1 & & 1 \\ 1 & & 0 \\ 0 & 1 & 0 \end{bmatrix}$$



Παράδειγμα με βάρη

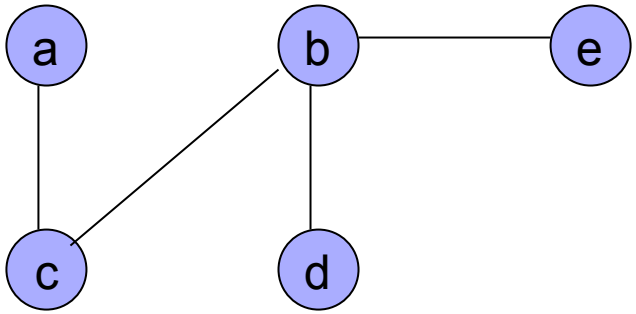


$$A = \begin{bmatrix} \infty & 2 & 3 & \infty \\ 2 & \infty & 4 & 7 \\ 3 & 4 & \infty & \infty \\ \infty & 7 & \infty & \infty \end{bmatrix}$$

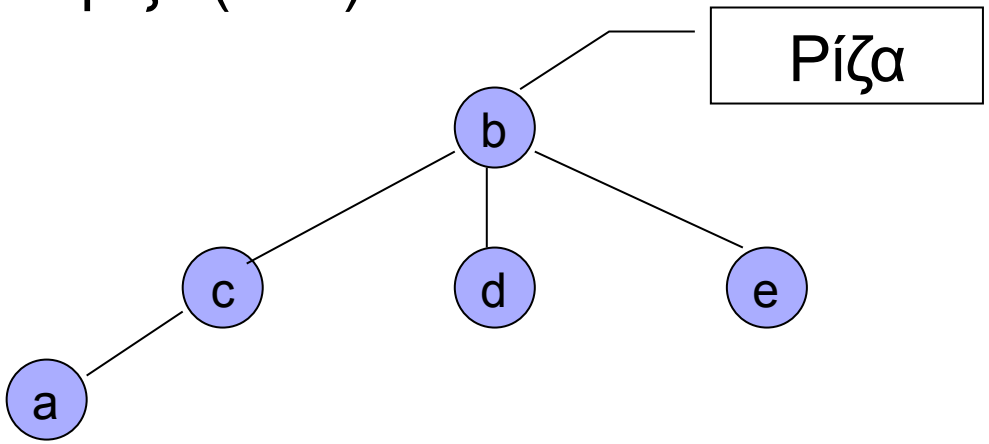


Δέντρα (Trees)

- Συνδεδεμένος Άκυκλος Γράφος
- Σε ένα συνδεδεμένο δέντρο ισχύει πάντα:
  - Υπάρχει ένα και μόνο μονοπάτι μεταξύ δύο κόμβων
- Ριζωμένα (rooted) δέντρα: δέντρα στα οποία αυθαίρετα επιλέγεται ένας κόμβος σαν ρίζα (root)

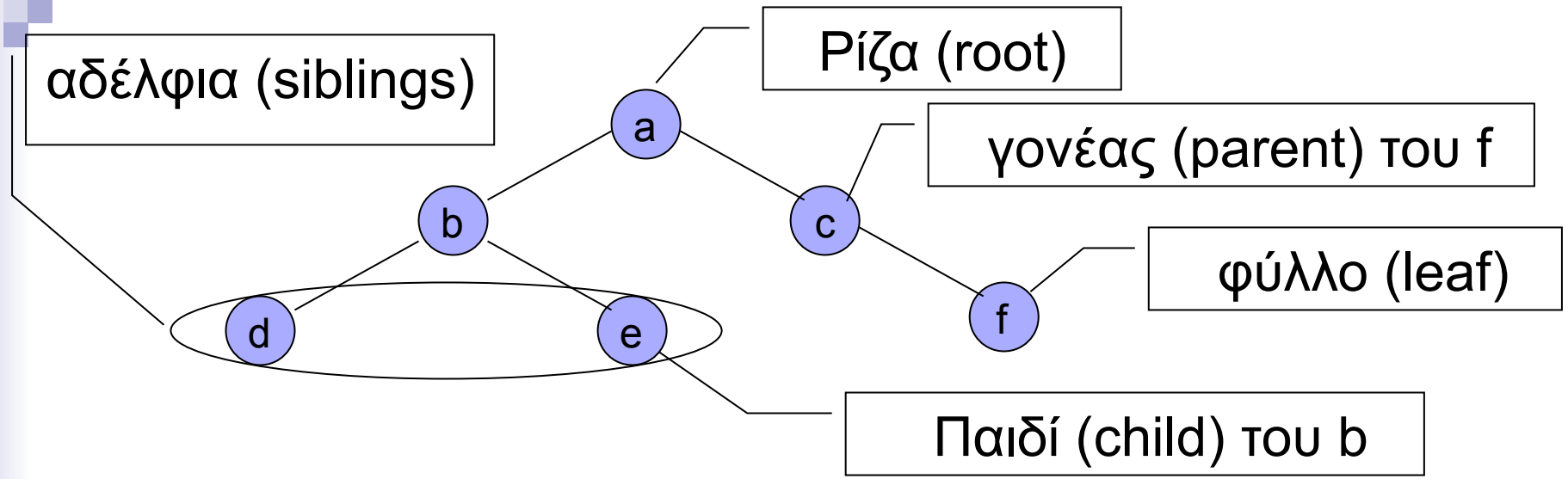


Ελεύθερο δέντρο



Ριζωμένο δέντρο

Δέντρα



- **Βαθμός** (degree) ενός κόμβου: αριθμός των παιδιών του.
- **Βαθμός του δέντρου**: μέγιστος βαθμός από τους κόμβους του δέντρου.
- **Βάθος** (depth) ενός κόμβου είναι η απόσταση του κόμβου από τη ρίζα
- **΄Υψος** του δέντρου είναι η απόσταση της ρίζας από το πιο απομακρυσμένο φύλλο.

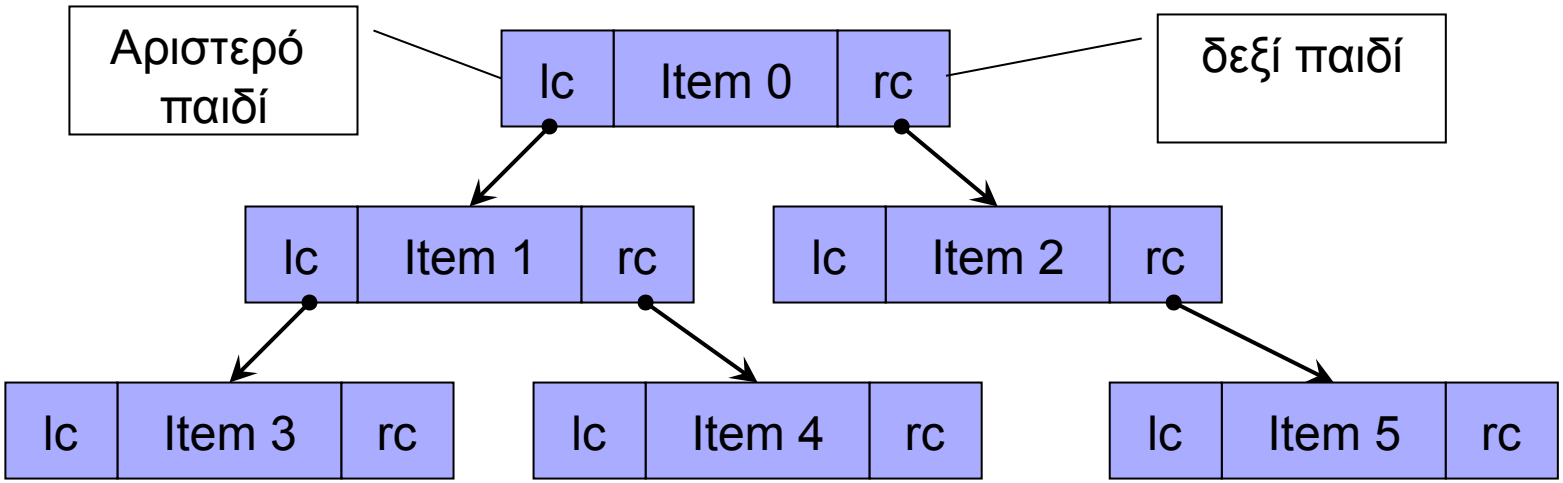
Απεικόνιση Δέντρων

- **Δυσδιάστατους Πίνακες (όπως οι γράφοι)**
  - **Όχι πολύ αποδοτική λύση αφού ένα δέντρο είναι ένας αραιός γράφος!**
- **Μονοδιάστατος Πίνακας**
  - Τα παιδιά κάθε κόμβου εμφανίζονται σε συγκεκριμένες θέσεις του πίνακα
  - Για αποδοτικότερη διαχείριση των δεδομένων, στον πίνακα εμφανίζονται δείκτες προς τα δεδομένα και όχι τα δεδομένα τα ίδια.
  - Παράδειγμα δυαδικού δέντρου
    - Τα παιδιά του κόμβου i είναι στις θέσεις $2i$, και $2i+1$.
    - Ο γονέας του κόμβου j βρίσκεται στη θέση $\lfloor j/2 \rfloor$

1	2	3	4	5	…	i	…	2*i*	2*i*+1	…	*n*

Απεικόνιση Δέντρων

- **Μη γραμμικές λίστες**
  - Κάθε κόμβος μπορεί να έχει ένα ή περισσότερα παιδιά
  - Παράδειγμα δυαδικού δέντρου



- Η ακόλουθη σχέση ισχύει για δυαδικά δέντρα με n κόμβους τους οποίους το ύψος είναι h

Ταξινομημένα Δέντρα

- **Δυαδικά ταξινομημένα δέντρα**
  - Το κλειδί του αριστερού παιδιού είναι μικρότερο από αυτό του γονιού
  - Το κλειδί του δεξιού παιδιού είναι μεγαλύτερο ή ίσο με αυτό του γονιού