Στοιχειώδεις Δομές Δεδομένων

Τύποι δεδομένων στη Java

- Ακέραιοι (int, long)
- Αριθμοί κινητής υποδιαστολής (float, double)
- Χαρακτήρες (char)
- Δυαδικοί (boolean)

Από τους παραπάνω μπορούμε να φτιάξουμε σύνθετους τύπους

Π.χ.

```
public class Point
{
        private float x;
        private float y;
}
```

Πίνακες

Πίνακας (array) A με στοιχεία τύπου T

- Σύνολο στοιχείων ιδίου τύπου

  Π.χ. πίνακας ακεραίων, πίνακας δεικτών σε ακέραιους, …

- Συνεχόμενες θέσεις στη μνήμη

  Το i-οστο στοιχείο `A[i-1]` βρίσκεται στη θέση `X + (i-1)*L`, όπου `X` η θέση του πρώτου στοιχείου `A[0]` και `L` το μέγεθος του κάθε στοιχείου

- Αναφορά με χρήση ακέραιου δείκτη

  Π.χ. ανάθεση τιμής στο 3ο στοιχείο του πίνακα `A[2]=t;`

Πίνακες

Πίνακες στη Java

Στατική κατανομή μνήμης

```
int[] A = { 1, 1, 2, 0, 5, 8 };
```

Δυναμική κατανομή μνήμης

```
int[] A;
A = new int [N];
```

Η δέσμευση μνήμης γίνεται κατά το χρόνο εκτέλεσης, όταν η τιμή του N είναι γνωστή

ή πιο σύντομα

```
int[] A = new int [N];
```

Πίνακες

Πίνακες στη Java

Στατική κατανομή μνήμης

```
int[] A = { 1, 1, 2, 0, 5, 8 };
```

Δυναμική κατανομή μνήμης

```
int[] A;
A = new int [N];
```

Η δέσμευση μνήμης γίνεται κατά το χρόνο εκτέλεσης, όταν η τιμή του N είναι γνωστή

ή πιο σύντομα

```
int[] A = new int [N];
```

Τι γίνεται αν δε γνωρίζουμε ποια τιμή του N είναι κατάλληλη για το πρόγραμμά μας;

Δυναμικοί πίνακες

Δυναμικοί πίνακες με εισαγωγές και διαγραφές

Συντελεστής πληρότητας πίνακα A : $\alpha = \frac{n}{|A|}$

όπου $n =$ πλήθος αποθηκευμένων στοιχείων στον A

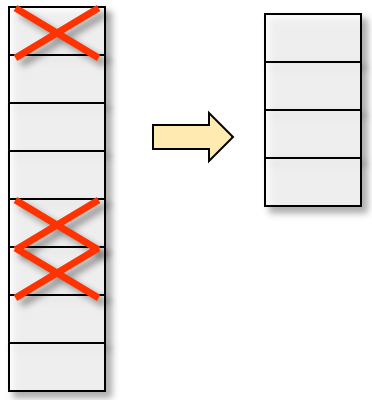
$|A| =$ μέγεθος του A

Επιθυμητές ιδιότητες :

α) περιορισμένη σπατάλη χώρου : $\alpha \geq$ σταθερά

β) μικρό συνολικό κόστος για οποιαδήποτε ακολουθία πράξεων

Για να πετύχουμε την ιδιότητα (α) πρέπει να αντιγράφουμε τα στοιχεία σε μικρότερο πίνακα μετά από αρκετές διαγραφές

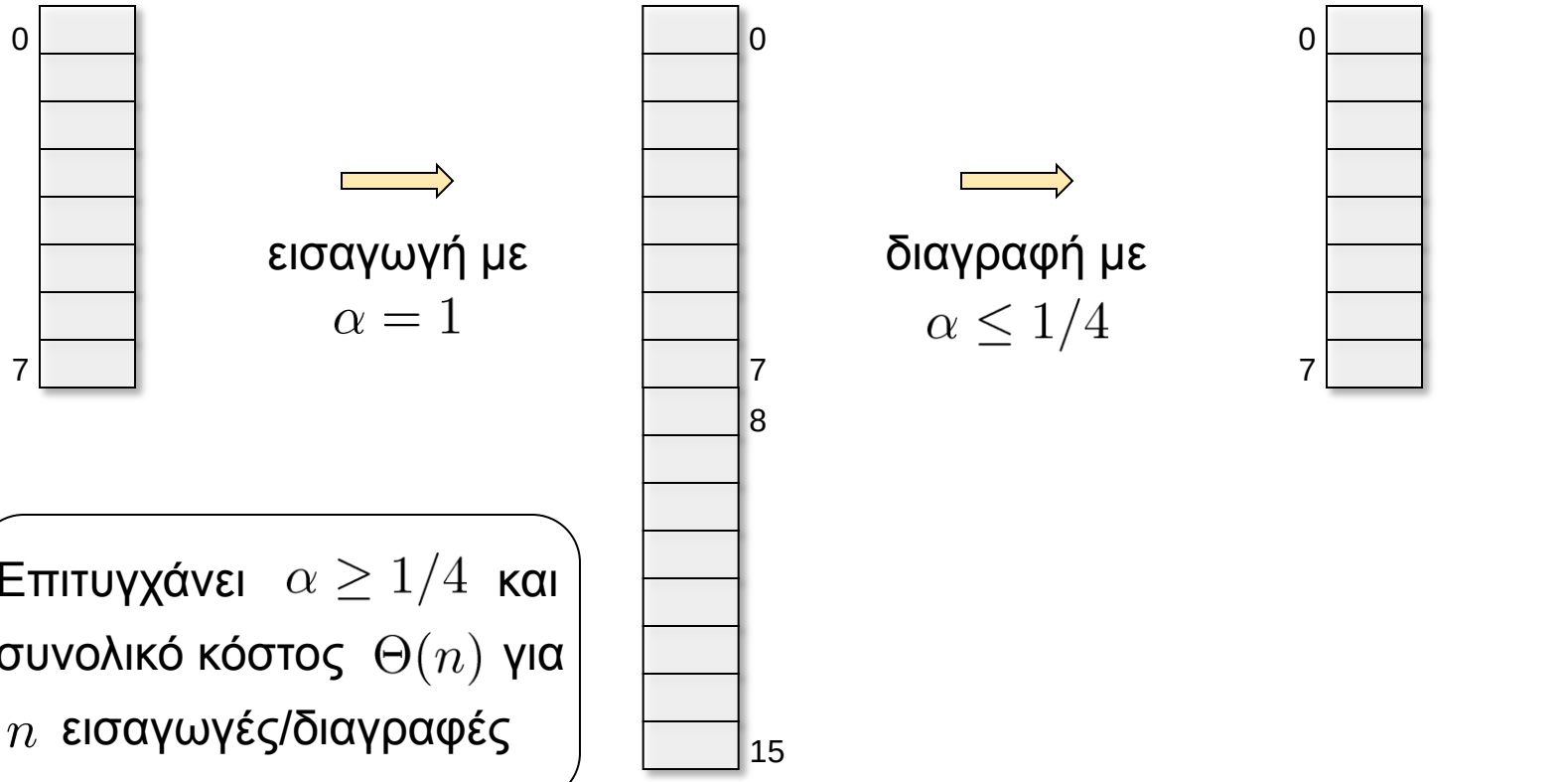


Δυναμικοί πίνακες

Δυναμικοί πίνακες με εισαγωγές και διαγραφές

διπλασιασμός

υποδιπλασιασμός



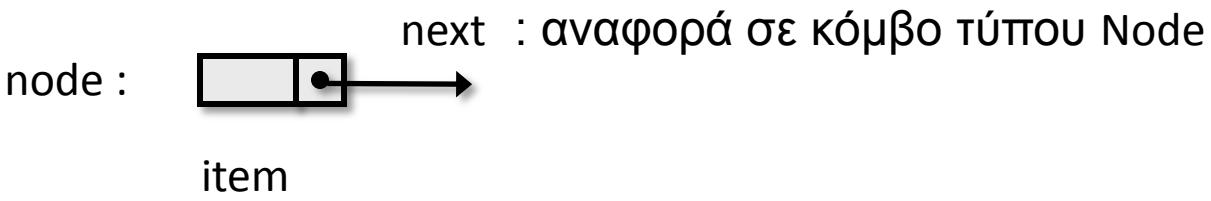
Επιτυγχάνει $\alpha \geq 1/4$ και συνολικό κόστος $\Theta(n)$ για n εισαγωγές/διαγραφές

Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

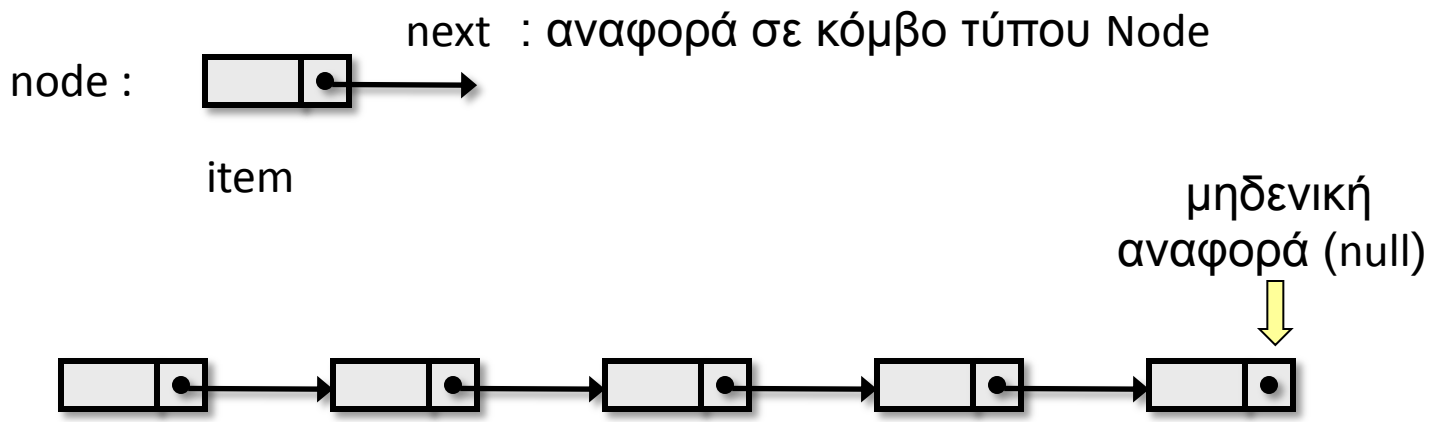


Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

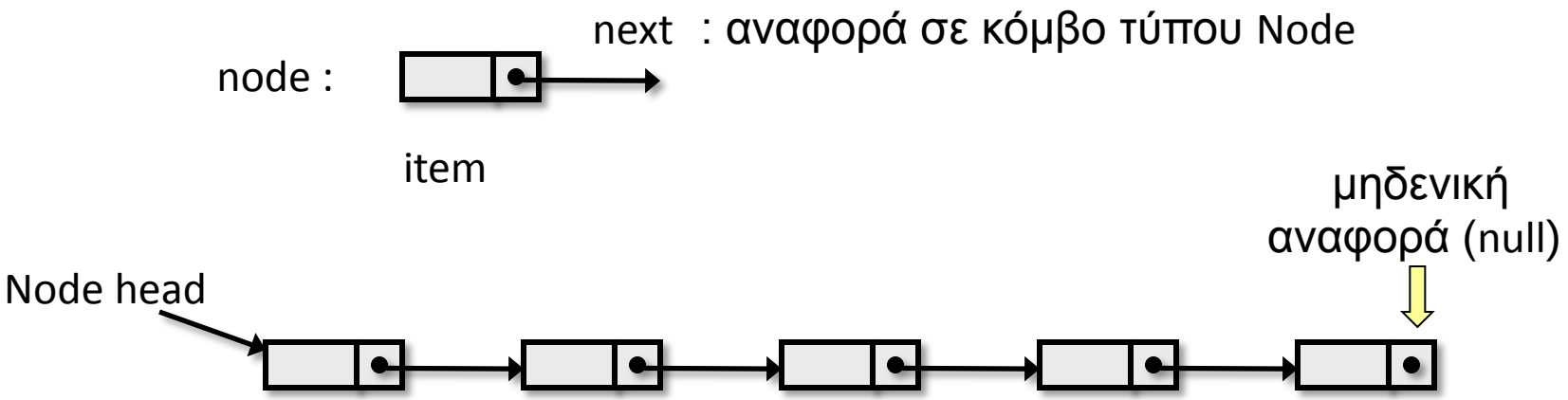


Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```



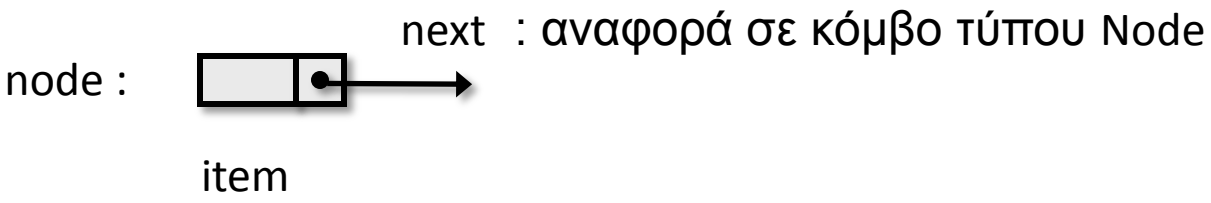
Για να προσπελάσουμε τα στοιχεία της λίστας χρειαζόμαστε μια αναφορά στον πρώτο κόμβο της λίστας. Η αναφορά αποθηκεύεται στη μεταβλητή head (τύπου Node).

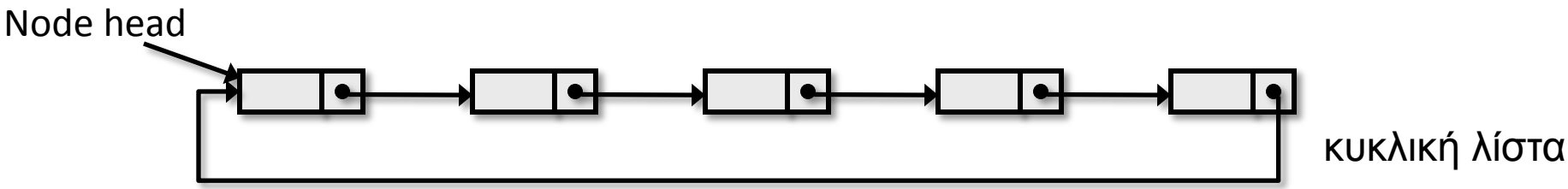
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```





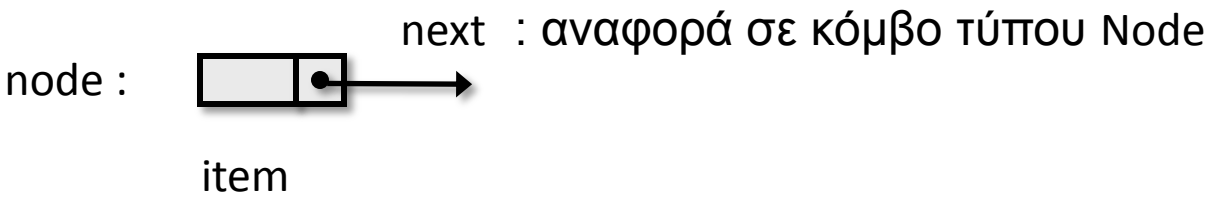
Σε ορισμένες περιπτώσεις που θέλουμε να επεξεργαστούμε τα στοιχεία της λίστας πολλαπλές φορές είναι βολικό να κάνουμε τη λίστα κυκλική

Συνδεδεμένες Λίστες

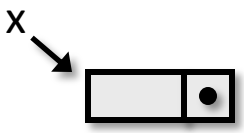
Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```



δημιουργία νέου κόμβου Node x = new Node();



x.item = πεδίο item του κόμβου με αναφορά x

x.next = πεδίο next του κόμβου με αναφορά x

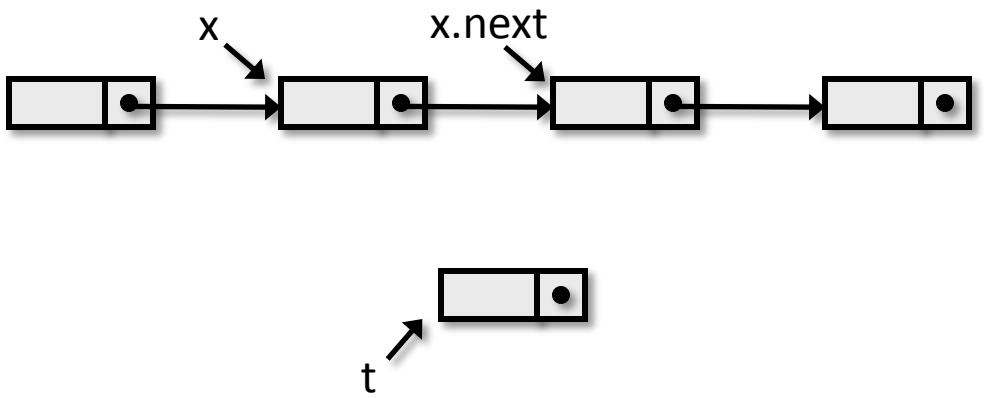
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Εισαγωγή του κόμβου t μετά τον x



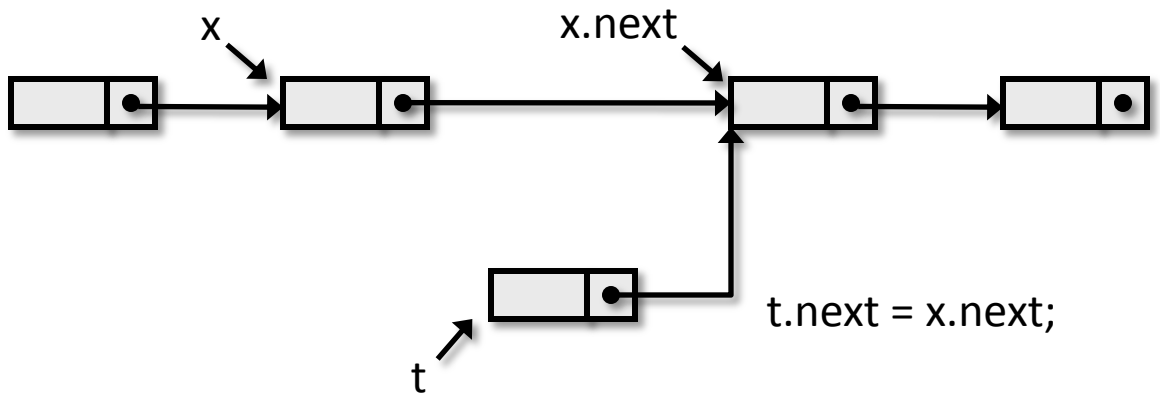
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Εισαγωγή του κόμβου t μετά τον x



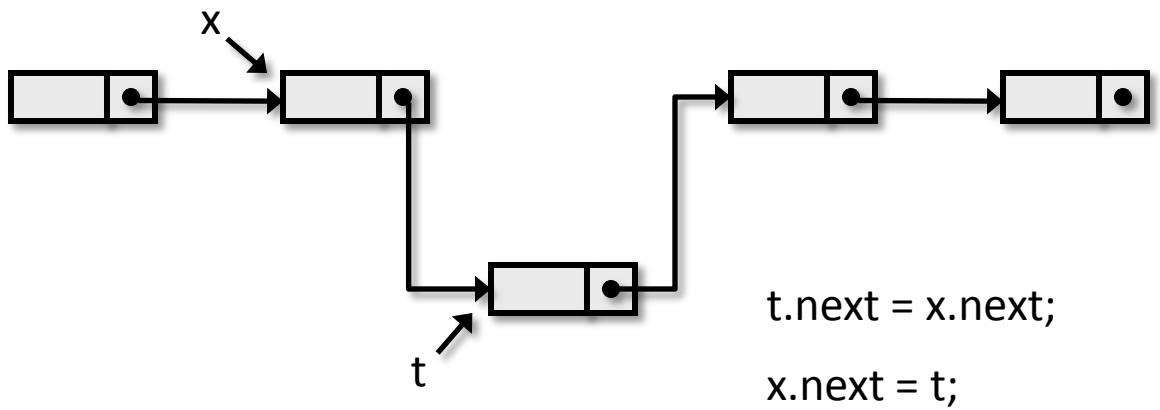
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Εισαγωγή του κόμβου t μετά τον x



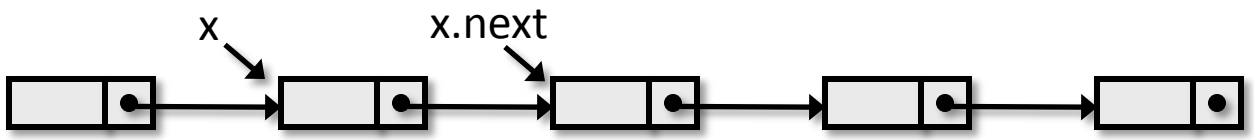
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Διαγραφή του κόμβου μετά τον x



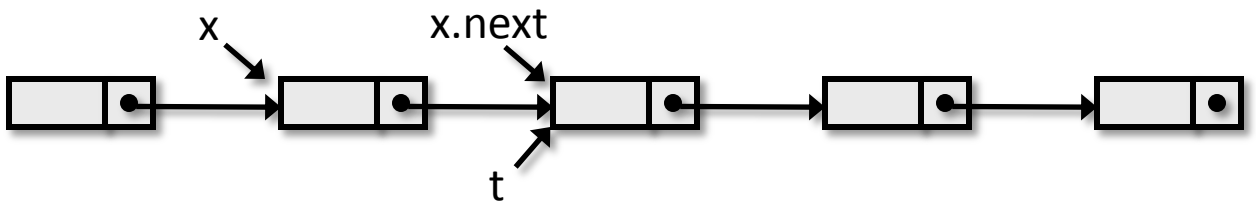
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Διαγραφή του κόμβου μετά τον x



t = x.next;

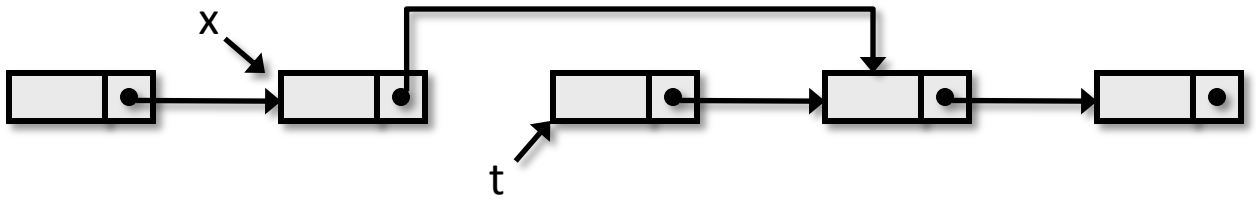
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Διαγραφή του κόμβου μετά τον x



t = x.next; x.next = t.next;

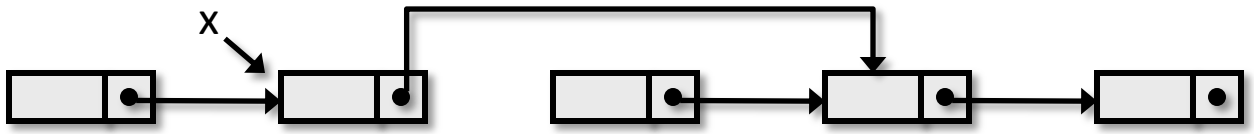
Συνδεδεμένες Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```

Διαγραφή του κόμβου μετά τον x



ή πιο απλά `x.next = x.next.next;`

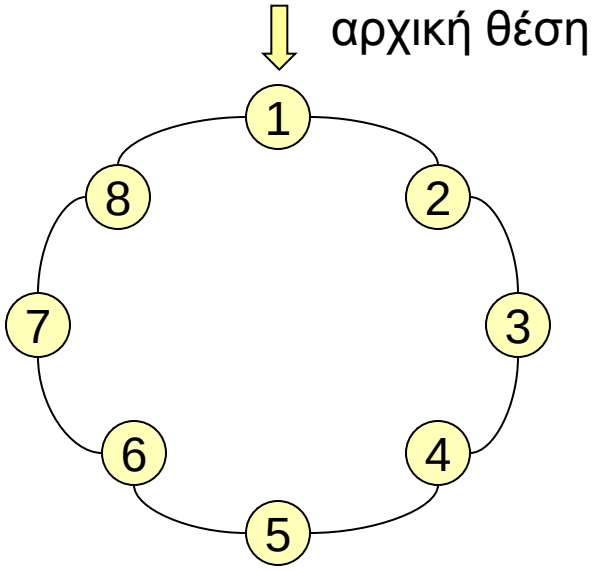
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



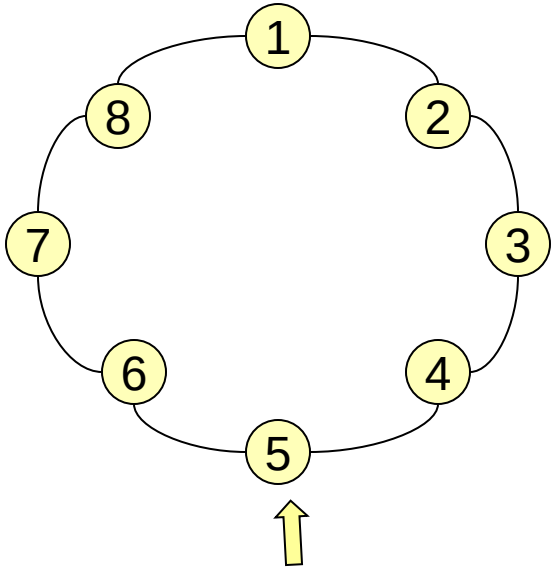
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



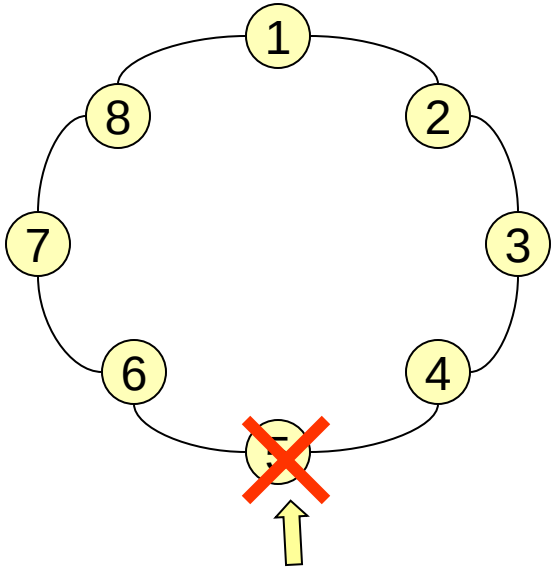
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



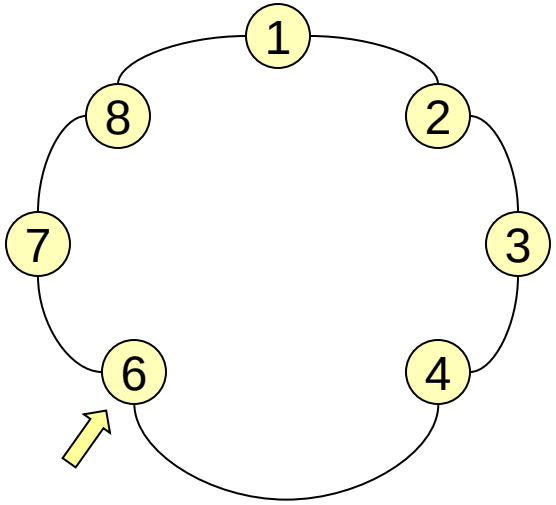
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



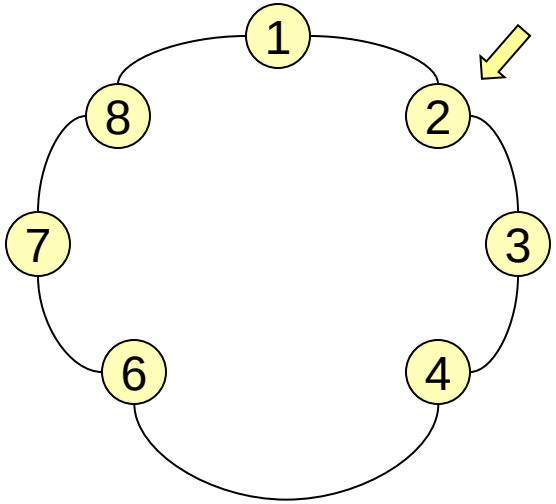
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



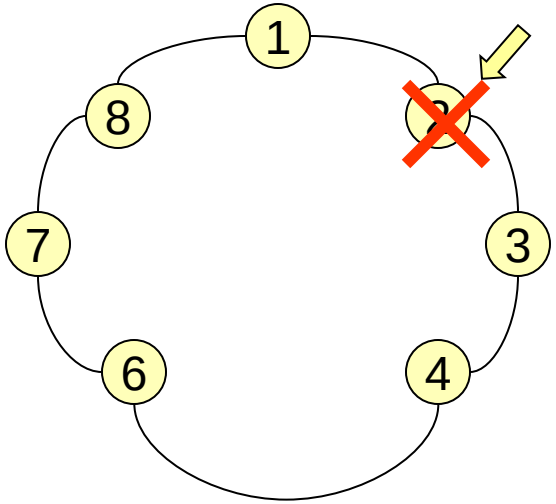
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



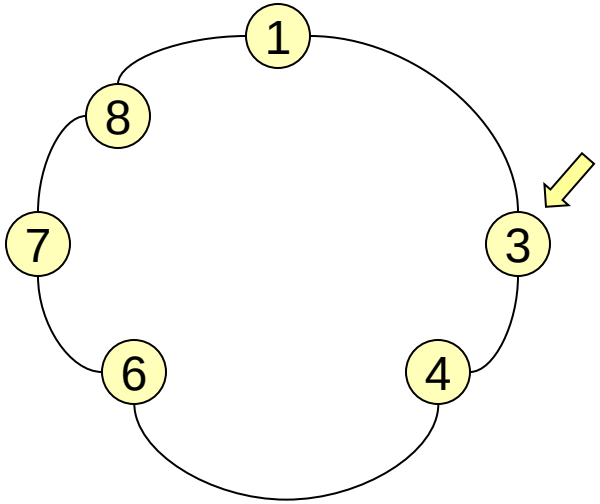
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



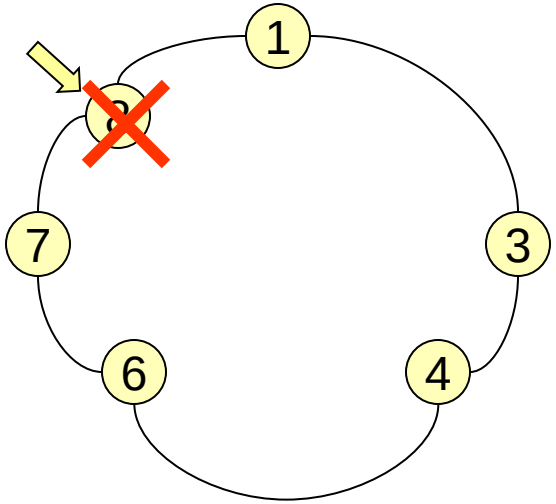
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



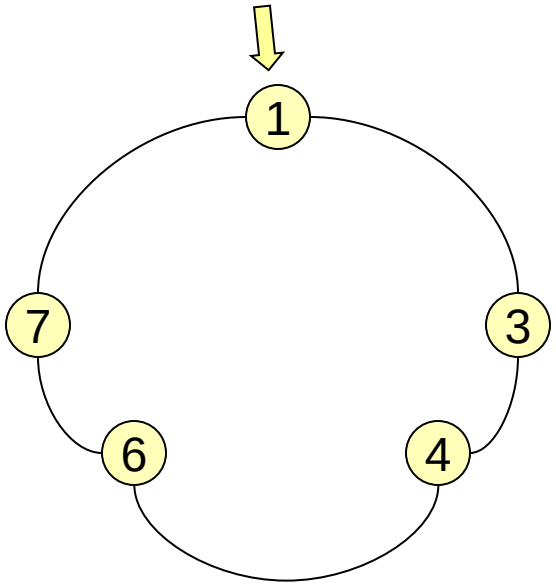
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



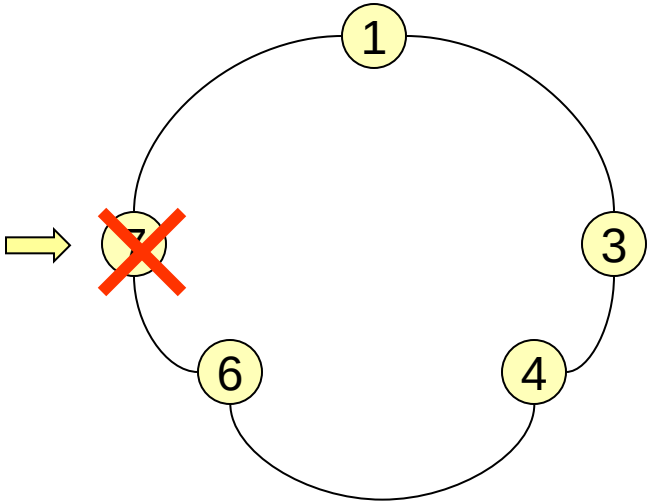
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



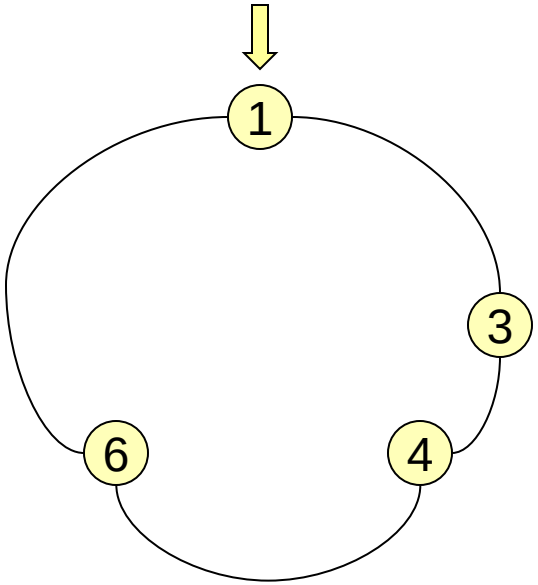
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



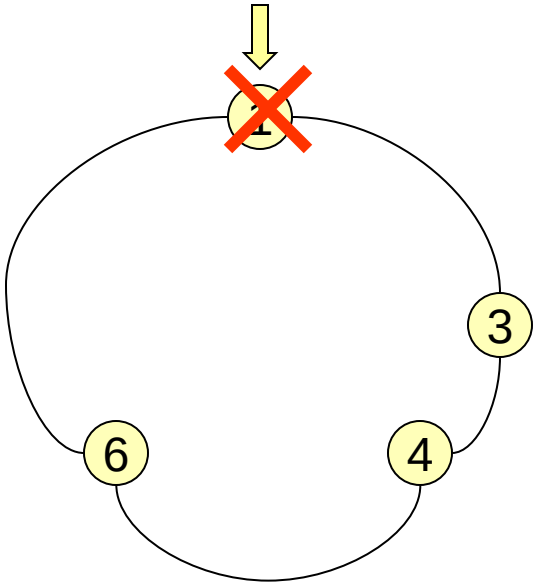
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



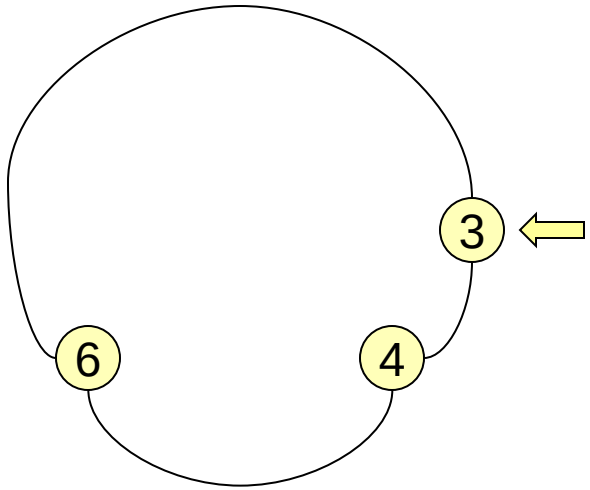
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



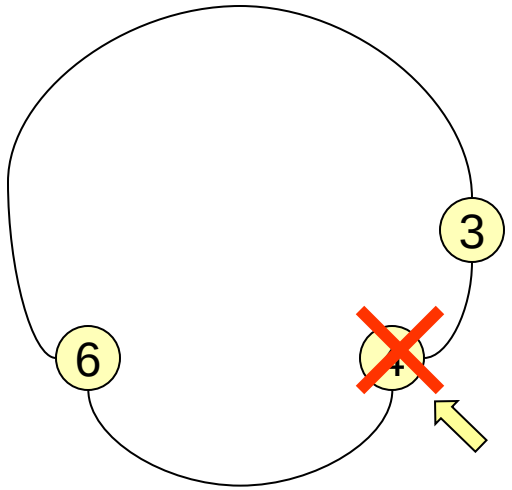
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



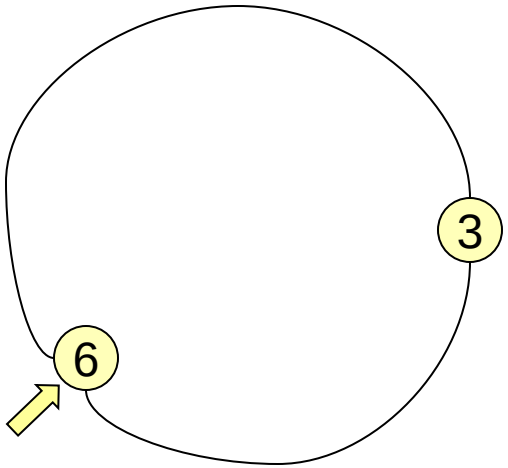
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



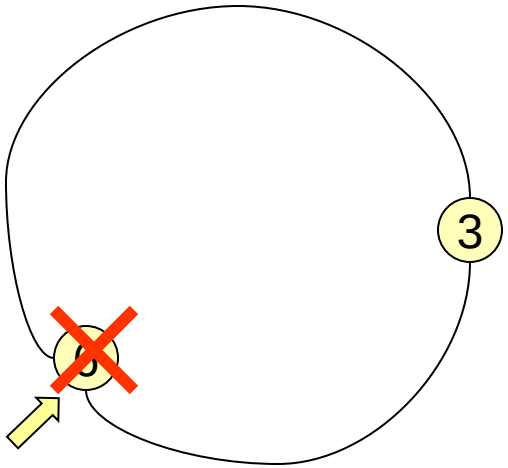
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



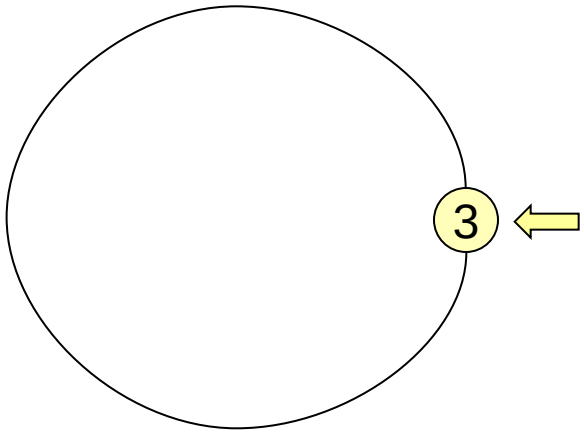
Πρόβλημα του Josephus

Ν άτομα στέκονται σε ένα κύκλο και περιμένουν να εκτελεστούν.

Σε κάθε βήμα εκτελείται το Μ-οστό άτομο όπως διατρέχουμε τον κύκλο.

Στον τελευταίο που θα μείνει δίνεται χάρη.

Π.χ. Ν=8, Μ=5



Πρόβλημα του Josephus

```
class Josephus {
        static class Node {
                int item; Node next;
                Node(int v) { item = v; }
        }

        public static main(String[] args) {
                int N = Integer.parseInt(args[0]);
                int M = Integer.parseInt(args[1]);
                Node t = new Node(1);
                Node x = t;
                for (int i=2; i<=N; i++)   x = (x.next = new Node(i));
                x.next = t;
                while (x != x.next) {
                        for (int i=1; i<M; i++) x=x.next;
                        x.next=x.next.next;
                }
                System.out.println("Winner is " + x.item);
        }
}
```

Υλοποίηση Κυκλικής Λίστας

Διασύνδεση κυκλικής λίστας για αντικείμενα τύπου int

Node next(Node x) :	Επιστρέφει τον κόμβο που βρίσκεται μετά τον x.
int item(Node x) :	Επιστρέφει το αντικείμενο του κόμβου x.
Node insert(Node x, int v) :	Εισαγάγει το αντικείμενο v σε νέο κόμβο μετά τον κόμβο x. Αν x=null τότε δημιουργεί μια νέα λίστα.
void remove(Node x) :	Διαγράφει τον κόμβο μετά τον κόμβο x. Αν ο x είναι ο μοναδικός κόμβος της λίστας τότε η μέθοδος δεν έχει καμία επίδραση.

Υλοποίηση Κυκλικής Λίστας

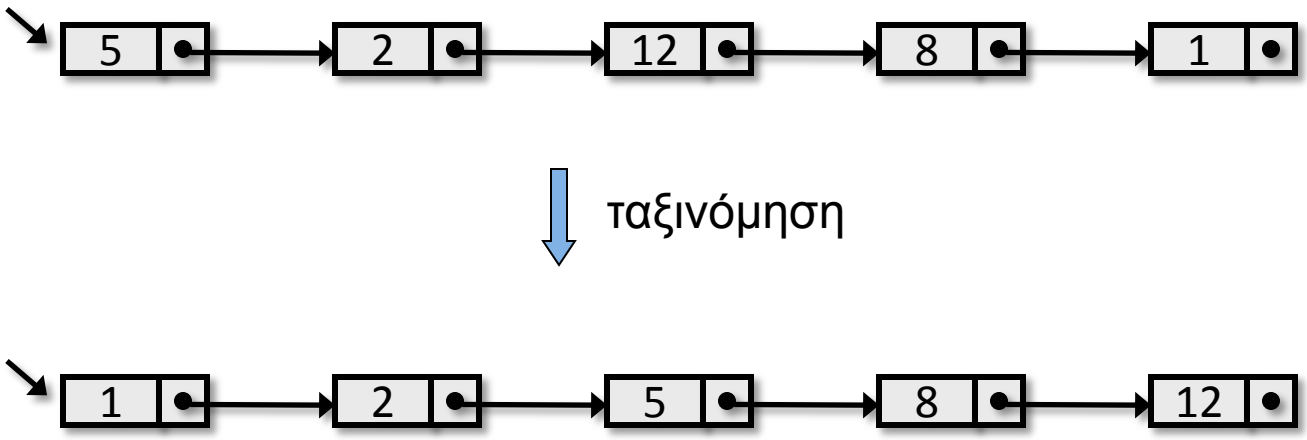
```
class CircularList {
        static class Node {
                int item; Node next;
                Node(int v) { item = v; }
        }

        Node next(Node x) { return x.next; }

        int item(Node x) { return x.item;}

        Node insert(Node x, int v) {
                Node t = new Node(v);
                if (x == null) t.next = t;
                else { t.next = x.next; x.next = t; }
                return t;
        }

        void remove(Node x) { x.next = x.next.next; }
```

Λύση στο πρόβλημα του Josephus με πρόγραμμα-πελάτη

```
class Josephus {
        public static main(String[] args) {
                int N = Integer.parseInt(args[0]);
                int M = Integer.parseInt(args[1]);

                CircularList L = new CircularList();
                CircularList.Node x = null;

                for (int i=1; i<=N; i++)   x = L.insert(x,i);

                while (x != L.next(x)) {
                        for (int i=1; i<M; i++) x=L.next(x);
                        L.remove(x);
                }
                System.out.println("Winner is " + L.item(x));
        }
}
```

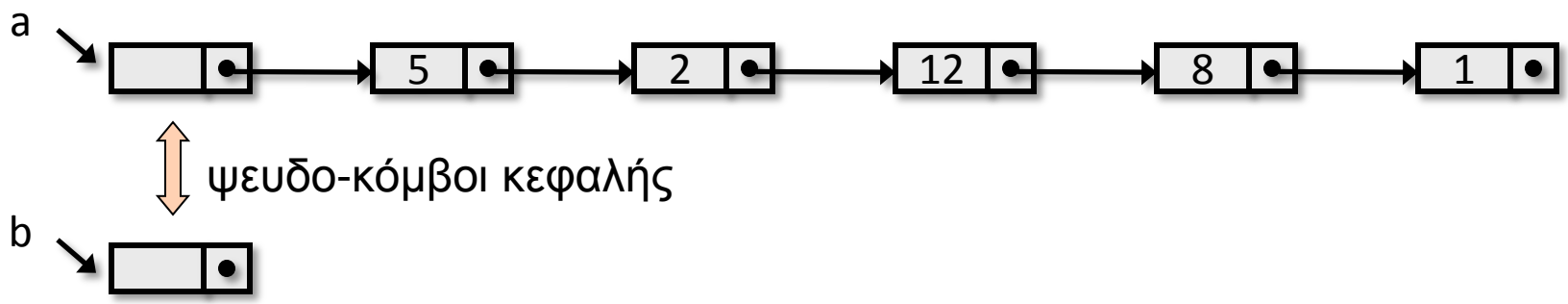
Ταξινόμηση Συνδεδεμένης Λίστας

Μη διατεταγμένη λίστα



Ταξινόμηση Συνδεδεμένης Λίστας

Ταξινόμηση με εισαγωγή



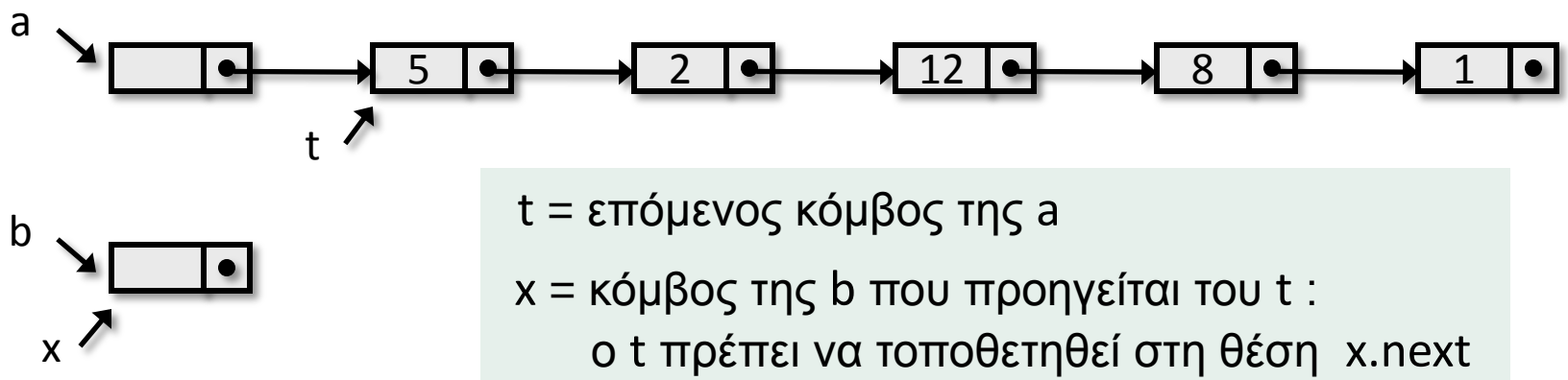
Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

Ταξινόμηση με εισαγωγή



t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

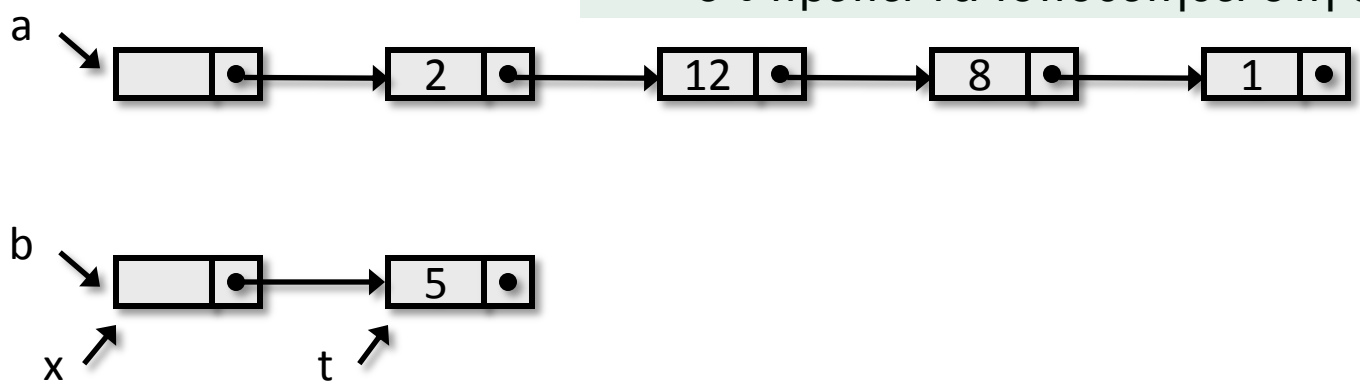
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

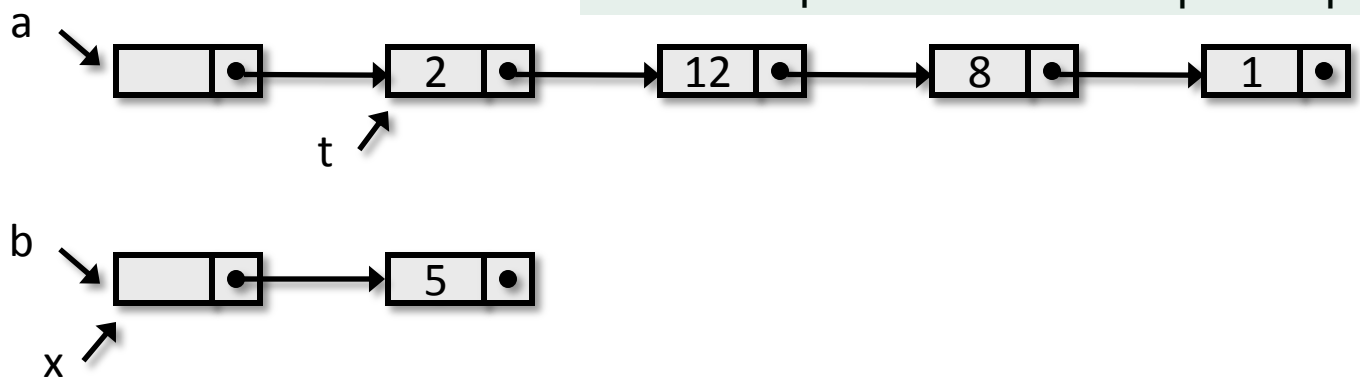
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

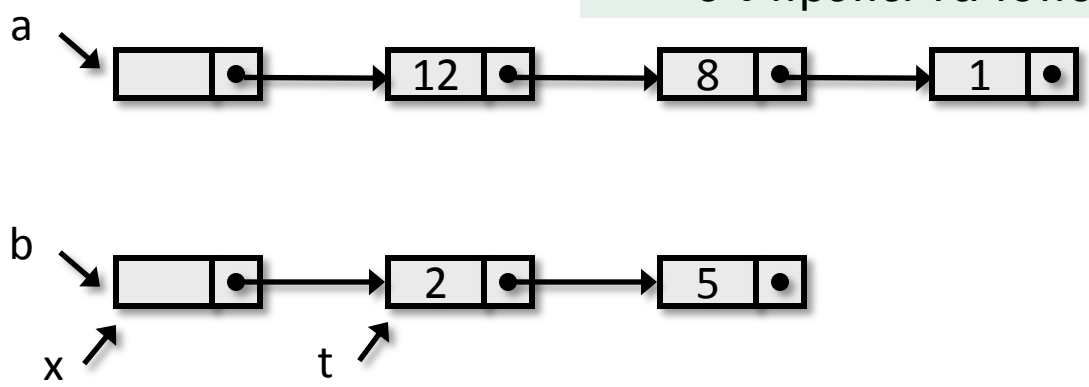
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

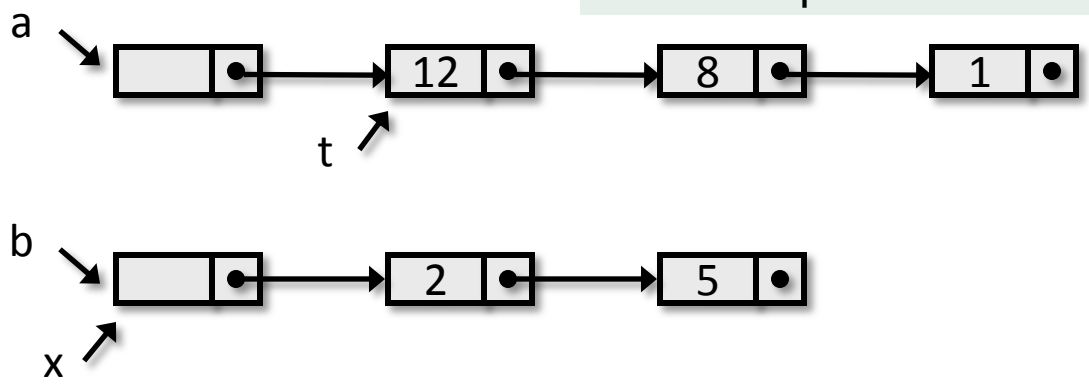
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

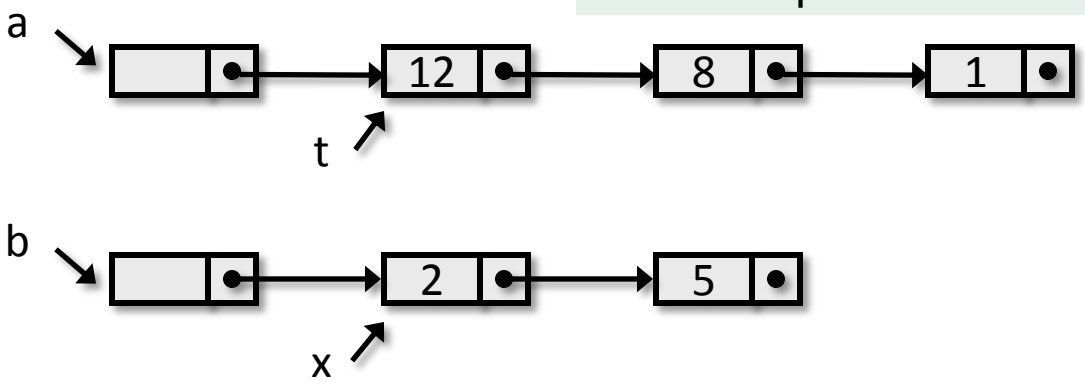
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

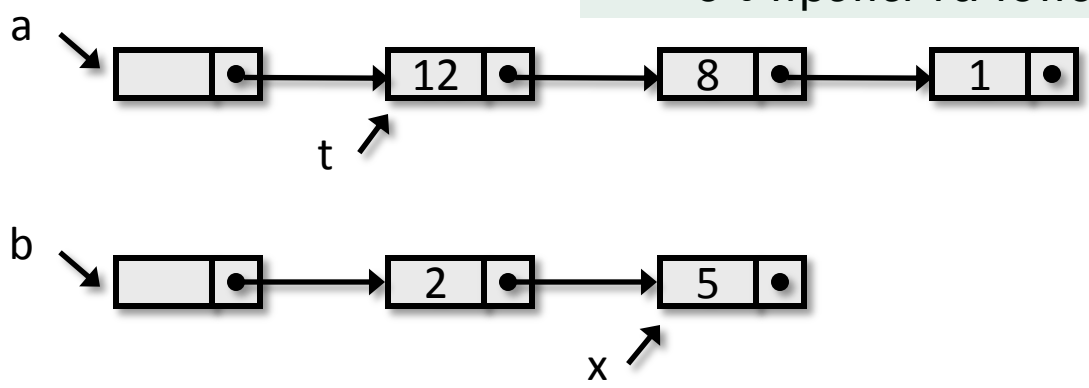
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

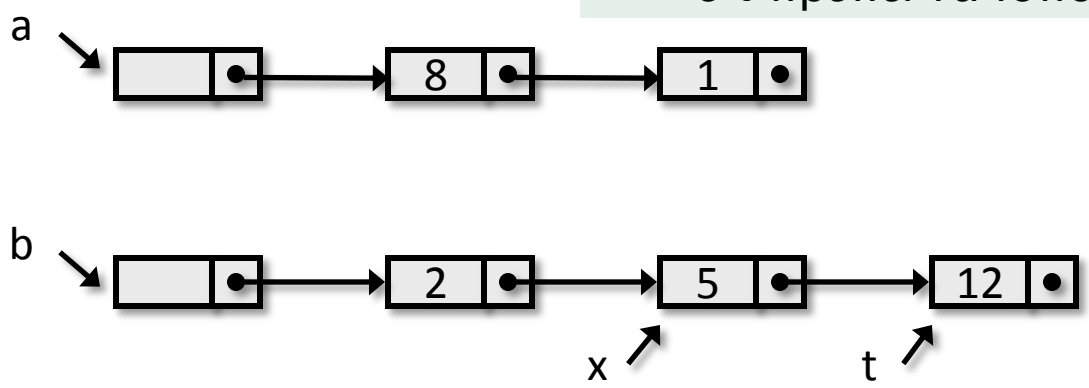
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

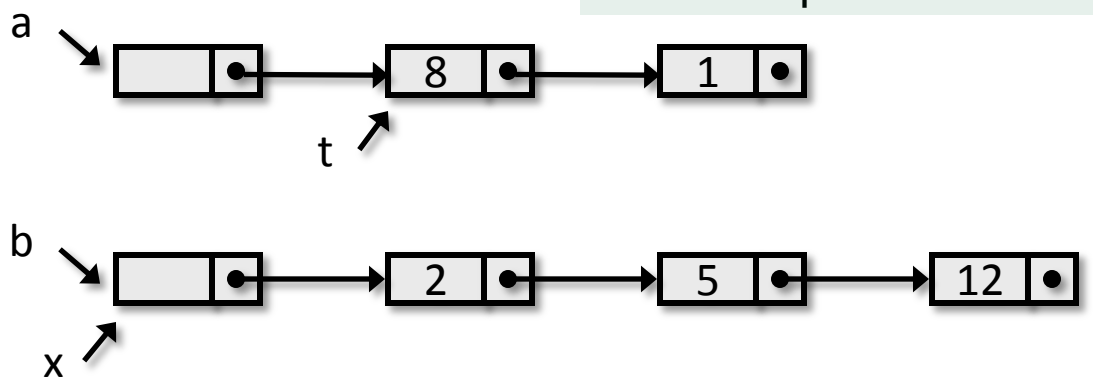
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

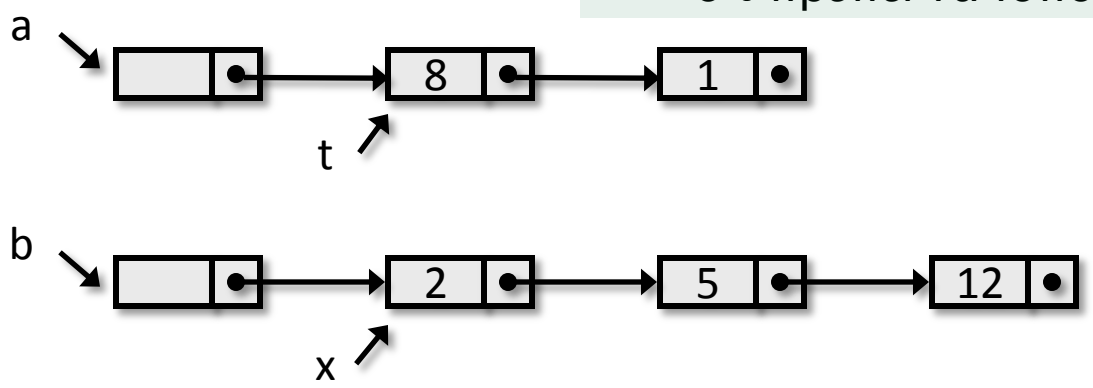
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

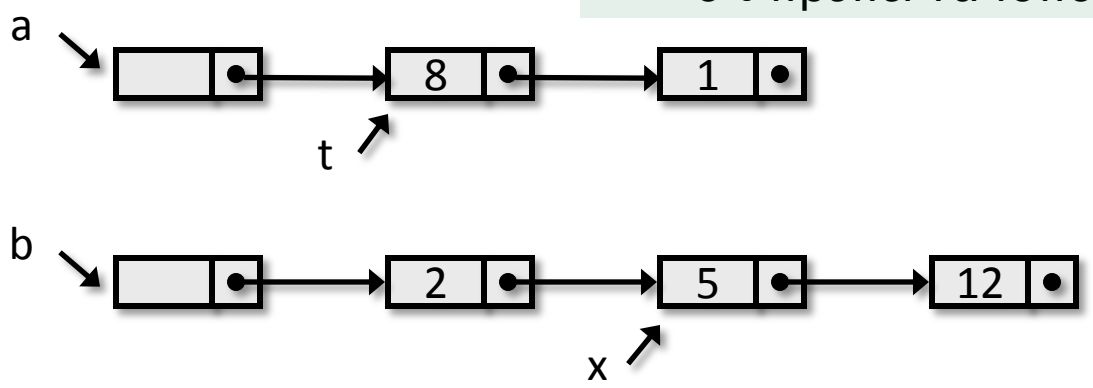
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a

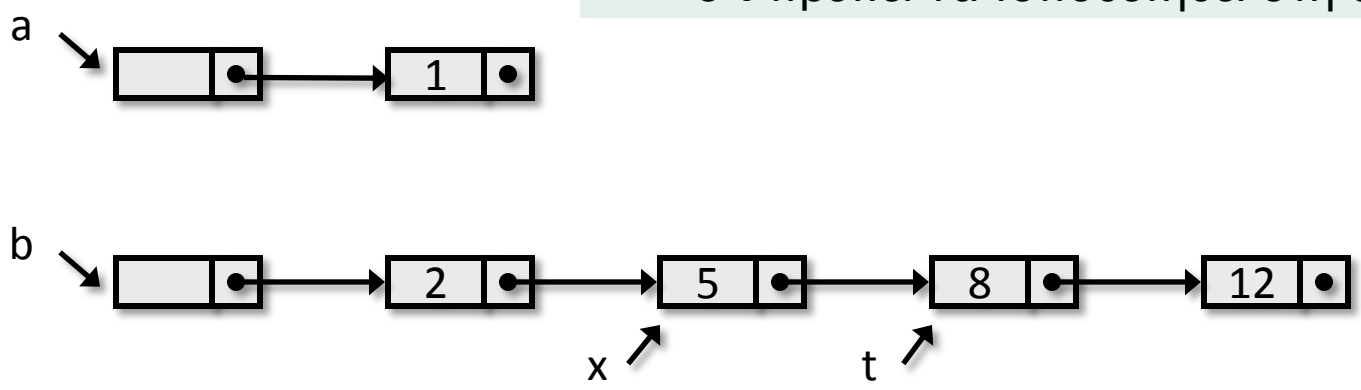
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

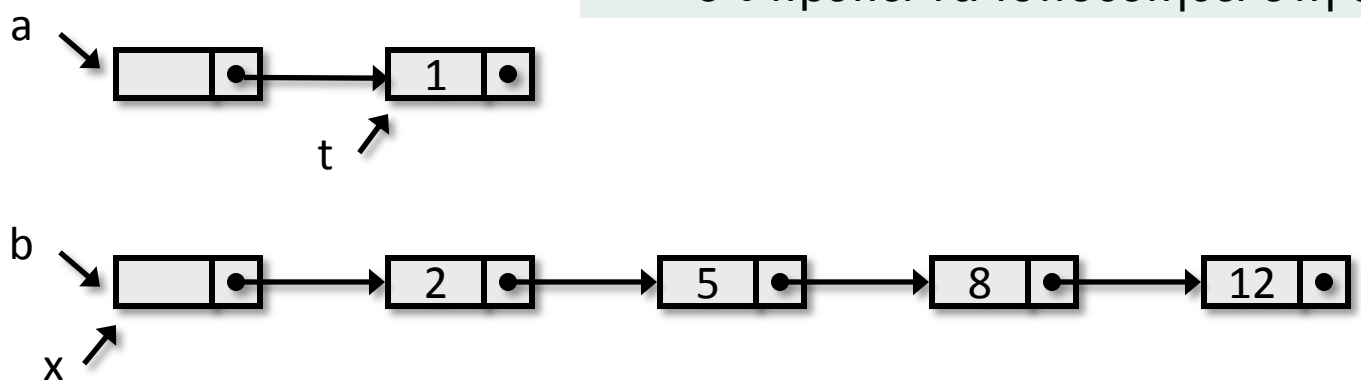
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

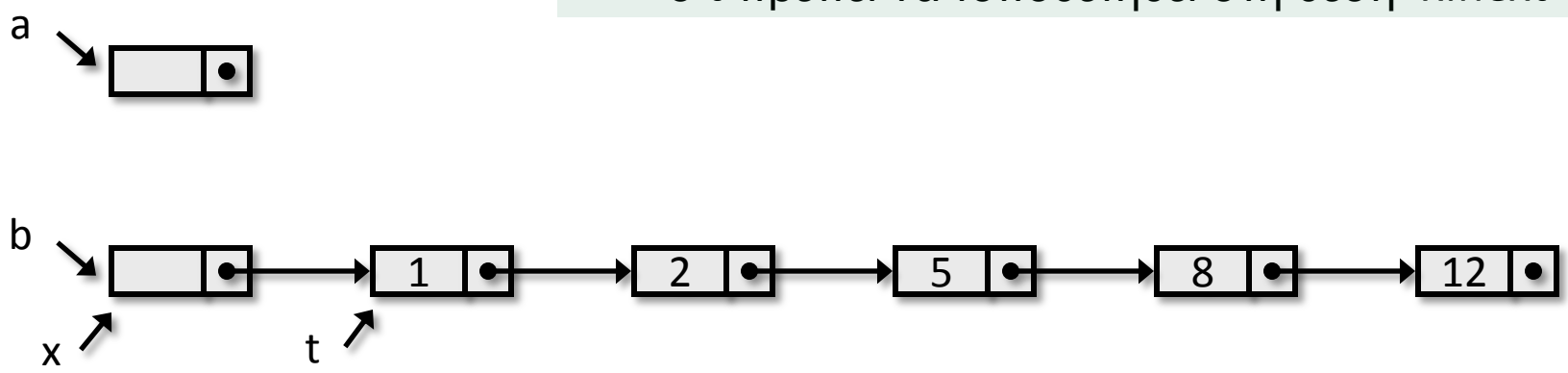
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

Ταξινόμηση με εισαγωγή



Έστω a η αρχική λίστα.
Διατηρούμε μία ακόμα λίστα b που θα είναι διατεταγμένη.
Για ευκολία τοποθετούμε ένα ψευδο-κόμβο κεφαλής στην αρχή κάθε λίστας.

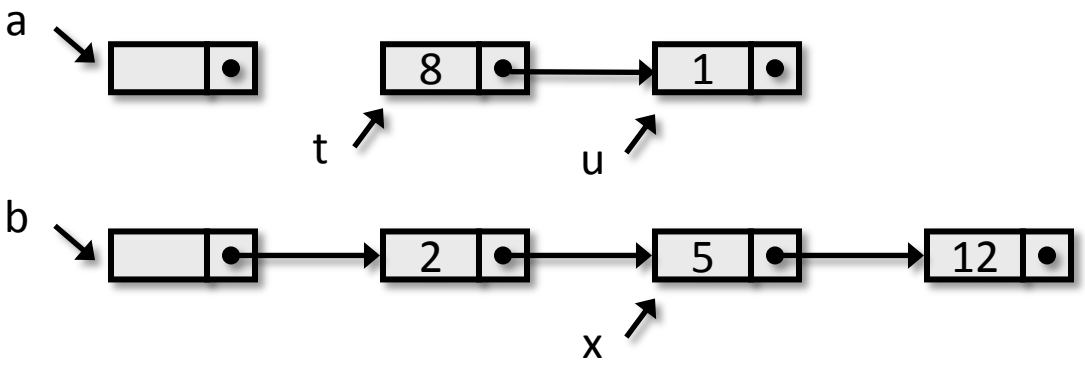
Αρχικά η b περιλαμβάνει μόνο τον ψευδο-κόμβο κεφαλής.
Μέχρι να αδειάσει η a
διαγράφουμε το πρώτο στοιχείο της a
και το εισάγουμε στη σωστή του θέση στη b.

Ταξινόμηση Συνδεδεμένης Λίστας

Ταξινόμηση με εισαγωγή

t = επόμενος κόμβος της a

x = κόμβος της b που προηγείται του t :
o t πρέπει να τοποθετηθεί στη θέση x.next

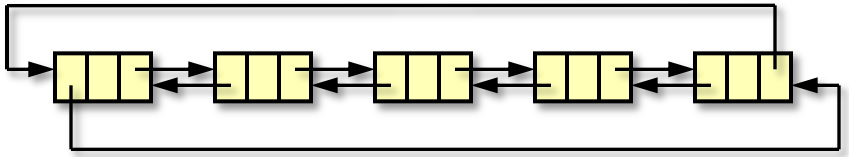



```
class ListSort {
        static class Node { int item; Node next;
                Node(int v, Node t) { item = v; next = t; } }

        static Node sort(Node a) { Node t, u, x, b = new Node(0,null);
                while (a.next  !=  null) {
                        t = a.next; u = t.next; a.next = u;
                        for (x = b; x.next != null; x = x.next)
                                if (x.next.item > t.item) break;
                        t.next = x.next; x.next = t;
                }
                return b;  }
}
```

Διπλά Συνδεδεμένη Λίστα

```
private class Node
{
        Item item; Node prev; Node next;
}
```

διπλά συνδεδεμένη λίστα



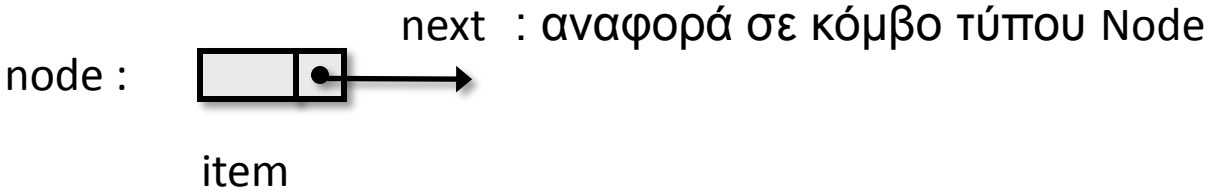
διπλά συνδεδεμένη κυκλική λίστα

Κατανομή Μνήμης για Λίστες

Συνδεδεμένη λίστα: Αποθηκεύει ένα σύνολο στοιχείων σε κόμβους.

Κάθε κόμβος περιλαμβάνει ένα σύνδεσμο προς τον επόμενο κόμβο.

```
private class Node
{Item item; Node next;}
```



δημιουργία νέου κόμβου `Node x = new Node();`

Με αυτόν τον τρόπο η διαχείριση της μνήμης για τους κόμβους της λίστας γίνεται από το σύστημα.

Κατανομή Μνήμης για Λίστες

Σε ορισμένες περιπτώσεις μπορούμε να αναλάβουμε μόνοι μας τη διαχείριση της μνήμης που χρειάζεται η λίστα. Αυτό μπορεί να γίνει όταν :

(α) κάθε αντικείμενο χρειάζεται τον ίδιο χώρο μνήμης, και
(β) γνωρίζουμε τον μέγιστο αριθμό Ν των αντικειμένων που θα αποθηκεύσει.

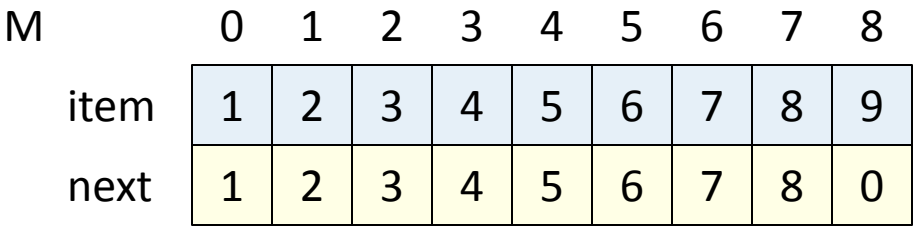
Αν δεν έχουμε ένα καλό άνω φράγμα για την τιμή του Ν, αλλά μας ενδιαφέρει μόνο ο συνολικός χρόνος εκτέλεσης του προγράμματος, μπορούμε να χρησιμοποιήσουμε την τεχνική του διπλασιασμού/υποδιπλασιασμού του Ν.

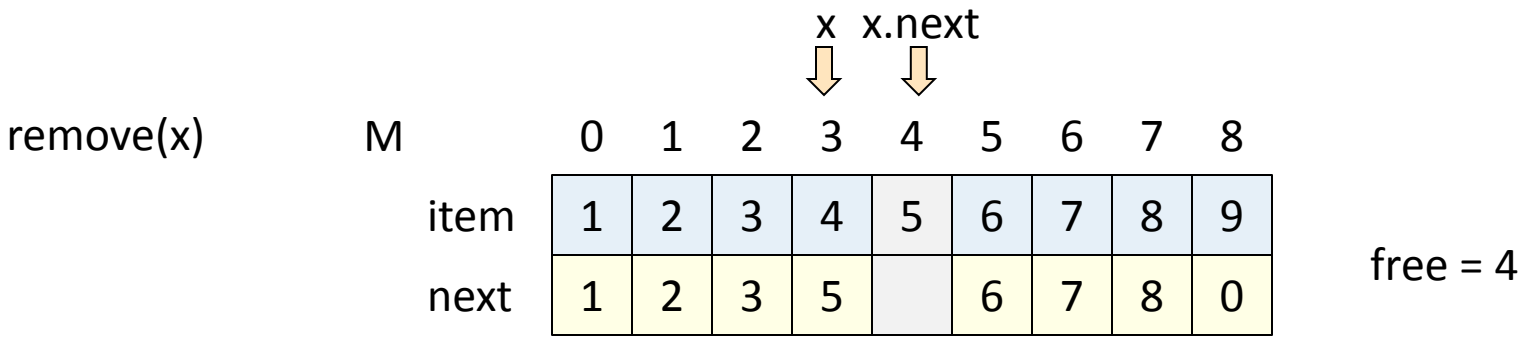
Όταν ισχύουν τα παραπάνω τότε μπορούμε να δεσμεύσουμε εξαρχής τον απαιτούμενο χώρο για την λίστα σε ένα πίνακα. Με τον τρόπο αυτό μπορεί το πρόγραμμά μας να έχει βελτιωμένη απόδοση.

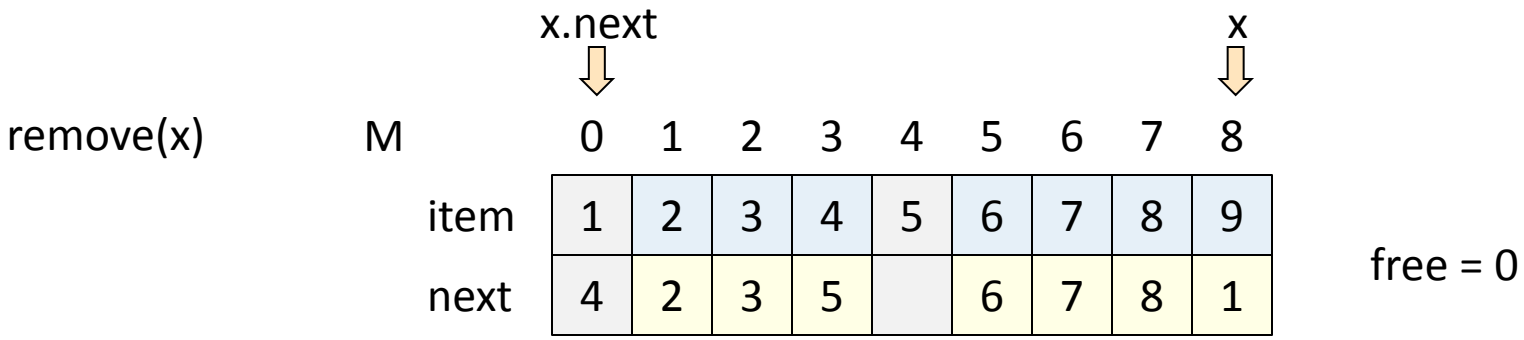
Κατανομή Μνήμης για Λίστες

```
class CircularList {
        static class Node { int item; int next; }
        static Node M[];        // πίνακας που αποθηκεύει τα στοιχεία της λίστας
        static int free;        // πρώτη ελεύθερη θέση
        static int N = 10000;   // μέγιστος αριθμός στοιχείων της λίστας

        CircularList() {  M = new Node[N+1];
                for (int i = 0; i < N; i++)   {   M[i] = new Node(); M[i].next = i+1;  }
                M[N] = new Node(); M[N].next = 0; free = 0;  }

        Node next(Node x)  { return M[x.next]; }

        int item(Node x)  { return x.item; }

        Node insert  (Node x, int v )  {  int i = free;  free = M[free].next;
                M[i].item = v;
                if (x == null) M[i].next = i;  else {  M[i].next = x.next;  x.next = i; }
                return M[i];  }

        void remove(Node x) {  int i = x.next;  x.next = M[i].next;
                M[i].next = free;  free = i;  }
}
```

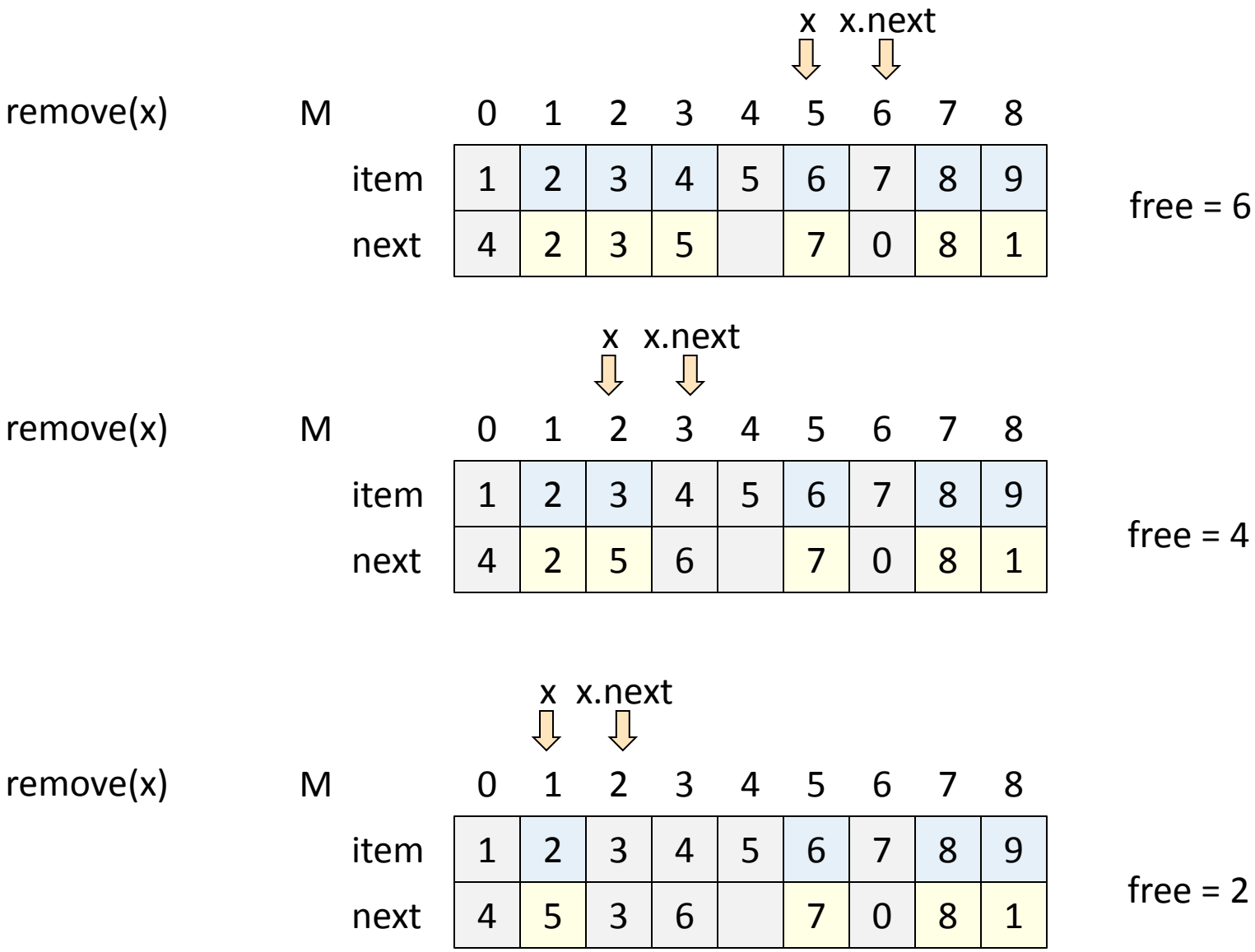
Κατανομή Μνήμης για Λίστες







Κατανομή Μνήμης για Λίστες



Σύνθετες Δομές Δεδομένων

Πίνακες (μήτρες) στη γραμμική άλγεβρα

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \vdots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Υλοποίηση στη Java

```
double[][] A = new double[m][n];
```

Γενικά για πίνακα με d διαστάσεις

```
double[][]…[] A = new double[k1][k2]…[kd];
```

όπου k1,...,kd θετικοί ακέραιοι

Σύνθετες Δομές Δεδομένων

Πολλαπλασιασμός πινάκων $C[1:p, 1:r] = A[1:p, 1:q] \times B[1:q, 1:r]$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1q} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2q} \\ & & & \vdots & & \\ a_{k1} & a_{k2} & \cdots & a_{kj} & \cdots & a_{kq} \\ & & & \vdots & & \\ a_{p1} & a_{p2} & \cdots & a_{pj} & \cdots & a_{pq} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2r} \\ & & & \vdots & & \\ b_{k1} & b_{k2} & \cdots & b_{kj} & \cdots & b_{kr} \\ & & & \vdots & & \\ b_{q1} & b_{q2} & \cdots & b_{qj} & \cdots & b_{qr} \end{pmatrix}$$

$C = A \times B$

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1j} & \cdots & c_{1r} \\ c_{21} & c_{22} & \cdots & c_{2j} & \cdots & c_{2r} \\ & & & \vdots & & \\ c_{k1} & c_{k2} & \cdots & c_{kj} & \cdots & c_{kr} \\ & & & \vdots & & \\ c_{p1} & c_{p2} & \cdots & c_{pj} & \cdots & c_{pr} \end{pmatrix}$$

$$c_{kj} = \sum_{i=1}^{q} a_{ki} b_{ij}$$

Σύνθετες Δομές Δεδομένων

Πολλαπλασιασμός πινάκων $C[1:p, 1:r] = A[1:p, 1:q] \times B[1:q, 1:r]$

```
for (i = 0; i < p; i++)
        for (j = 0; j < r; j++)
        {
                C[i][j]=0;
                for (k = 0; k < q; k++)
                {
                        C[i][j] += A[i][k]*B[k][j];
                }
        }
```

Εκτελεί $\Theta(pqr)$ πράξεις

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

$$A = \begin{pmatrix} A[0,0] & A[0,1] & \cdots & A[0,j-1] & \cdots & A[0,q-1] \\ A[1,0] & A[1,1] & \cdots & A[1,j-1] & \cdots & A[1,q-1] \\ & & & \vdots & & \\ A[k-1,0] & A[k-1,1] & \cdots & A[k-1,j-1] & \cdots & A[k-1,q-1] \\ & & & \vdots & & \\ A[p-1,0] & A[p-1,1] & \cdots & A[p-1,j-1] & \cdots & A[p-1,q-1] \end{pmatrix}$$

Αποθήκευση κατά γραμμές

$$[\ A[0,0] \quad A[0,1] \quad \ldots \quad A[0,q-1] \qquad A[1,0] \quad A[1,1] \quad \ldots \quad A[1,q-1] \qquad \ldots$$

$$A[k-1,0] \quad A[k-1,1] \quad \ldots \quad A[k-1,q-1] \quad \ldots \quad A[p-1,0] \quad A[p-1,1] \quad \ldots \quad A[p-1,q-1]\ ]$$

Το στοιχείο A[i,j] βρίσκεται στη θέση X + (i*q+j)*L, όπου X η θέση του πρώτου στοιχείου A[0,0] και L το μέγεθος του κάθε στοιχείου

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

$$A = \begin{pmatrix} A[0,0] & A[0,1] & \cdots & A[0,j-1] & \cdots & A[0,q-1] \\ A[1,0] & A[1,1] & \cdots & A[1,j-1] & \cdots & A[1,q-1] \\ & & & \vdots & & \\ A[k-1,0] & A[k-1,1] & \cdots & A[k-1,j-1] & \cdots & A[k-1,q-1] \\ & & & \vdots & & \\ A[p-1,0] & A[p-1,1] & \cdots & A[p-1,j-1] & \cdots & A[p-1,q-1] \end{pmatrix}$$

Αποθήκευση κατά στήλες

$$[\ A[0,0] \quad A[1,0] \quad \ldots \quad A[p-1,0] \qquad A[0,1] \quad A[1,1] \quad \ldots \quad A[p-1,1] \qquad \ldots$$

$$A[0,j-1] \quad A[1,j-1] \quad \ldots \quad A[p-1,j-1] \qquad \ldots \qquad A[0,q-1] \quad A[1,q-1] \quad \ldots \quad A[p-1,q-1]\ ]$$

Το στοιχείο A[i,j] βρίσκεται στη θέση X + (i+j*p)*L, όπου X η θέση του πρώτου στοιχείου A[0,0] και L το μέγεθος του κάθε στοιχείου

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

Σε πολλές εφαρμογές προκύπτουν πίνακες με ειδική μορφή. Μπορούμε να αποθηκεύσουμε τα στοιχεία ενός τέτοιοι πίνακα σε ένα μονοδιάστατο πίνακα για εξοικονόμηση χώρου.

Συμμετρικοί πίνακες $A[1:n, 1:n], A[i,j] = A[j,i]$

$$A = \begin{pmatrix} 12 & 3 & -2 & 0 & 5 & 4 \\ 3 & 1 & 4 & 8 & -1 & 3 \\ -2 & 4 & 0 & 0 & 7 & -9 \\ 0 & 8 & 0 & -7 & 12 & 6 \\ 5 & -1 & 7 & 12 & 1 & 13 \\ 4 & 3 & -9 & 6 & 13 & -9 \end{pmatrix}$$

Συμπιεσμένη αποθήκευση κατά γραμμές

$$[\ 12\ \ 3\ \ -2\ \ 0\ \ 5\ \ 4\ \ |\ 1\ \ 4\ \ 8\ \ -1\ \ 3\ \ |\ 0\ \ 0\ \ 7\ \ -9\ \ |\ -7\ \ 12\ \ 6\ \ |\ 1\ \ 13\ \ |\ -9\ ]$$

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

Σε πολλές εφαρμογές προκύπτουν πίνακες με ειδική μορφή. Μπορούμε να αποθηκεύσουμε τα στοιχεία ενός τέτοιοι πίνακα σε ένα μονοδιάστατο πίνακα για εξοικονόμηση χώρου.

Συμμετρικοί πίνακες $A[1:n, 1:n], A[i,j] = A[j,i]$

$$A = \begin{pmatrix} 12 & 3 & -2 & 0 & 5 & 4 \\ 3 & 1 & 4 & 8 & -1 & 3 \\ -2 & 4 & 0 & 0 & 7 & -9 \\ 0 & 8 & 0 & -7 & 12 & 6 \\ 5 & -1 & 7 & 12 & 1 & 13 \\ 4 & 3 & -9 & 6 & 13 & -9 \end{pmatrix}$$

Συμπιεσμένη αποθήκευση κατά γραμμές

[12 3 −2 0 5 4 | 1 4 8 −1 3 | 0 0 7 −9 | −7 12 6 | 1 13 | −9]

Αριθμός στοιχείων = $n + (n-1) + (n-2) + \ldots + 2 + 1 = n(n+1)/2$

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

Σε πολλές εφαρμογές προκύπτουν πίνακες με ειδική μορφή. Μπορούμε να αποθηκεύσουμε τα στοιχεία ενός τέτοιοι πίνακα σε ένα μονοδιάστατο πίνακα για εξοικονόμηση χώρου.

Άνω τριγωνικοί πίνακες $A[1:n, 1:n], A[i,j] = 0$ αν $i > j$

$$A = \begin{pmatrix} 12 & 3 & -2 & 0 & 5 & 4 \\ 0 & 1 & 4 & 8 & -1 & 3 \\ 0 & 0 & 0 & 0 & 7 & -9 \\ 0 & 0 & 0 & -7 & 12 & 6 \\ 0 & 0 & 0 & 0 & 1 & 13 \\ 0 & 0 & 0 & 0 & 0 & -9 \end{pmatrix}$$

Συμπιεσμένη αποθήκευση κατά γραμμές

$$[\ 12\ \ 3\ \ -2\ \ 0\ \ 5\ \ 4\ \ 1\ \ 4\ \ 8\ \ -1\ \ 3\ \ 0\ \ 0\ \ 7\ \ -9\ \ -7\ \ 12\ \ 6\ \ 1\ \ 13\ \ -9\ ]$$

Αριθμός στοιχείων = $n + (n-1) + (n-2) + \ldots + 2 + 1 = n(n+1)/2$

Σύνθετες Δομές Δεδομένων

Αποθήκευση διδιάστατου πίνακα ως μονοδιάστατου

Σε πολλές εφαρμογές προκύπτουν πίνακες με ειδική μορφή. Μπορούμε να αποθηκεύσουμε τα στοιχεία ενός τέτοιοι πίνακα σε ένα μονοδιάστατο πίνακα για εξοικονόμηση χώρου.

Κάτω τριγωνικοί πίνακες $A[1:n, 1:n], A[i,j] = 0$ αν $i < j$

$$A = \begin{pmatrix} 12 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 5 & 0 & 0 & 0 \\ 1 & 2 & 8 & -7 & 0 & 0 \\ 10 & -6 & 12 & -9 & 1 & 0 \\ -3 & 5 & -4 & 1 & 13 & -2 \end{pmatrix}$$

Συμπιεσμένη αποθήκευση κατά γραμμές

[12 | 3 1 | 0 4 5 | 1 2 8 −7 | 10 −6 12 −9 1 | −3 5 −4 1 13 −2]

Αριθμός στοιχείων = $n + (n-1) + (n-2) + \ldots + 2 + 1 = n(n+1)/2$

Σύνθετες Δομές Δεδομένων

Αραιοί πίνακες $A[1:m, 1:n]$ με $N = o(mn)$ μη μηδενικά στοιχεία

$$A = \begin{pmatrix} 12 & 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 5 & -11 & 0 & 2 \\ 0 & 0 & 8 & -7 & 12 & 0 \\ -6 & 0 & 0 & -9 & 0 & 3 \\ 0 & -3 & 0 & 0 & 1 & 12 \end{pmatrix}$$

Αποθηκεύουμε τα μη μηδενικά στοιχεία διαδοχικά ανά γραμμές σε μονοδιάστατο πίνακα V

$$V = [\ \boxed{12\ \ 4}\ \boxed{3\ \ 6}\ \boxed{5\ \ -11\ \ 2}\ \boxed{8\ \ -7\ \ 12}\ \boxed{-6\ \ -9\ \ 3}\ \boxed{-3\ \ 1\ \ 12}\ ]$$

Επιπλέον αποθηκεύουμε τη στήλη και γραμμή που αντιστοιχεί σε κάθε μη μηδενικό στοιχείο σε μονοδιάστατους πίνακες col και row

$$col = [\ 1\ \ 4\ \ 1\ \ 3\ \ 3\ \ 4\ \ 6\ \ 3\ \ 4\ \ 5\ \ 1\ \ 4\ \ 6\ \ 2\ \ 5\ \ 6\ ]$$

$$row = [\ 1\ \ 1\ \ 2\ \ 2\ \ 3\ \ 3\ \ 3\ \ 4\ \ 4\ \ 4\ \ 5\ \ 5\ \ 5\ \ 6\ \ 6\ \ 6\ ]$$

Σύνθετες Δομές Δεδομένων

Αραιοί πίνακες $A[1:m, 1:n]$ με $N = o(mn)$ μη μηδενικά στοιχεία

$$A = \begin{pmatrix} 12 & 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 5 & -11 & 0 & 2 \\ 0 & 0 & 8 & -7 & 12 & 0 \\ -6 & 0 & 0 & -9 & 0 & 3 \\ 0 & -3 & 0 & 0 & 1 & 12 \end{pmatrix}$$

Χώρος = $3N$

Αποθηκεύουμε τα μη μηδενικά στοιχεία διαδοχικά ανά γραμμές σε μονοδιάστατο πίνακα V

$$V = [\ 12 \quad 4 \quad 3 \quad 6 \quad 5 \quad -11 \quad 2 \quad 8 \quad -7 \quad 12 \quad -6 \quad -9 \quad 3 \quad -3 \quad 1 \quad 12\ ]$$

Επιπλέον αποθηκεύουμε τη στήλη και γραμμή που αντιστοιχεί σε κάθε μη μηδενικό στοιχείο σε μονοδιάστατους πίνακες col και row

$$col = [\ 1 \quad 4 \quad 1 \quad 3 \quad 3 \quad 4 \quad 6 \quad 3 \quad 4 \quad 5 \quad 1 \quad 4 \quad 6 \quad 2 \quad 5 \quad 6\ ]$$

$$row = [\ 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 4 \quad 4 \quad 4 \quad 5 \quad 5 \quad 5 \quad 6 \quad 6 \quad 6\ ]$$

Σύνθετες Δομές Δεδομένων

Αραιοί πίνακες $A[1:m, 1:n]$ με $N = o(mn)$ μη μηδενικά στοιχεία

$$A = \begin{pmatrix} 12 & 0 & 0 & 4 & 0 & 0 \\ 3 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 5 & -11 & 0 & 2 \\ 0 & 0 & 8 & -7 & 12 & 0 \\ -6 & 0 & 0 & -9 & 0 & 3 \\ 0 & -3 & 0 & 0 & 1 & 12 \end{pmatrix}$$

Χώρος = $2N + n$

Αποθηκεύουμε τα μη μηδενικά στοιχεία διαδοχικά ανά γραμμές σε μονοδιάστατο πίνακα V

$$V = [\ \boxed{12 \quad 4}\ \boxed{3 \quad 6}\ \boxed{5 \quad -11 \quad 2}\ \boxed{8 \quad -7 \quad 12}\ \boxed{-6 \quad -9 \quad 3}\ \boxed{-3 \quad 1 \quad 12}\ ]$$

Εναλλακτικά, αντί για τον πίνακα γραμμών row, αποθηκεύουμε έναν άλλο μονοδιάστατο πίνακα rowpos που δίνει τη θέση που ξεκινά στον V η κάθε γραμμή του A

$$col = [\ 1 \quad 4 \quad 1 \quad 3 \quad 3 \quad 4 \quad 6 \quad 3 \quad 4 \quad 5 \quad 1 \quad 4 \quad 6 \quad 2 \quad 5 \quad 6\ ]$$

$$rowpos = [\ 1 \quad 3 \quad 5 \quad 8 \quad 11 \quad 14\ ]$$