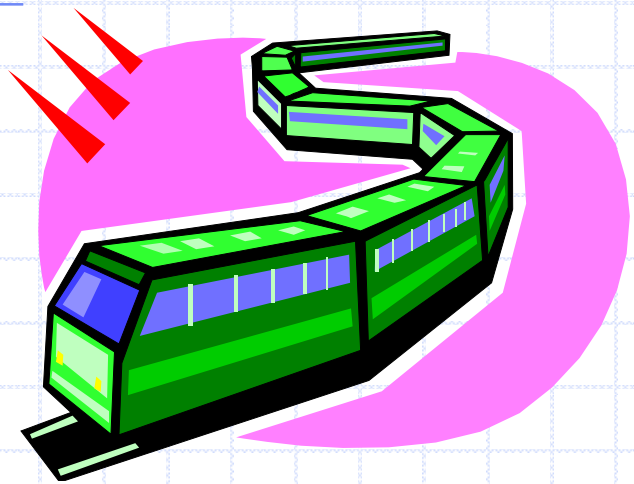
Λίστες



Αφηρημένος Τύπος Δεδομένων (ΑΤΔ) Θέσης

- Ο ΑΤΔ θέσης μοντελοποιεί την έννοια της θέσης σε μια δομή όπου αποθηκεύεται ένα αντικείμενο
- Υποστηρίζει μια ενωποιημένη όψη των διαφορετικών τρόπων αποθήκευσης δεδομένων όπως
  - ένα κελλί σε ένα πίνακα
  - ένα κόμβο σε μια συνδεδεμένη λίστα
- Μόνο μια μέθοδος:
  - object element(): επιστρέφει το στοιχείο που είναι αποθηκευμένο στη θέση

ΑΤΔ λίστας κόμβων

- Ο ΑΤΔ Λίστα Κόμβων μοντελοποιεί μια ακολουθία θέσεων που αποθηκεύουν αυθαίρετα στοιχεία
- Υποστηρίζει μια σχέση σειράς πριν/μετά μεταξύ των θέσεων
- Πρωταρχικές μέθοδοι:
  - size(), isEmpty()

Μέθοδοι προσπέλασης:

- first(), last()
- prev(p), next(p)

- Μέθοδοι ενημέρωσης:
  - set(p, e)
  - addBefore(p, e), addAfter(p, e),
  - addFirst(e), addLast(e)
  - remove(p)

Θα συμβεί μια συνθήκη λάθους αν μια θέση που περνάει σαν παράμετρος σε μια από τις πράξεις λίστας δεν είναι έγκυρη. Τέτοιες περιπτώσεις μπορεί να είναι:

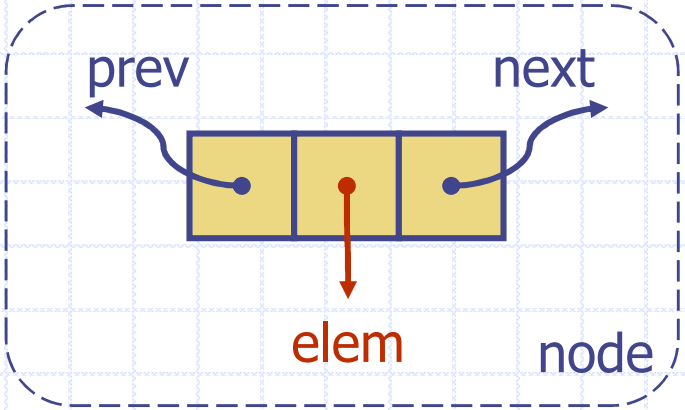
- p=null
- το p είχε προηγουμένως διαγραφεί από τη λίστα
- το p είναι μια θέση άλλης λίστας
- το p είναι η πρώτη θέση μιας λίστας και καλούμε την prev(p)
- το p είναι η τελευταία θέση μιας λίστας και καλούμε την next(p)

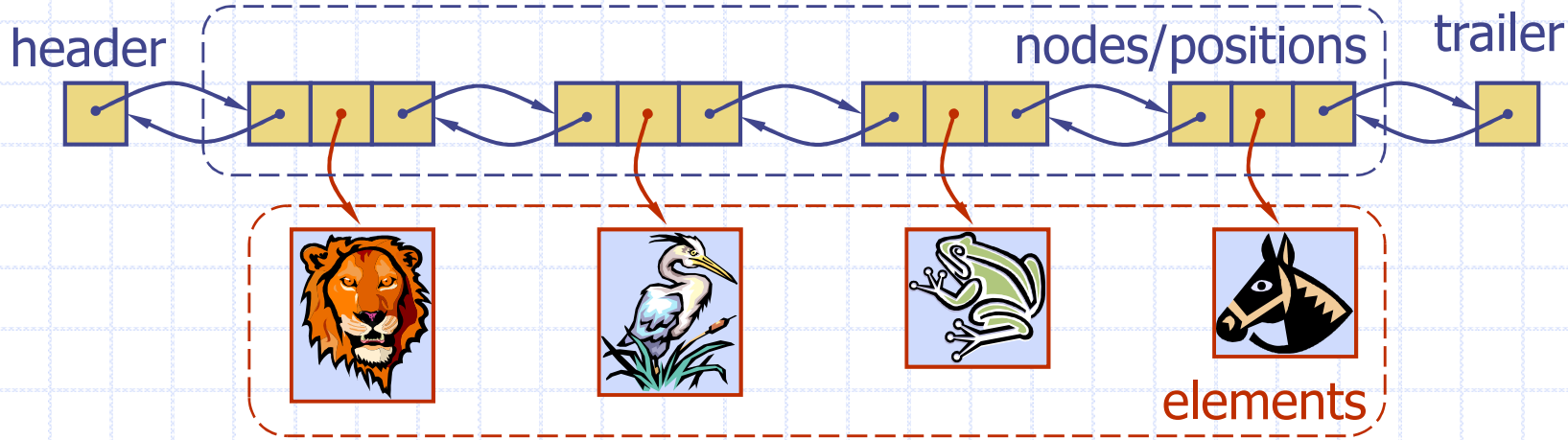
Το παράδειγμα δείχνει πράξεις σε μια κενή λίστα S. Οι πράμετροι p_1 p_2 κοκ. Υποδηλώνουν διαφορετικές θέσεις και δείχνουμε το αντικείμενο που αποθηκεύεται στην αντίστοιχη θέση.

Πράξη	Έξοδος	S
addFirst(8)	-	(8)
first()	$P_1(8)$	(8)
addAfter(p_1,5)	-	(8,5)
next(p_1)	$P_2(5)$	(8,5)
addBefore(p_2, 3)	-	(8,3,5)
prev(p_2)	$P_3(3)$	(8,3,5)
addFirst(9)	-	(9,8,3,5)
Last()	$P_2(5)$	(9,8,3,5)
eemove(first())	9	(8,3,5)
set(p_3,7)	3	(8,7,5)
addAfter(first(),2)	-	(8,2,7,5)

Διπλά συνδεδμένη λίστα

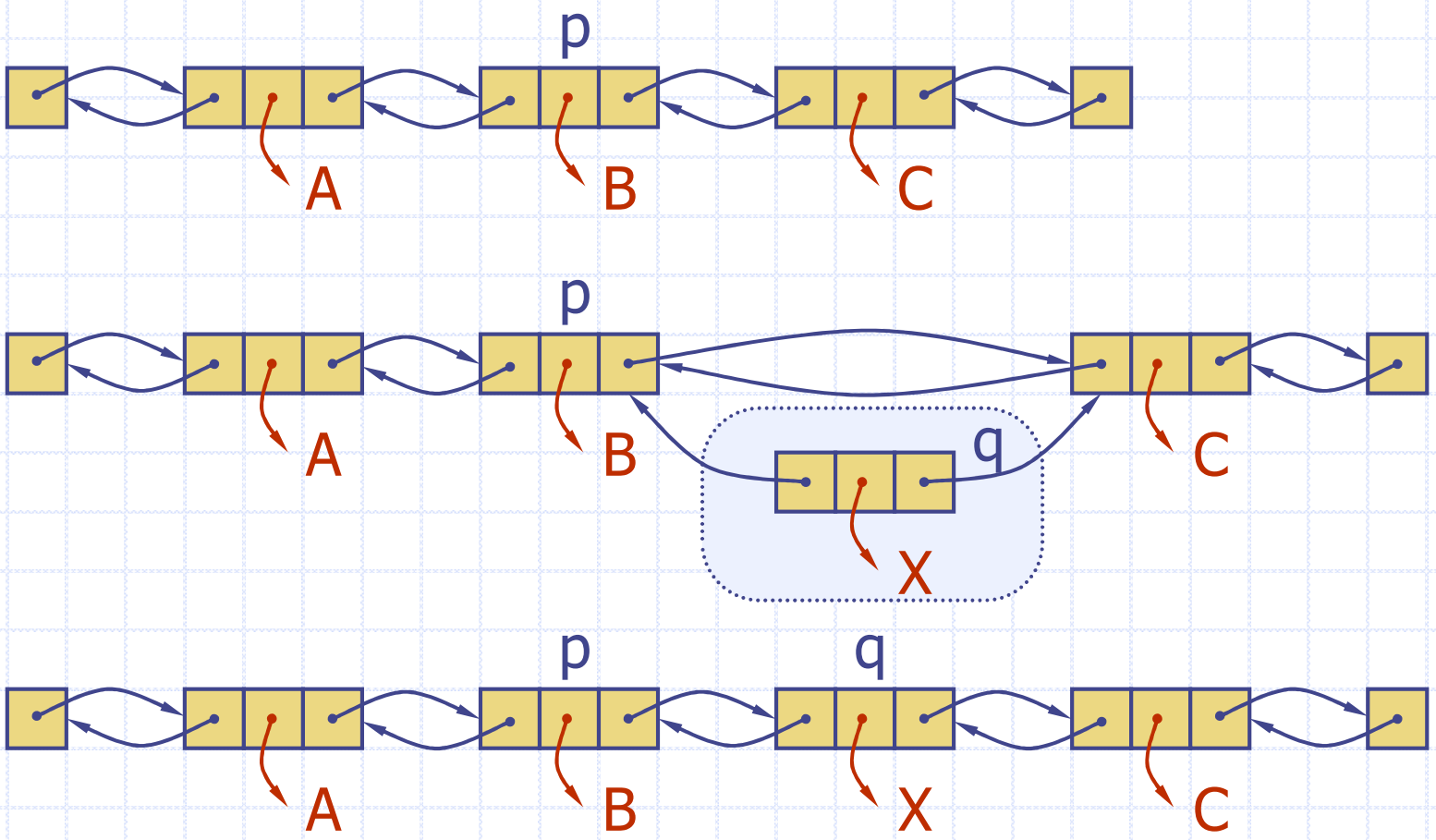
- Μια διπλά συνδεδεμένη λίστα αποτελεί τη φυσική υλοποίηση του ΑΤΔ λίστας κόμβων
- Οι κόμβοι υλοποιούν τη θέση και αποθηκεύουν:
  - στοιχείο
  - σύνδεσμο στον προηγούμενο κόμβο
  - σύνδεσμο στον επόμενο κόμβο
- Ειδικούς κόμβους αρχής και τέλους





Εισαγωγή

- Οπτικοποίηση της εισαγωγής insertAfter(p, X), που επιστρέφει τη θέση q



Αλγόριθμος Εισαγωγής

Algorithm addAfter(p,e):

Create a new node v

v.setElement(e)

v.setPrev(p) {σύνδεση του v με τον προηγούμενό του}

v.setNext(p.getNext()) {σύνδεση του v με τον επόμενο του}

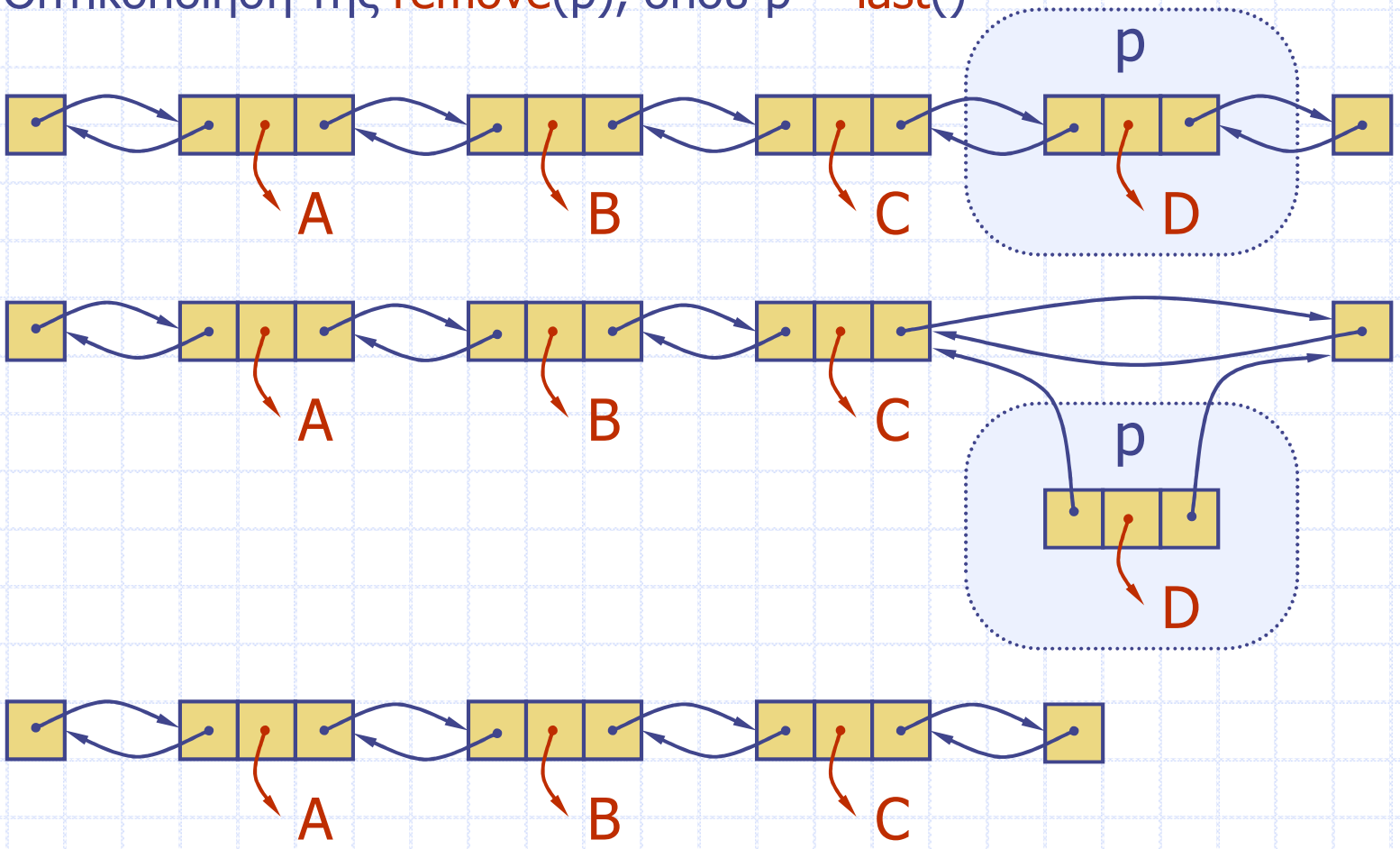
(p.getNext()).setPrev(v) {σύνδεση του παλαιού διαδόχου του p με το παλαιό v}

p.setNext(v) {σύνδεση του p με τον νέο διάδοχό του , v}

return v {τη θέση του στοιχείου e}

Διαγραφή

- Οπτικοποίηση της remove(p), όπου p = last()



Αλγόριθμος Διαγραφής

Algorithm remove(p):

t = p.element {μια προσωρινή μεταβλητή για την τιμή που επιστρέφεται }

(p.getPrev()).setNext(p.getNext()) {σύνδεση με το p}

(p.getNext()).setPrev(p.getPrev())

p.setPrev(**null**) {ακύρωση της θέσης p}

p.setNext(**null**)

return t

Απόδοση

- Στην υλοποίηση του ΑΤΔ λίστας με χρήση διπλά συνδεδεμένης λίστας
  - Ο χώρος που απαιτεί μια λίστα με n στοιχεία είναι $O(n)$
  - Ο χώρος που απαιτεί κάθε θέση της λίστας είναι $O(1)$
  - Όλες οι πράξεις του ΑΤΔ λίστας τρέχουν σε χρόνο $O(1)$
  - Η πράξη element() του ΑΤΔ τρέχει σε $O(1)$ χρόνο

Υποθέστε ότι διατηρούμε μια συλλογή C από στοιχεία έτσι ώστε, κάθε φορά που προσθέτουμε ένα νέο στοιχείο στη συλλογή, αντιγράφουμε το περιεχόμενο της C σε μια νέα γραμμική λίστα με ακριβώς το σωστό μέγεθος. Ποιος είναι ο χρόνος τρεξίματος για την προσθήκη n στοιχείων σε μια αρχικά κενή συλλογή C.

Δίδεται μια λίστα L από n θετικούς ακέραιους, που ο καθένας παριστάνεται $k=\lceil \log n \rceil+1$ bits, περιγράψτε μια μέθοδο που τρέχει σε O(n) χρόνο για την εύρεση ενός k bit ακέραιου που δεν είναι στη λίστα L.

Δίδεται μια λίστα L με n αυθαιρέτους ακέραιους, σχεδιάστε μια O(n) μέθοδο για την εύρεση ενός ακεραίου που δεν μπορεί να σχηματισθεί σαν άθροισμα δυο ακεραίων στην L.