HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2008-09
Παναγιώτα Φατούρου

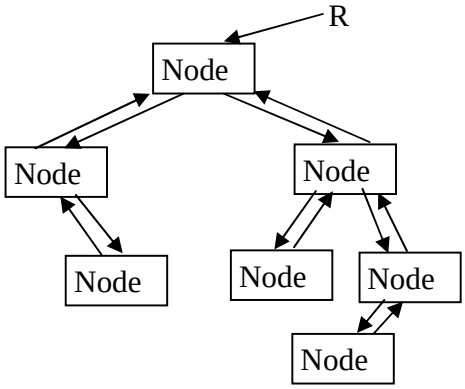
Προγραμματιστική Εργασία
2ο Μέρος

Ημερομηνία Παράδοσης: Παρασκευή, 12 Δεκεμβρίου 2008, ώρα 23:59.
Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα submit.

Στο 2ο μέρος της προγραμματιστικής εργασίας ζητείται να γίνουν τροποποιήσεις στο πρόγραμμα του 1ου μέρους. Το πρόγραμμα που θα προκύψει θα περιγράφει (προσομοιώνει) και πάλι τη λειτουργία ενός διομότιμου συστήματος (το οποίο ακολουθεί τους κανόνες λειτουργίας που έχουν περιγραφεί στο 1ο μέρος). Το 2ο μέρος της εργασίας αποτελείται από δύο προγραμματιστικά καθήκοντα.

1ο Προγραμματιστικό καθήκον: Υλοποίηση Ταξινομημένων Διπλά-Συνδεδεμένων Δυαδικών Δένδρων (Doubly-Linked Binary Search Trees)

Οι πληροφορίες για τους κόμβους που συμμετέχουν κάθε χρονική στιγμή στο σύστημα θα πρέπει τώρα να αποθηκεύονται σε ένα ταξινομημένο δένδρο, δηλαδή η ταξινομημένη κυκλική διπλά συνδεδεμένη λίστα των κόμβων θα πρέπει να αντικατασταθεί με ένα ταξινομημένο διπλά-συνδεδεμένο δένδρο (doubly-linked binary search tree). Το δένδρο αυτό (Σχήμα 1) ονομάζεται *δένδρο κόμβων* (NodeTree).



Σχήμα 1

Για κάθε κόμβο n διατηρούνται οι ακόλουθες πληροφορίες:

- id: το αναγνωριστικό του κόμβου που είναι ένας ακέραιος (και αποτελεί το κλειδί του κόμβου).
- DocL: δείκτης στο πρώτο στοιχείο μιας **απλά συνδεδεμένης** λίστας η οποία αποθηκεύει πληροφορίες για τα αρχεία που διαχειρίζεται ο κόμβος n. Η λίστα αυτή ονομάζεται λίστα αρχείων (DocList) του κόμβου. Οι πληροφορίες που αποθηκεύονται στους κόμβους της είναι ίδιες με αυτές του 1ου μέρους της εργασίας.
- lc: ένας δείκτης προς τον αριστερό θυγατρικό κόμβο του n.
- rc: ένας δείκτης προς το δεξιό θυγατρικό κόμβο του n.
- p: ένας δείκτης προς τον πατρικό κόμβο του n.

Η μορφή των δομών δεδομένων που χρησιμοποιεί το πρόγραμμα φαίνεται στο Σχήμα 2, όπου η μεταβλητή R είναι ένας δείκτης στη ρίζα του δένδρου των κόμβων.

Όπως και στο 1ο μέρος της εργασίας, το πρόγραμμα θα πρέπει να εκτελείται καλώντας την

run <m> <input-file>

όπου run είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος, <m> είναι ένας ακέραιος που καθορίζει το μέγιστο πλήθος κόμβων στο σύστημα (το οποίο θα είναι 2^m) και <input-file> είναι το όνομα ενός αρχείου εισόδου που περιέχει γεγονότα των μορφών που περιγράφηκαν στο 1ο μέρος. Η συμπεριφορά κάθε γεγονότος πρέπει να είναι αυτή που περιγράφεται στην εκφώνηση του 1ου μέρους της εργασίας.

Το πρόγραμμα θα πρέπει να τυπώνει αντίστοιχες πληροφορίες κατά την εκτέλεση κάποιου γεγονότος με αυτές που τυπώνονταν στο 1ο μέρος της εργασίας.

2ο Προγραμματιστικό καθήκον: Υλοποίηση Ουράς Προτεραιότητας

Υποθέστε ότι κάθε γεγονός συνοδεύεται από μια χρονοσφραγίδα <tm>. Έτσι, τα γεγονότα που περιγράφονται στο αρχείο εισόδου έχουν τώρα την ακόλουθη μορφή:

- J <id> <tm>
- L <id> <tm>
- I <id> <tp> <tm>
- F <id> <tm>
- D <id> <tm>
- P <tm>

όπου <tm> είναι η χρονοσφραγίδα κάθε γεγονότος (δηλαδή ένας ακέραιος που εκφράζει τη χρονική στιγμή στην οποία πρέπει να εκτελεστεί το γεγονός), ενώ τα υπόλοιπα πεδία (<id>,<tp>) παίζουν τον ίδιο ρόλο όπως και στο 1ο μέρος της εργασίας.

Οι ενέργειες που σηματοδοτεί το κάθε γεγονός είναι ακριβώς οι ίδιες με αυτές που περιγράφονται στο 1ο μέρος της προγραμματιστικής εργασίας. Στο μέρος αυτό ωστόσο, το πρόγραμμα θα διαβάζει το αρχείο εισόδου και αντί να εκτελεί κατευθείαν κάθε ένα από τα γεγονότα που περιγράφονται εκεί, θα τα τοποθετεί πρώτα σε μια ουρά προτεραιότητας βάσει της χρονοσφραγίδας τους, όπου τη μέγιστη προτεραιότητα έχει το γεγονός με τη μικρότερη χρονοσφραγίδα. Στη συνέχεια επαναληπτικά, όσο η ουρά προτεραιότητας περιέχει ακόμη γεγονότα, το γεγονός με τη μεγαλύτερη προτεραιότητα θα εξέρχεται της ουράς προτεραιότητας και θα εκτελείται. Υλοποιήστε την ουρά προτεραιότητας χρησιμοποιώντας τη δομή του σωρού. Για κάθε γεγονός διατηρούνται στο σωρό οι ακόλουθες πληροφορίες:

- <type>: τύπος γεγονότος που είναι ένας ακέραιος (π.χ., μπορεί να έχει την τιμή 1 για γεγονότα τύπου join, την τιμή 2 για γεγονότα τύπου leave, κλπ.)
- <tm>: χρονοσφραγίδα γεγονότος (που αποτελεί και το κλειδί του).
- <id>: αναγνωριστικό κόμβου ή αρχείου ανάλογα με το είδος του γεγονότος
- <tp>: είδος αρχείου (χρησιμοποιείται μόνο για γεγονότα που αφορούν τη διαχείριση των αρχείων που έχουν αποθηκευθεί στο σύστημα).

Το πρόγραμμα θα πρέπει να εκτελείται καλώντας την εντολή

run <m> <input-file> <number_of_events>

όπου run είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος, <m> είναι ένας ακέραιος που καθορίζει το μέγιστο πλήθος κόμβων στο σύστημα (το οποίο θα είναι 2^m), <input-file> είναι το όνομα

ενός αρχείου εισόδου που περιέχει γεγονότα **με χρονοσφραγίδες** και <number_of_events> είναι το μέγιστο πλήθος γεγονότων που θα εκτελεστούν.

- Το πρόγραμμα θα πρέπει να τυπώνει αντίστοιχες πληροφορίες κατά την εκτέλεση κάποιου γεγονότος με αυτές που τυπώνονταν στο 1^{ο} μέρος της εργασίας, έχοντας ωστόσο προσθέσει μετά τη λέξη DONE σε κάθε μήνυμα που τυπώνεται την πληροφορία TM = <tm>, όπου <tm> είναι η χρονοσφραγίδα του γεγονότος. Για παράδειγμα, μετά το πέρας της εκτέλεσης ενός γεγονότος τύπου Join, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  J <id> DONE, TM = <tm>, PRED = <pid>, SUCC = <sid>, DocList =<fid1, fid2, …, fidr>

  όπου <tm> είναι η χρονοσφραγίδα του γεγονότος, <pid> και <sid> είναι τα αναγνωριστικά του προηγούμενου και του επόμενου, αντίστοιχα, του προς εισαγωγή κόμβου στο δένδρο των κόμβων, και fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που έχουν αποθηκευτεί στη λίστα των αρχείων του προς εισαγωγή κόμβου. Αντίστοιχες αλλαγές θα πρέπει να γίνουν σε όλα τα μηνύματα που τυπώνονται.

Οδηγίες και Συμβουλές για την Ομαλή Διεκπεραίωση της Εργασίας

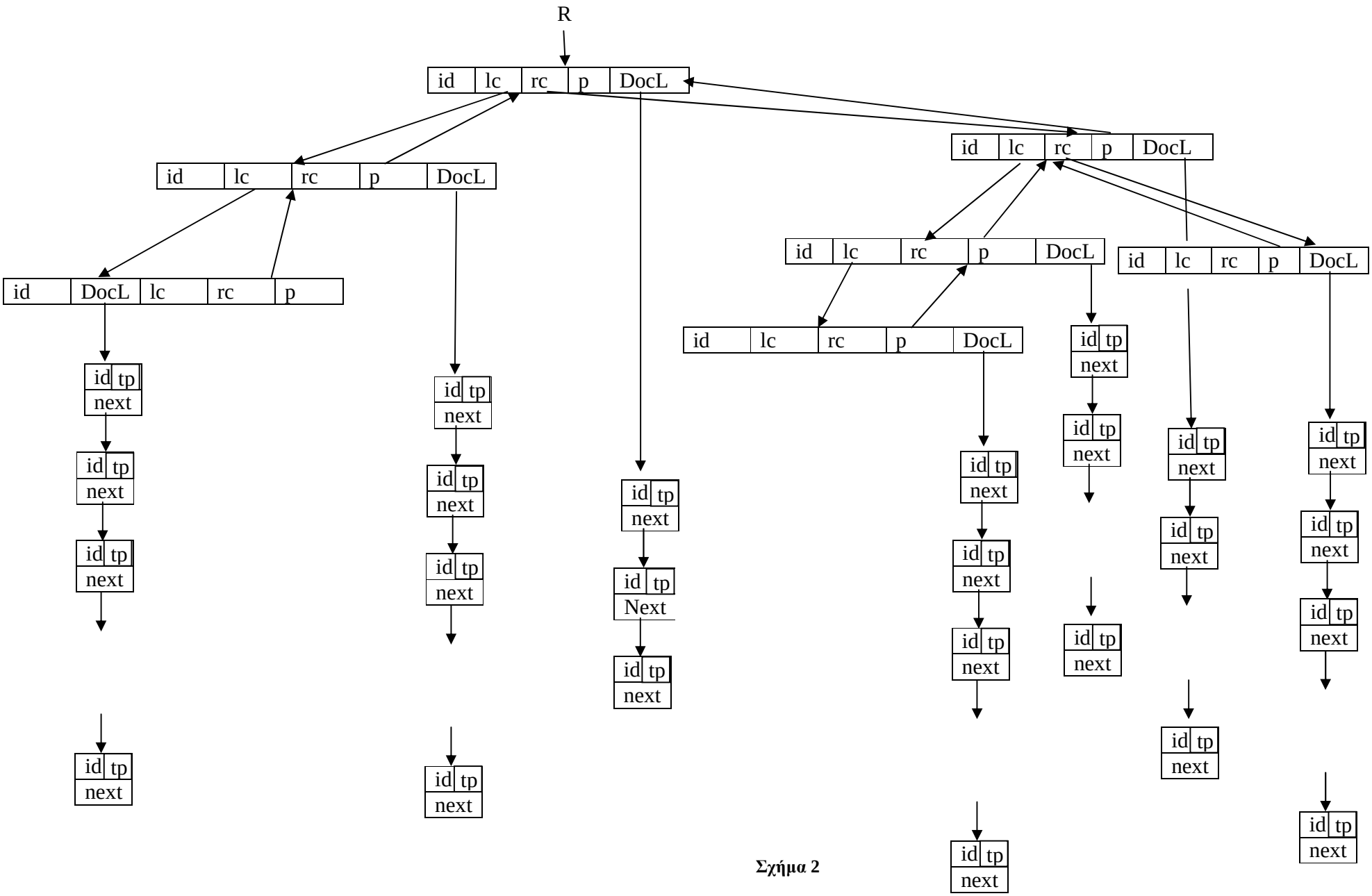
Η εργασία θα πρέπει να πραγματοποιηθεί σε βήματα. Ο κάθε φοιτητής είναι υπεύθυνος να αποφασίσει ποια βήματα ταιριάζουν στον τρόπο εργασίας του. Στη συνέχεια, παρατίθεται μια δυνητική ακολουθία βημάτων που θα μπορούσε να ακολουθηθεί για την ομαλή διεκπεραίωση της εργασίας.

Βήμα 1: Ξεκινήστε υλοποιώντας ένα ταξινομημένο διπλά-συνδεδεμένο δένδρο (δηλαδή ένα ταξινομημένο δένδρο του οποίου κάθε κόμβος αποθηκεύει έναν επιπρόσθετο δείκτη προς τον πατρικό του κόμβο). Είναι αξιοσημείωτο ότι ενδεχόμενα χρειαστείτε να υλοποιήσετε περισσότερες συναρτήσεις από τις BinarySearchInsert(), BinarySearchDelete() και BinarySearchLookUp() προκειμένου να είναι δυνατή η υλοποίηση των διαφόρων γεγονότων στο βήμα 2. Μελετήστε προσεκτικά τις συναρτήσεις που είναι απαραίτητο να υλοποιηθούν πριν αρχίσετε τη συγγραφή κώδικα. Μόνο όταν ο κώδικας του βήματος 1 εκτελείται σωστά θα πρέπει να προχωρήστε στο βήμα 2.

Βήμα 2: Αντικαταστήστε τη λίστα των κόμβων του 1^{ου} μέρους της εργασίας με δένδρο κόμβων χρησιμοποιώντας την υλοποίηση του Βήματος 1.

Βήμα 3: Σε ένα άλλο αρχείο, υλοποιήστε μια ουρά προτεραιότητας χρησιμοποιώντας σωρό.

Βήμα 4: Συνδυάστε τους κώδικες των αρχείων των Βημάτων 2 και 3. Ίσως να απαιτούνται μικρές τροποποιήσεις στη συνάρτηση main προκειμένου το πρόγραμμα να εκτελείται καλώντας τη run με τον τρόπο που περιγράφηκε παραπάνω. Εξετάστε την ορθότητα των γεγονότων όλων των τύπων.



Σχήμα 2