HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2022-23
Παναγιώτα Φατούρου

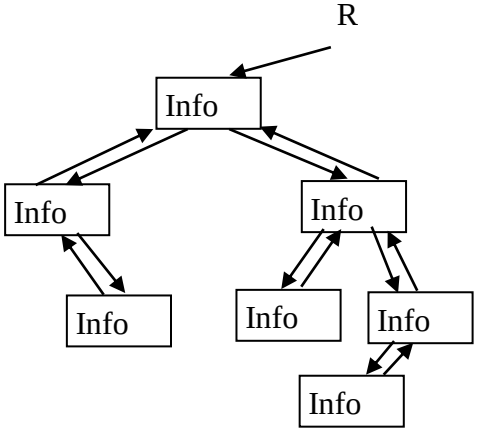
Προγραμματιστική Εργασία
2ο Μέρος

Ημερομηνία Παράδοσης: Παρασκευή, 23/12/2022 (ώρα 23:59).
Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα turnin.

Στο μέρος αυτό της προγραμματιστικής εργασίας ζητείται να γίνουν τροποποιήσεις στο πρόγραμμα που προέκυψε κατά την υλοποίηση του 1ου μέρους. Το πρόγραμμα που θα προκύψει θα περιγράφει (προσομοιώνει) και πάλι τη λειτουργία ενός συστήματος δημοσιοποίησης/συνδρομής (το οποίο ακολουθεί τους κανόνες λειτουργίας που έχουν περιγραφεί στο 1ο μέρος).

Αντικατάσταση της λίστας πληροφοριών κάθε ομάδας με ένα ταξινομημένο, διπλά συνδεδεμένο, δυαδικό δένδρο αναζήτησης (binary search tree)

Οι πληροφορίες που συσχετίζονται με κάθε ομάδα κάθε χρονική στιγμή στο σύστημα, θα πρέπει τώρα να αποθηκεύονται σε ένα ταξινομημένο δυαδικό δένδρο. Επομένως, η ταξινομημένη διπλά συνδεδεμένη λίστα πληροφοριών κάθε ομάδας θα πρέπει να αντικατασταθεί με ένα ταξινομημένο, διπλά-συνδεδεμένο, δυαδικό δένδρο (doubly-linked binary search tree). Το δένδρο αυτό (Σχήμα 1) ονομάζεται *δένδρο πληροφοριών* (InfoTree).



Σχήμα 1

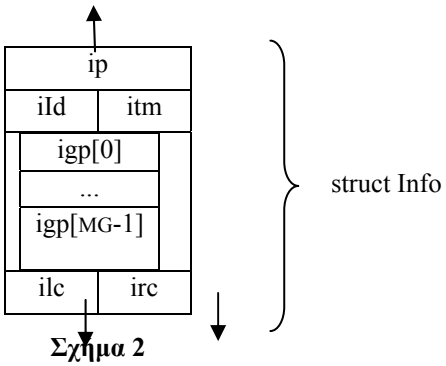
Όπως ίσχυε για τη λίστα πληροφοριών του 1ου μέρους, το δένδρο πληροφοριών κάθε ομάδας περιέχει εγγραφές που περιγράφουν τις πληροφορίες που συσχετίζονται με την ομάδα. Κάθε τέτοια πληροφορία αποθηκεύεται ως μια εγγραφή (struct Info) που αποτελείται από τα παρακάτω πεδία.

- Έναν ακέραιο iId που αποτελεί το αναγνωριστικό της πληροφορίας.
- Έναν ακέραιο itm που αποθηκεύει τη «χρονική στιγμή» στην οποία η πληροφορία αυτή κατέφθασε στο σύστημα (δηλαδή τη χρονοσφραγίδα άφιξης της πληροφορίας). Η χρονοσφραγίδα άφιξης παρέχεται από το χρήστη όπως στο μέρος 1 της προγραμματιστικής εργασίας.
- Έναν πίνακα igp[MG] των MG (Maximum_Groups) θέσεων που περιγράφει τα groups πληροφορίων στα οποία ανήκει η πληροφορία. Εάν ισχύει igp[i] == 1, $0 \leq i \leq MG-1$, τότε η πληροφορία έχει

συσχετισθεί (ή ανήκει) στην ομάδα πληροφοριών i. Αντίθετα, αν igp[i] == 0, η πληροφορία δεν συσχετίζεται με την ομάδα i.

- Έναν δείκτη ilc προς τον αριστερό θυγατρικό κόμβο του κόμβου που αποθηκεύει την πληροφορία στο δένδρο.
- Έναν δείκτη irc προς το δεξιό θυγατρικό κόμβο του κόμβου που αποθηκεύει την πληροφορία στο δένδρο.
- Έναν δείκτη ip προς τον πατρικό κόμβο του κόμβου που αποθηκεύει την πληροφορία στο δένδρο.

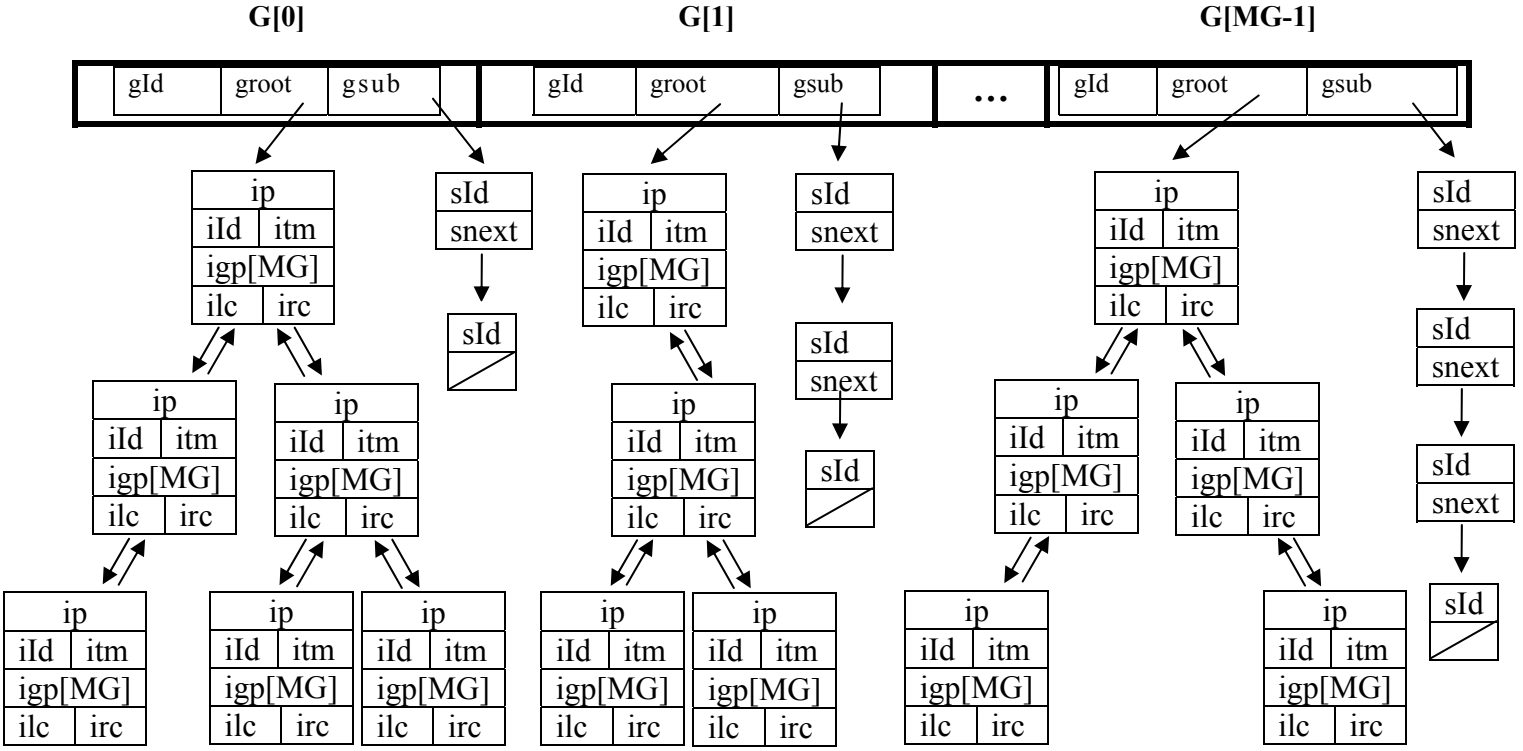
Η εγγραφή τύπου Info παρουσιάζεται στο Σχήμα 2. Το δένδρο πληροφοριών κάθε ομάδας είναι **ταξινομημένο ως προς το αναγνωριστικό** των εγγραφών που περιέχει.



Σχήμα 2

Η λίστα συνδρομών κάθε ομάδας εξακολουθεί να υπάρχει και να διέπεται από τους ίδιους κανόνες λειτουργίας όπως στο πρώτο μέρος της προγραμματιστικής εργασίας.

Η μορφή των δομών δεδομένων που περιγράφηκαν παραπάνω παρουσιάζεται στο Σχήμα 3.



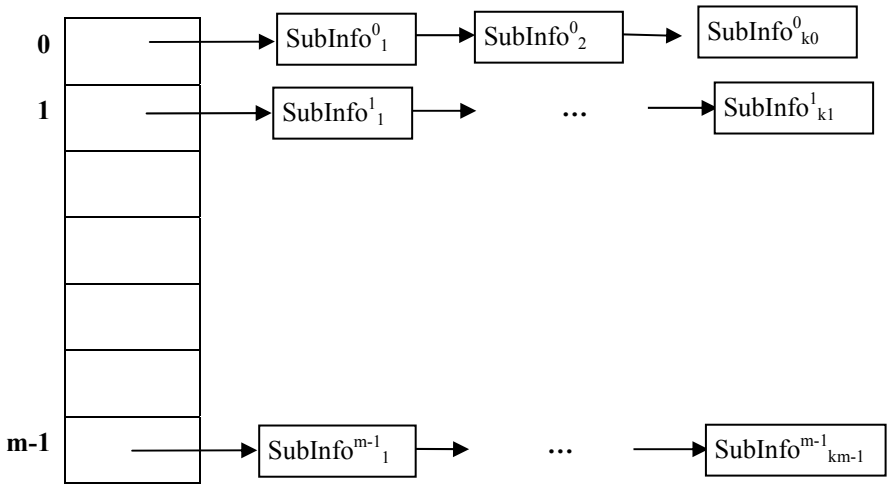
Σχήμα 3

Υλοποίηση Πίνακα Κατακερματισμού και φυλλοπροσανατολισμένων δυαδικών δένδρων (Hash tables - leaf-oriented binary search trees)

Η λίστα συνδρομητών αντικαθίσταται από έναν πίνακα κατακερματισμού, όπως φαίνεται στο Σχήμα 4. Το πλήθος θέσεων του πίνακα κατακερματισμού θα καθορίζεται από το χρήστη στη γραμμή εντολών. Ζητείται να υλοποιήσετε την τεχνική κατακερματισμού **των ταξινομημένων ξεχωριστών αλυσίδων.** Κάθε αλυσίδα διατηρείται **ταξινομημένη ως προς τα αναγνωριστικά** των συνδρομητών.

Η συνάρτηση κατακερματισμού που θα χρησιμοποιήσετε θα πρέπει να επιλέγεται τυχαία από μια καθολική οικογένεια συναρτήσεων κατακερματισμού (δηλαδή ζητείται να υλοποιήσετε μια καθολική οικογένεια συναρτήσεων κατακερματισμού, βάσει της θεωρίας που διδάσκεται στο μάθημα). Θεωρήστε ότι τα αναγνωριστικά των συνδρομητών θα επιλεχθούν από το σύνολο {0,..,<p>-1}, όπου <p> είναι ένας πρώτος αριθμός που παρέχεται από το χρήστη στη γραμμή εντολών.

Η μορφή της δομής δεδομένων που περιγράφει τους συνδρομητές παρουσιάζεται στο Σχήμα 4.



Σχήμα 4

Σε αυτό το μέρος της εργασίας, κάθε εγγραφή SubInfo περιέχει τα παρακάτω πεδία:

- έναν ακέραιο sId που αποτελεί το αναγνωριστικό του συνδρομητή.
- Έναν ακέραιο stm που αποτελεί τη χρονοσφραγίδα άφιξης του συνδρομητή στο σύστημα. Η πληροφορία αυτή παρέχεται από το χρήστη όπως και στο μέρος 1 της προγραμματιστικής εργασίας.
- Έναν πίνακα tgp[MG] ο οποίος περιέχει MG δείκτες έναν για κάθε ομάδα. Είναι αξιοσημείωτο πως ο πίνακας αυτός αποτελεί επιπρόσθετο πεδίο του struct SubInfo συγκριτικά με τη μορφή του struct SubInfo του 1^ο μέρους της εργασίας.

  Έστω οποιαδήποτε ομάδα k, $0 \leq k \leq MG-1$. Αν ο δείκτης tgp[k] μιας εγγραφής του πίνακα κατακερματισμού συνδρομητών έχει την τιμή 1, ο συνδρομητής δεν είναι εγγεγραμμένος στην ομάδα με αναγνωριστικό k. Διαφορετικά, ο δείκτης tgp[k] δείχνει στη ρίζα ενός δένδρου που αποθηκεύει πληροφορίες δημοσιευμένες στην ομάδα G[k]. Το δένδρο αυτό ονομάζεται *δένδρο κατανάλωσης*. Με κάθε συνδρομητή συσχετίζονται τόσα δένδρα κατανάλωσης όσες και οι ομάδες για τις οποίες ενδιαφέρεται ο συνδρομητής. Τους κόμβους ρίζες αυτών των δένδρων δεικτοδοτούν κατάλληλοι δείκτες του πίνακα tgp.

  Περιοδικά, ο εξυπηρέτης (server) αποστέλλει κάποιες από τις πληροφορίες που περιέχονται στις ομάδες στους συνδρομητές κάθε ομάδας. Αυτό γίνεται μέσω **ενός νέου γεγονότος που ονομάζεται Prune**. Συγκεκριμένα, όταν εκτελείται το γεγονός αυτό συμβαίνουν τα εξής. Για κάθε ομάδα G[k], $0 \leq k \leq MG-1$, διαγράφονται από το δένδρο πληροφοριών της ομάδας όλες οι πληροφορίες που έχουν

χρονοσφραγίδα μικρότερη από ή ίση με μια δοθείσα χρονοσφραγίδα που συσχετίζεται με το συγκεκριμένο γεγονός τύπου Prune και παρέχεται στο αρχείο εισόδου. Για κάθε συνδρομητή S που είναι εγγεγραμμένος στην ομάδα k, όλες αυτές οι πληροφορίες (μία προς μία) εισάγονται στο δένδρο κατανάλωσης του S που αντιστοιχεί στην ομάδα k.

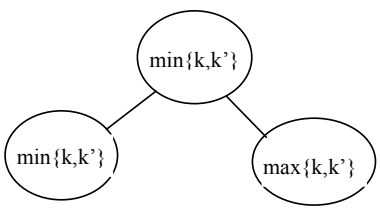
Το δένδρο κατανάλωσης είναι ένα ταξινομημένο, δυαδικό, διπλά-συνδεδεμένο, **φυλλοπροσανατολισμένο** δένδρο του οποίου τα φύλλα συνενώνονται προκειμένου να σχηματίσουν μια ταξινομημένη απλά συνδεδεμένη λίστα. Ακριβέστερος ορισμός των φυλλοπροσανατολισμένων δένδρων παρουσιάζεται στη συνέχεια. Αναλυτική περιγραφή του γεγονότος Prune περιγράφεται επίσης σε επόμενη παράγραφο.

- Έναν πίνακα sgp[MG] ο οποίος περιέχει MG δείκτες, έναν για κάθε ομάδα. Έστω οποιαδήποτε ομάδα k, $0 \leq k \leq MG-1$. Όπως στο $1^{ο}$ μέρος της προγραμματιστικής εργασίας, αν ο συνδρομητής δεν είναι εγγεγραμμένος στην ομάδα με αναγνωριστικό k, ο δείκτης sgp[k] του συνδρομητή έχει την τιμή 1. Διαφορετικά, ο δείκτης sgp[k] δείχνει σε ένα από τα στοιχεία της λίστας των φύλλων του δένδρου κατανάλωσης tgp[k] του συνδρομητή.

  Περιοδικά, κάθε συνδρομητής μπορεί να καταναλώνει τις πληροφορίες που περιέχονται στα δένδρα κατανάλωσης της εγγραφής του. Συγκεκριμένα, για κάθε τέτοιο δένδρο tgp[k] καταναλώνει όλες τις πληροφορίες που δημοσιεύθηκαν μετά από την τελευταία εκτέλεση από το συνδρομητή της λειτουργίας κατανάλωσης για το δένδρο αυτό. Αυτό γίνεται με την ανάγνωση των στοιχείων της λίστας των φύλλων του δένδρου ξεκινώντας από την εγγραφή sgp[k] (δηλαδή από την τελευταία εγγραφή που ο συνδρομητής έχει καταναλώσει σε αυτό το δένδρο κατανάλωσης) μέχρι την πιο πρόσφατη εγγραφή που τοποθετήθηκε στο δένδρο κατανάλωσης tgp[k]. Στο τέλος της διαδικασίας κατανάλωσης, ο δείκτης sgp[k] ενημερώνεται ώστε να δείχνει στο δεξιότερο φύλλο του δένδρου κατανάλωσης tgp[k]. Είναι σημαντικό πως ο δείκτης sgp[k] θα έχει την τιμή NULL εάν δεν έχει προηγηθεί γεγονός τύπου Prune ή αν το «κλάδεμα» (pruning) του δένδρου πληροφοριών της ομάδας G[k] που πραγματοποιήθηκε από τα γεγονότα τύπου Prune που προηγούνται της διαδικασίας κατανάλωσης δεν οδήγησαν στην εισαγωγή εγγραφών πληροφοριών στο δένδρο κατανάλωσης.
- Έναν δείκτη next στο επόμενο στοιχείο της αλυσίδας στην οποία ανήκει η εγγραφή που αφορά τον συνδρομητή στον πίνακα κατακερματισμού των συνδρομητών.

Κάθε ένα από τα δένδρα κατανάλωσης ενός συνδρομητή είναι ένα **ταξινομημένο , διπλά συνδεδεμένο, φυλλοπροσανατολισμένο δυαδικό δένδρο** του οποίου τα φύλλα έχουν διασυνδεθεί ώστε να σχηματίσουν μια απλά συνδεδεμένη (ταξινομημένη) λίστα. Ένα δυαδικό δένδρο ονομάζεται φυλλοπροσανατολισμένο όταν όλα τα κλειδιά που εισάγονται στο δένδρο βρίσκονται πάντα αποθηκευμένα στα φύλλα, από αριστερά προς τα δεξιά κατά μη φθίνουσα τιμή κλειδιού, ενώ στους εσωτερικούς κόμβους αποθηκεύονται κατάλληλα κλειδιά που καθιστούν εφικτή την ορθή διεκπεραίωση λειτουργιών LookUp() στο δένδρο. Ένα φυλλοπροσανατολισμένο δένδρο είναι ταξινομημένο όταν κάθε χρονική στιγμή ισχύει η εξής ιδιότητα: «Το κλειδί κάθε κόμβου u είναι μεγαλύτερο από ή ίσο με καθένα από τα κλειδιά των κόμβων του αριστερού υποδένδρου του u και μεγαλύτερο από καθένα από τα κλειδιά των κόμβων που βρίσκονται στο δεξιό υποδένδρο του u.»

Η κύρια διαφορά των ταξινομημένων φυλλοπροσανατολισμένων δένδρων από τα ταξινομημένα δυαδικά δένδρα είναι πως κάθε κλειδί που εισάγεται στο δένδρο βρίσκεται σε κάποιο φύλλο και έτσι αναζητήσεις κλειδιών είναι επιτυχημένες μόνο αν βρίσκουν το ζητούμενο κλειδί σε έναν από τους κόμβους φύλλα του δένδρου. Πιο συγκεκριμένα, για την εισαγωγή ενός νέου κλειδιού k, ακολουθείται ο κλασικός αλγόριθμος αναζήτησης του k σε ένα ταξινομημένο δυαδικό (όχι απαραίτητα φυλλοπροσανατολισμένο) δένδρο μέχρι να

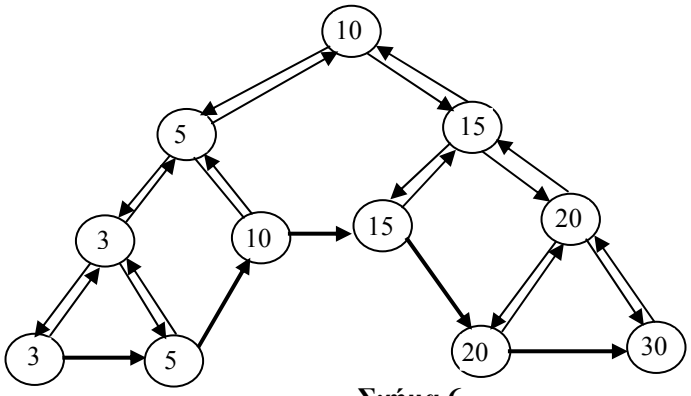


Σχήμα 5

προσεγγιστεί κάποιος κόμβος φύλλο (εύρεση του προς αναζήτηση κλειδιού σε εσωτερικό κόμβο δεν αρκεί αφού τα κλειδιά που έχουν εισαχθεί στο δένδρο πρέπει να βρίσκονται όλα στα φύλλα). Ο τελευταίος κόμβος στο μονοπάτι αναζήτησης είναι κάποιο φύλλο. Έστω k' το κλειδί του φύλλου αυτού. Τότε, η εισαγωγή γίνεται αντικαθιστώντας το φύλλο με κλειδί k' με την τριάδα κόμβων που φαίνεται στο Σχήμα 5.

Είναι αξιοσημείωτο ότι , βάσει ορισμού, σε ένα φυλλοπροσανατολισμένο δένδρο, οι εσωτερικοί κόμβοι έχουν και τους δύο θυγατρικούς δείκτες μη κενούς, ενώ οι αντίστοιχοι δείκτες των φύλλων είναι κενοί (άρα κάθε φυλλοπροσανατολισμένο δένδρο είναι γεμάτο).

Στην εργασία αυτή το φυλλοπροσανατολισμένο δένδρο που ζητείται να υλοποιήσετε έχει την επιπρόσθετη ιδιότητα πως όλα τα φύλλα του είναι συνδεδεμένα μεταξύ τους σχηματίζοντας μια ταξινομημένη (ως προς τη χρονοσφραγίδα κάθε εγγραφής) απλά συνδεδεμένη λίστα. Το πρώτο στοιχείο αυτής της λίστας είναι το αριστερότερο φύλλο του φυλλοπροσανατολισμένου δένδρου, ενώ το τελευταίο στοιχείο της λίστας είναι το δεξιότερο φύλλο του δένδρου. Η ιδιότητα αυτή θα πρέπει να διατηρείται κατά την εισαγωγή νέων κλειδιών στο δένδρο. Ένα παράδειγμα φυλλοπροσανατολισμένου δένδρου που έχει την ιδιότητα σύνδεσης των φύλλων του σε λίστα παρουσιάζεται στο Σχήμα 6 (όπου οι δείκτες που αποτελούν τη λίστα των φύλλων παρουσιάζονται με έντονο χρώμα). Στην εργασία αυτή κάθε αναφορά σε φυλλοπροσανατολισμένα δένδρα αναφέρεται σε φυλλοπροσανατολισμένα δένδρα των οποίων τα φύλλα έχουν διασυνδεθεί ώστε να σχηματίζουν λίστα (όπως φαίνεται στο Σχήμα 6). Σε ένα διπλά συνδεδεμένο τέτοιο δένδρο, κάθε κόμβος έχει ένα δείκτη που δείχνει στον πατρικό του κόμβο. Στο τέλος της παρούσας εργασίας, συγκεκριμένα στο Σχήμα 9, εξηγείται ο τρόπος με τον οποίο μπορεί να παραχθεί το δένδρο του Σχήματος 6, δηλαδή παρουσιάζονται διαδοχικά οι εισαγωγές των κόμβων του.



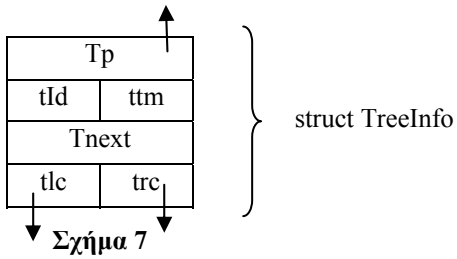
Σχήμα 6

Κάθε ένα από τα φυλλοπροσανατολισμένα δένδρα ενός συνδρομητή είναι **ταξινομημένο ως προς τη χρονοσφραγίδα των εγγραφών** που αποθηκεύονται σε αυτό. Κάθε τέτοια εγγραφή, είναι τύπου TreeInfo και έχει τα εξής πεδία:

- Έναν ακέραιο tId που αποτελεί το αναγνωριστικό της πληροφορίας.
- Έναν ακέραιο ttm που αποθηκεύει τη χρονοσφραγίδα άφιξης της πληροφορίας.
- Έναν δείκτη tlc προς τον αριστερό θυγατρικό κόμβο στο δένδρο.
- Έναν δείκτη trc προς το δεξιό θυγατρικό κόμβο στο δένδρο.
- Έναν δείκτη tp προς τον πατρικό κόμβο στο δένδρο.

- Έναν δείκτη tnext που χρησιμοποιείται αν ο κόμβος είναι φύλλο και δείχνει στο αμέσως επόμενο (προς τα δεξιά) φύλλο στο δένδρο. Έτσι, όλα τα φύλλα του δένδρου συνενώνονται σε μια λίστα με το αριστερότερο φύλλο να είναι το πρώτο στοιχείο της λίστας και το δεξιότερο φύλλο να είναι το τελευταίο στοιχείο της λίστας. Όλοι οι εσωτερικοί κόμβοι του δένδρου κατανάλωσης έχουν το πεδίο next της εγγραφής τους ίσο με NULL.

Η εγγραφή τύπου TreeInfo παρουσιάζεται στο Σχήμα 7.



Σχήμα 7

Περιγραφή Γεγονότων Αρχείου Εισόδου

Το πρόγραμμα θα πρέπει να εκτελείται καλώντας την εντολή

run <m> <p> <input-file>

όπου run είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος, <m> είναι ένας πρώτος αριθμός που καθορίζει το μέγεθος του πίνακα κατακερματισμού των συνδρομητών, <p> είναι ένας πρώτος αριθμός που θα χρησιμοποιήσετε για την σχεδίαση της καθολικής κλάσης των συναρτήσεων κατακερματισμού και <input-file> είναι το όνομα ενός αρχείου εισόδου που περιέχει γεγονότα των μορφών που περιγράφηκαν στο 1ο μέρος της εργασίας.

Το αρχείου εισόδου περιέχει γεγονότα των ακόλουθων μορφών:

- I <itm> <iId> <gId_1> <gId_2> … <gId_k> -1: Γεγονός τύπου Insert_Info το οποίο έχει την ίδια μορφή με εκείνη του 1ου μέρους της προγραμματιστικής εργασίας.

  Τα δένδρα πληροφοριών των ομάδων με τις οποίες συσχετίζεται η πληροφορία <iId> πρέπει να ενημερωθούν (δηλαδή θα πρέπει να προστεθεί σε κάθε ένα από αυτά μια εγγραφή που να περιγράφει την πληροφορία <iId>). Η εγγραφή αυτή θα πρέπει να εισαχθεί στην κατάλληλη θέση σε κάθε ένα από αυτά τα δένδρα ώστε αυτά να παραμείνουν ταξινομημένα ως προς τα αναγνωριστικά των πληροφοριών που αποθηκεύονται σε αυτά.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ίδια πληροφορία (και στην ίδια μορφή) όπως και στο 1ο μέρος της προγραμματιστικής εργασίας.

- S <sTM> <sId> <gId1> <gId2> … <gIdm> -1: Γεγονός τύπου Subscriber_Registration το οποίο σηματοδοτεί την εγγραφή ενός νέου συνδρομητή με αναγνωριστικό <sId> στο σύστημα την χρονική στιγμή <sTM>, όπως και στο 1ο μέρος της εργασίας. Ο νέος συνδρομητής πρέπει να προστεθεί στον πίνακα κατακερματισμού συνδρομητών. Επίσης πρέπει να προστεθεί μια εγγραφή στη λίστα συνδρομών κάθε ομάδας για την οποία ενδιαφέρεται ο νέος συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει αντίστοιχη πληροφορία με εκείνη που τυπωνόταν στο 1ο μέρος της προγραμματιστικής εργασίας.

- **R <tm>:** Γεγονός τύπου Prune το οποίο σηματοδοτεί την εκκίνηση ενέργειας αποστολής των πληροφοριών κάθε ομάδας που έχουν χρονοσφραγίδα μικρότερη της <tm> στους συνδρομητές που είναι εγγεγραμμένοι στην ομάδα αυτή. Ενέργεια αυτή ονομάζεται και ενέργεια «κλαδέματος» (pruning) των δένδρων πληροφοριών που κρατούνται στον εξυπηρέτη. Αυτό το γεγονός μπορεί π.χ., να χρειαστεί να συμβεί επειδή ο εξυπηρετητής (server) αποφασίζει πως ο χώρος που διαθέτει δεν

επαρκεί για την αποθήκευση περισσότερων πληροφοριών, οπότε θα πρέπει να προωθήσει μερικές από τις αποθηκευμένες πληροφορίες στους συνδρομητές (ελευθερώνοντας έτσι χώρο για την αποθήκευση νέων πληροφοριών).

Κατά την εκτέλεση του γεγονότος αυτού, για κάθε ομάδα G[k] πραγματοποιούνται οι εξής ενέργειες. Γίνεται διάσχιση του δένδρου πληροφοριών της G[k] και για κάθε εγγραφή I η οποία έχει χρονοσφραγίδα μικρότερη από ή ίση με <tm>, η εγγραφή I διαγράφεται από το δένδρο. Για κάθε εγγραφή S της λίστας συνδρομών της G[k], η I εισάγεται στο δένδρο κατανάλωσης του S που αντιστοιχεί στην ομάδα G[k] (δηλαδή στο δένδρο κατανάλωσης που δεικτοδοτείται από τον δείκτη tgp[k] του S).

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

R DONE

GROUPID = <gId1>, INFOLIST = <iId_1^1, iId_2^1, …, iId_{r1}^1>, SUBLIST = < sId_1^1, sId_2^1, …, sId_{n1}^1>

GROUPID = <gId2>, INFOLIST = <iId_1^2, iId_2^2, …, iId_{r2}^2>, SUBLIST = < sId_1^2, sId_2^2, …, sId_{n2}^2>

…

GROUPID = <gIdMG>, INFOLIST= <iId_1^{MG}, iId_2^{MG}, …, iId_{rMG}^{MG}>, SUBLIST= < sId_1^{MG}, sId_2^{MG},…, sId_{nMG}^{MG}>

SUBSCRIBERID = <sId1>, GROUPLIST =

<gId_1^1> , TREELIST= <tId_1^1(<gId_1^1>), tId_2^1(<gId_1^1>), …, tId_{r1}^1(<gId_1^1>)>,

…

<gId_{v1}^1>, TREELIST = <tId_1^1(<gId_{v1}^1>), tId_2^1(<gId_{v1}^1>), …, tId_{rv1}^1(<gId_{v1}^1>)>,

SUBSCRIBERID = <sId2>, GROUPLIST =

<gId_1^2>, TREELIST = < tId_1^2(<gId_1^2>), tId_2^2(<gId_1^2>), …, tId_{r2}^2(<gId_1^2>)>

…

<gId_{v2}^2>, TREELIST = < tId_1^2(<gId_{v2}^2>), tId_2^2(<gId_{v2}^2>), …, tId_{rv2}^2(<gId_{v2}^2>)>

…

SUBSCRIBERID = <sIdq>, GROUPLIST =

<gId_1^q> , TREELIST = < tId_1^q(<gId_1^q>), tId_2(<gId_1^q>), …, tId_{rq}^q(<gId_1^q>)>

…

<gId_{vq}^q>, TREELIST = < tId_1^q(<gId_{vq}^q>), tId_2^q(<gId_{vq}^q>), …, tId_{rvq}^q(<gId_{vq}^q>)>

όπου <gIdi>, $1 \leq i \leq MG$, είναι το αναγνωριστικό της ομάδας G[<gIdi>], <iId_1^i, iId_2^i, …, iId_{ri}^i> είναι τα αναγνωριστικά των εγγραφών πληροφοριών που είναι αποθηκευμένα στο δένδρο πληροφοριών της ομάδας και <sId_1^i, sId_2^i, …, sId_{ni}^i> είναι η λίστα συνδρομών της. Επιπρόσθετα, <sIdj> είναι το αναγνωριστικό του j-οστού στοιχείου του πίνακα κατακερματισμού των συνδρομητών, <gId_1^j, …, gId_{vj}^j> είναι τα αναγνωριστικά των ομάδων για τις οποίες ενδιαφέρεται ο συνδρομητής με αναγνωριστικό <sIdj> και για κάθε τέτοια ομάδα <gId_f^j>, <tId_1^j(<gId_f^j>), …, tId_{rvf}^j(<gId_f^j>)> είναι οι πληροφορίες που περιέχονται στο δένδρο κατανάλωσης του συνδρομητή με αναγνωριστικό <sIdj> που δεικτοδοτείται από τον δείκτη tgp[<gId_f^j>].

- C <sId>: Κατά το γεγονός τύπου Consume, κάθε συνδρομητής καταναλώνει **τις νέες πληροφορίες** που περιέχονται στα δένδρα πληροφοριών που δεικτοδοτούνται από τους tgp δείκτες που περιέχει η εγγραφή του συνδρομητή στον πίνακα κατακερματισμού συνδρομητών. Συγκεκριμένα, για κάθε ομάδα k για την οποία ενδιαφέρεται ο συνδρομητής, καταναλώνονται όλες οι πληροφορίες που δημοσιεύθηκαν μετά από την τελευταία εκτέλεση από το συνδρομητή της λειτουργίας κατανάλωσης για την ομάδα αυτή. Αυτό γίνεται με την ανάγνωση των στοιχείων της λίστας των φύλλων του δένδρου κατανάλωσης tgp[k] ξεκινώντας από την εγγραφή της λίστας αυτής στην οποία δείχνει ο

sgp[k] (δηλαδή από την τελευταία εγγραφή που ο συνδρομητής έχει καταναλώσει σε αυτή την ομάδα) μέχρι και την τελευταία εγγραφή της λίστας αυτής (δηλαδή μέχρι και το δεξιότερο φύλλο του δένδρου κατανάλωσης tgp[k]). Στο τέλος της διαδικασίας κατανάλωσης, ο δείκτης sgp[k] ενημερώνεται ώστε να δείχνει στο δεξιότερο φύλλο του δένδρου κατανάλωσης tgp[k]. Είναι σημαντικό πως ο δείκτης sgp[k] θα έχει την τιμή NULL εάν δεν έχει προηγηθεί γεγονός τύπου Prune ή αν δεν υπάρχουν δημοσιευμένες πληροφορίες που να αφορούν την ομάδα G[k] κατά την εκτέλεση του τελευταίου γεγονότος Prune.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

C <sId> DONE

GROUPID = <gId1>, TREELIST = <tId_1^1, tId_2^1, …, tId_{r1}^1>, NEWSGP = <tId1>

GROUPID = <gId2>, TREELIST = <tId_1^2, tId_2^2, …, tId_{r2}^2>, NEWSGP = <tId2>

…

GROUPID = <gIdk>, TREELIST = <tId_1^k, tId_2^k, …, tId_{rk}^k>, NEWSGP = <tIdk>

όπου <sId> είναι το αναγνωριστικό του συνδρομητή, <gIdi>, όπου $1 \leq i \leq k$, είναι τα αναγνωριστικά των ομάδων στις οποίες έχει εγγραφεί ο συνδρομητής και <tId_1^i, tId_2^i, …, tId_{ri}^i> είναι τα αναγνωριστικά των εγγραφών πληροφοριών που καταναλώθηκαν από εκείνο το δένδρο κατανάλωσης του συνδρομητή που αντιστοιχεί στην ομάδα <gIdi>. Τέλος, tIdi είναι το αναγνωριστικό της τελευταίας πληροφορίας που καταναλώθηκε από τη λίστα φύλλων του δένδρου κατανάλωσης tgp[<gIdi>] του συνδρομητή με αναγνωριστικό <sId>(στην οποία θα πρέπει να δείχνει ο δείκτης sgp[<gIdi>] του συνδρομητή <sId> όταν η λειτουργία κατανάλωσης τερματίσει την εκτέλεσή της).

- D <sId>: Γεγονός τύπου Delete_Subscriber. Το γεγονός αυτό σηματοδοτεί την αποχώρηση του συνδρομητή με αναγνωριστικό <sId> από το σύστημα. Η εγγραφή που αναφέρεται στον συνδρομητή με αναγνωριστικό <sId> θα πρέπει να διαγραφεί από τη λίστα συνδρομητών. Το ίδιο ισχύει για όλες τις δομές δεδομένων που διατηρεί αυτός ο συνδρομητής (όπως π.χ., τα δένδρα κατανάλωσης που αντιστοιχούν στις ομάδες στις οποίες είναι εγγεγραμμένος ο συνδρομητής). Επίσης, θα πρέπει να διαγραφεί η εγγραφή που αναφέρεται στη συνδρομή του εν λόγω συνδρομητή από κάθε ομάδα στην οποία έχει εγγραφεί ο συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ίδια πληροφορία όπως και στο $1^{ο}$ μέρος της προγραμματιστικής εργασίας.

- P: Γεγονός τύπου Print που σηματοδοτεί την εκτύπωση των δομών δεδομένων του συστήματος. Συγκεκριμένα, για κάθε μία από τις ομάδες, θα πρέπει να εκτυπώνεται η λίστα των πληροφοριών της και των συνδρομών της. Επίσης, θα πρέπει να εκτυπώνεται η λίστα με τους συνδρομητές και για κάθε έναν από αυτούς να εκτυπώνονται οι συνδρομές του, δηλαδή εκείνα τα στοιχεία του πίνακα sgp που περιγράφουν τις ομάδες για τις οποίες ενδιαφέρεται ο συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  P DONE

  GROUPID = <gId1>, INFOLIST = <iId_1^1, iId_2^1, …, iId_{r1}^1>, SUBLIST = < iId_1^1, iId_2^1, …, iId_{n1}^1>

  GROUPID = <gId2>, INFOLIST = <iId_1^2, iId_2^2, …, iId_{r2}^2>, SUBLIST = < iId_1^2, iId_2^2, …, iId_{n2}^2>

  …

  GROUPID = <gIdMG>, INFOLIST= <iId_1^{MG}, iId_2^{MG}, …, iId_{rk}^{MG}>, SUBLIST= < iId_1^{MG}, iId_2^{MG},…, iId_{nk}^{MG}>

  SUBSCRIBERLIST = <sId1, sId2, …, sIdm>

SUBSCRIBERID = <sId1>, GROUPLIST =

<gId_1^1> , TREELIST= <tId_1^1(<gId_1^1>), tId_2^1(<gId_1^1>), …, tId_{r1}^1(<gId_1^1>)>,

…

<gId_{v1}^1>, TREELIST = <tId_1^1(<gId_{v1}^1>), tId_2^1(<gId_{v1}^1>), …, tId_{rv1}^1(<gId_{v1}^1>)>,

SUBSCRIBERID = <sId2>, GROUPLIST =

<gId_1^2>, TREELIST = < tId_1^2(<gId_1^2>), tId_2^2(<gId_1^2>), …, tId_{r1}^2(<gId_1^2>)>

…

<gId_{v2}^2>, TREELIST = < tId_1^2(<gId_{v2}^2>), tId_2^2(<gId_{v2}^2>), …, tId_{rv2}^2(<gId_{v2}^2>)>

…

SUBSCRIBERID = <sIdq>, GROUPLIST =

<gId_1^q> , TREELIST = < tId_1^q(<gId_1^q>), tId_2^q(<gId_1^q>), …, tId_{rq}^q(<gId_1^q>)>

…

<gId_{vq}^q>, TREELIST = < tId_1^q(<gId_{vq}^q>), tId_2^q(<gId_{vq}^q>), …, tId_{rvq}^q(<gId_{vq}^q>)>

NO_GROUPS = <number of groups>, NO_SUBSCRIBERS = <number of subscribers>

όπου <gIdi>, $1 \le i \le MG$, είναι το αναγνωριστικό της ομάδας G[<gIdi>], <iId_1^i, iId_2^i, …, iId_{ri}^i> είναι η λίστα πληροφοριών της και < iId_1^i, iId_2^i, …, iId_{ni}^i> είναι η λίστα συνδρομών της. Επιπρόσθετα, <sIdj> είναι το αναγνωριστικό του j-οστού στοιχείου της λίστας συνδρομητών, <gId_1^j, …, gId_{vj}^j> είναι τα για κάθε τέτοια ομάδα <gId_l^j>, <tId_1^j(<gId_l^j>), …, tId_{rvl}^j(<gId_l^j>)> είναι οι πληροφορίες που περιέχονται στο δένδρο κατανάλωσης του συνδρομητή με αναγνωριστικό <sIdj> που δεικτοδοτείται από τον δείκτη tgp[<gId_l^j>]. Τέλος, <number of groups> είναι το συνολικό πλήθος των έγκυρων ομάδων (δηλαδή αυτών στις οποίες έχουν δημοσιευθεί δεδομένα) και <number of subscribers> είναι το συνολικό πλήθος των συνδρομητών στο σύστημα.

Το πρόγραμμα που θα δημιουργηθεί πρέπει να διαβάζει το αρχείο εισόδου και να εκτελεί με τη σειρά όλα τα γεγονότα που περιγράφονται σε αυτό.

Οδηγίες και Συμβουλές για την Ομαλή Διεκπεραίωση της Εργασίας

Όπως και με τα προηγούμενα μέρη της εργασίας, το μέρος αυτό 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 συνοπτική ακολουθία βημάτων που θα μπορούσε να ακολουθηθεί για την ομαλή διεκπεραίωση της εργασίας.

Βήμα 1: Ξεκινήστε υλοποιώντας, σε ένα νέο αρχείο, ένα ταξινομημένο, διπλά συνδεδεμένο, δυαδικό δένδρο. Γράψτε κώδικα για τις διαδικασίες BST_Insert, BST_Delete και BST_LookUp που θα επιτρέψουν να πραγματοποιείτε εισαγωγές, διαγραφές και αναζητήσεις, αντίστοιχα, σε ένα ταξινομημένο, διπλά συνδεδεμένο ταξινομημένο δένδρο. Οι διαδικασίες αυτές θα πρέπει να έχουν χρονική πολυπλοκότητα O(h), όπου h είναι το ύψος του δένδρου. Δημιουργήστε μια διαδικασία BST_Print() για την εκτύπωση των στοιχείων του δένδρου η οποία θα σας βοηθήσει να ελέγξετε την ορθότητα των διαδικασιών που περιγράφονται παραπάνω.

Δημιουργήστε τον κώδικα σας σε ένα νέο αρχείο και φτιάξτε μια δική σας main() για να ελέγξετε ότι ο κώδικας που υλοποιεί τις λειτουργίες ενός ταξινομημένου, διπλά συνδεδεμένου, δυαδικού δένδρου λειτουργεί σωστά. Όταν είστε σίγουροι για αυτό συνεχίστε με το επόμενο βήμα. Ο κώδικας που έχετε δημιουργήσει στο 1ο μέρος της προγραμματιστικής εργασίας δεν χρησιμοποιείται καθόλου σε αυτό το βήμα (εδώ απλά υλοποιείτε ένα ταξινομημένο, διπλά-συνδεδεμένο, δυαδικό δένδρο του οποίου κάθε στοιχείο είναι τύπου struct Info).

Βήμα 2: Τροποποιείστε **ένα αντίγραφο** του μέρους 1 της προγραμματιστικής εργασίας, έτσι ώστε η λίστα πληροφοριών κάθε ομάδας να αντικατασταθεί από ένα ταξινομημένο, διπλά συνδεδεμένο, δυαδικό δένδρο πληροφοριών (χρησιμοποιώντας τον κώδικα του βήματος 1). Στο βήμα αυτό απενεργοποιήστε (π.χ., βάλτε σε σχόλια ή αφαιρέστε) τον κώδικα που έχετε γράψει για το γεγονός τύπου Consume στο 1ο μέρος της προγραμματιστικής εργασίας. Διορθώστε κατάλληλα τον κώδικα που αντιστοιχεί στο γεγονός τύπου I, ώστε αυτός να λειτουργεί σωστά δεδομένου ότι η λίστα πληροφοριών κάθε ομάδας έχει αντικατασταθεί από ένα δένδρο πληροφοριών.

Βήμα 3: Δημιουργείστε ένα αντίγραφο του κώδικα που δημιουργήσατε στο Βήμα 1 και τροποποιήστε τον κώδικα αυτόν κατάλληλα ώστε να εκτελείται σωστά πάνω σε φυλλοπροσανατολισμένα, ταξινομημένα, διπλά συνδεδεμένα, δυαδικά δένδρα. Οι κόμβοι καθενός από τα δένδρα αυτά πρέπει να είναι τύπου struct TreeInfo και το δένδρο πρέπει να είναι ταξινομημένο ως προς το πεδίο ttm των κόμβων του. Γράψτε κώδικα για τις διαδικασίες LO_BST_Insert, και LO_BST_LookUp που θα επιτρέψουν να πραγματοποιείτε εισαγωγές και αναζητήσεις, αντίστοιχα, σε ένα φυλλοπροσανατολισμένο, ταξινομημένο, διπλά συνδεδεμένο, δυαδικό δένδρο.

Είναι αξιοσημείωτο πως προκειμένου να διατηρείται σωστά η ιδιότητα διασύνδεσης των φύλλων τέτοιων δένδρων χρειάζεστε μια συνάρτηση η οποία θα παίρνει ως παράμετρο έναν δείκτη p σε ένα κόμβο φύλλο του δένδρου και θα επιστρέφει έναν δείκτη στο αμέσως προηγούμενο φύλλο του δένδρου (δηλαδή στο φύλλο του δένδρου με το μεγαλύτερο κλειδί μεταξύ εκείνων των φύλλων που έχουν μικρότερα κλειδιά από εκείνο του κόμβου p). Η συνάρτηση αυτή θα πρέπει να υλοποιηθεί έτσι ώστε η χρονική της πολυπλοκότητα να είναι O(h), όπου h είναι το ύψος του φυλλοπροσανατολισμένου δένδρου. Την ίδια χρονική πολυπλοκότητα θα πρέπει να έχουν οι LO_BST_Insert() και LO_BST_LookUp().

Ο κώδικας που προκύπτει από το βήμα 2 δεν χρειάζεται στο βήμα αυτό. Μόνο όταν είστε βέβαιοι πως οι λειτουργίες LO_BST_Insert() και LO_BST_LookUp() λειτουργούν σωστά όταν εκτελούνται σε φυλλοπροσανατολισμένα δένδρα, προχωρήστε στο επόμενο βήμα.

Βήμα 4: Ενσωματώστε τον κώδικα του Βήματος 3 στον κώδικα που προέκυψε από το Βήμα 2 και δημιουργήστε μια πρώτη έκδοση της προγραμματιστικής εργασίας στην οποία θα έχετε υλοποιήσει το γεγονός τύπου Prune αλλά χωρίς να έχετε αντικαταστήσει τη λίστα συνδρομητών από πίνακα κατακερματισμού. Διορθώστε κατάλληλα τον κώδικα του γεγονότος τύπου Consume (που είχατε δημιουργήσει στο 1^{ο} μέρος) ώστε να λειτουργεί με τον τρόπο που περιγράφεται στο τρέχον μέρος της εργασίας. Διορθώστε κατάλληλα τα γεγονότα τύπου S, D και P ώστε να λειτουργούν σωστά με τα νέα δεδομένα.

Βήμα 5: Σε ένα νέο αρχείο γράψτε κώδικα ο οποίος θα υλοποιεί έναν πίνακα κατακερματισμού που χρησιμοποιεί τη μέθοδο επίλυσης συγκρούσεων των ξεχωριστών αλυσίδων. Τα στοιχεία των αλυσίδων του πίνακα κατακερματισμού πρέπει να είναι τύπου struct SubInfo. Υλοποιήστε μια συνάρτηση Universal_Hash_Function() η οποία θα υλοποιεί την καθολική κλάση συναρτήσεων προγραμματισμού. Υλοποιήστε τις συναρτήσεις HT_Insert(), HT_Delete() και HT_LookUp(), οι οποίες πραγματοποιούν εισαγωγές, διαγραφές και αναζητήσεις, αντίστοιχα, στον πίνακα κατακερματισμού. Γράψτε κώδικα ο οποίος θα εκτυπώνει όλα τα δεδομένα του πίνακα κατακερματισμού (σε μορφή κοντινή αυτής που ζητείται στο τρέχον μέρος της εργασίας). Μόνο όταν σιγουρευτείτε πως ο κώδικας που ετοιμάσατε στο βήμα αυτό είναι σωστός προχωρήστε στο επόμενο βήμα.

Βήμα 6: Συνδυάστε τον κώδικα του Βήματος 5 με εκείνον του βήματος 4 για να πάρετε την τελική μορφή της εργασίας.

Βήμα 7: Ελέγξετε την ορθότητα του κώδικα που δημιουργήσατε εκτελώντας τον κώδικά σας σε όλα τα αρχεία εκτέλεσης που θα σας παρέχουν οι βοηθοί του μαθήματος. Επιπρόσθετα, δημιουργήστε τα δικά σας αρχεία γεγονότων για να ελέγξετε με περισσότερη ακρίβεια την ορθότητα του κώδικά σας.

Δομές Δεδομένων

Στο Σχήμα 8 παρουσιάζονται οι δομές σε C που πρέπει να χρησιμοποιηθούν για την υλοποίηση της παρούσας εργασίας.

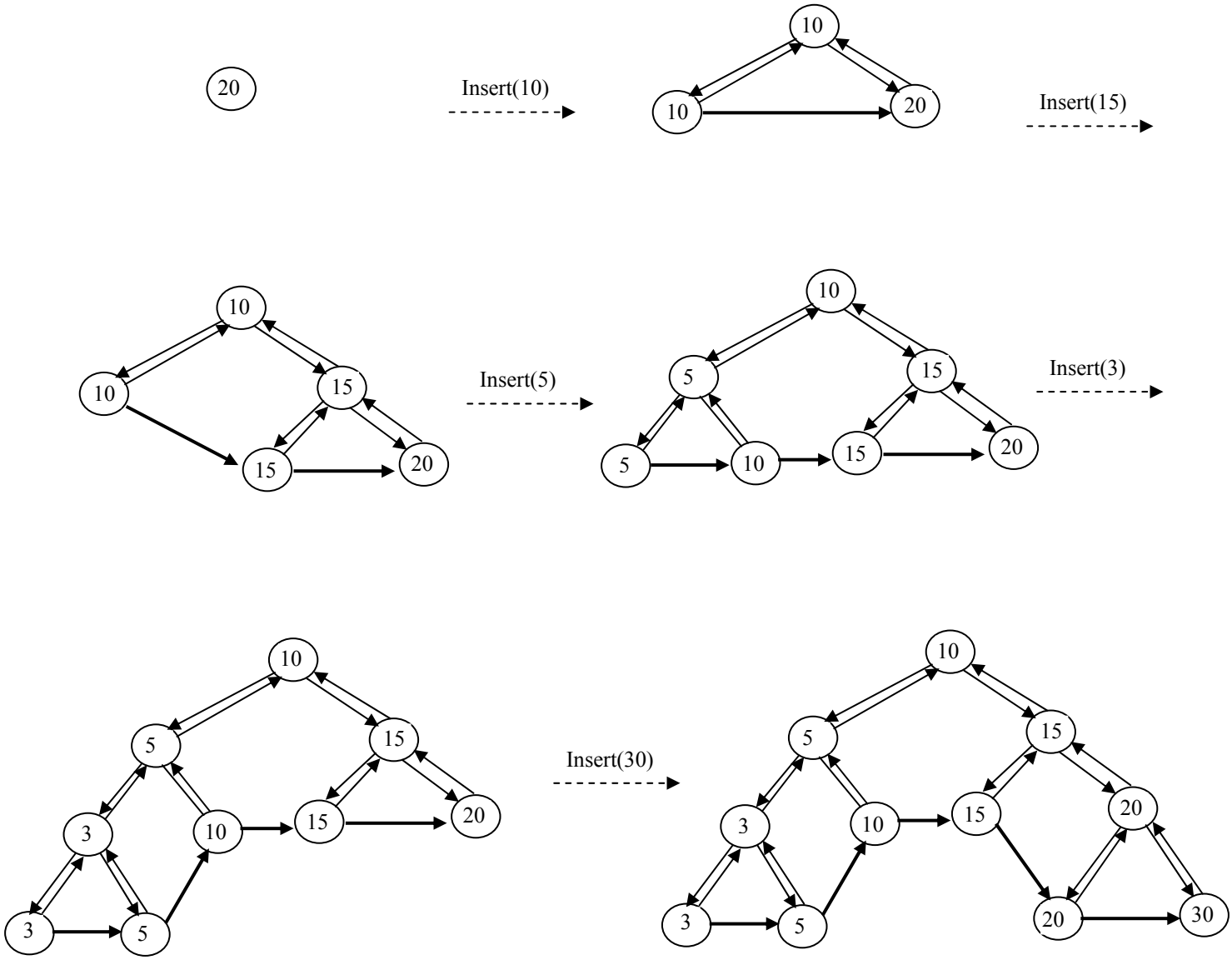
```
struct Info {                       // Εγγραφή δένδρου πληροφοριών
        int iId;                    // Αναγνωριστικό πληροφορίας
        int itm;                    // Χρονοσφραγίδα δημοσίευσης πληροφορίας
        int igp[MG];                // Αυτό πρέπει να είναι ο πίνακας igp MG θέσεων
        struct Info *ilc;           // Δείκτης στον αριστερό θυγατρικό κόμβο
        struct Info *irc;           // Δείκτης στο δεξιό θυγατρικό κόμβο
        struct Info *ip;            // Δείκτης στον πατρικό κόμβο
};

struct Subscription {               // Εγγραφή λίστας συνδρομών
        int sId;                    // Αναγνωριστικό Συνδρομητή
        struct Subscription *snext; // Επόμενη εγγραφή λίστας συνδρομών
};
```

```
struct Group {                          // Εγγραφή  πίνακα ομάδων
        int gId;                        // Αναγνωριστικό ομάδας
        struct Subscription *gsub;      // Δείκτης στο πρώτο στοιχείο λίστας συνδρομών
        struct Info *gr;                // Δείκτης στη ρίζα του δένδρου πληροφοριών
};


struct SubInfo {                        // Εγγραφή πίνακα κατακερματισμού  συνδρομητών
        int sId;                        // Αναγνωριστικό συνδρομητή
        int stm;                        // Χρονοσφραγίδα δημιουργίας συνδρομητή
        struct TreeInfo *tgp[MG];       // Πίνακας tgp MG θέσεων
        struct TreeInfo *sgp[MG];       // Πίνακας sqp MG θέσεων
        struct SubInfo *snext;          // Δείκτης στο επόμενο στοιχείο της αλυσίδας του πίνακα κατακερματισμού συνδρομητών
};

Struct TreeInfo {               // Εγγραφή δένδρου κατανάλωσης
        int tId;                // Αναγνωριστικό πληροφορίας
        int ttm;                // Χρονοσφραγίδα δημοσίευσης πληροφορίας
        struct TreeInfo *tlc;   // Δείκτης στον αριστερό θυγατρικό κόμβο
        struct TreeInfo *trc;   // Δείκτης στον δεξιό θυγατρικό κόμβο
        struct TreeInfo *tp;    // Δείκτης στον πατρικό κόμβο
        struct TreeInfo *next;  // Δείκτης στο επόμενο φύλλο προς τα δεξιά (NULL αν η εγγραφή δεν αντιστοιχεί σε κόμβο φύλλο)
};

// Οι παρακάτω δομές θα ήταν προτιμότερο αν δηλωθούν όχι ως global  μεταβλητές
// αλλά ως τοπικές μεταβλητές της main και συνίσταται στους φοιτητές που έχουν
// πολύ καλές γνώσεις προγραμματισμού να ακολουθήσουν αυτή την τακτική
// παρότι η δυσκολία υλοποίησης αυξάνει με αυτό τον τρόπο.

struct Group G[MG];             // Πίνακας ομάδωνG MG θέσεων
struct SubInfo *T[MG];          // Πίνακας κατακερματισμού συνδρομητώνMG θέσεων
```

Σχήμα 8

Παράδειγμα δημιουργίας φυλλοπροσανατολισμένου δένδρου

Στο Σχήμα 9 εξηγείται ο τρόπος με τον οποίο μπορεί να παραχθεί το δένδρο του Σχήματος 6. Συγκεκριμένα παρουσιάζονται διαδοχικά οι εισαγωγές των κόμβων του.


20
Insert(10)
10
10
20
Insert(15)
10
10
15
15
20
Insert(5)
10
5
15
5
10
15
20
Insert(3)
10
5
15
3
10
15
20
3
5
Insert(30)
10
5
15
3
10
15
20
3
5
20
30

Σχήμα 9