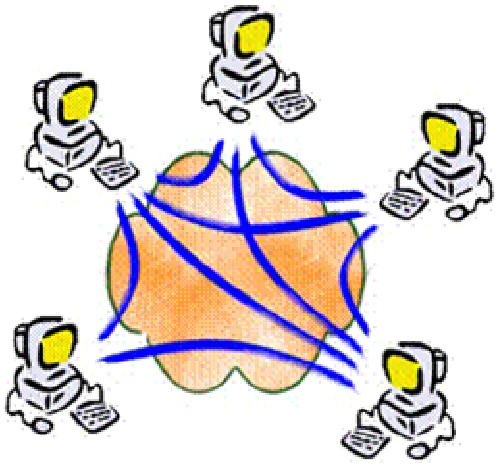
HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2008-09
Παναγιώτα Φατούρου

Προγραμματιστική Εργασία
1ο Μέρος

Ημερομηνία Παράδοσης: Παρασκευή, 7 Νοεμβρίου 2008, ώρα 23:59.
Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα submit. Πληροφορίες για το πώς λειτουργεί το submit θα σταλούν στη λίστα του μαθήματος από τους βοηθούς τις επόμενες μέρες.

Ένα διομότιμο (Peer-to-Peer ή P2P) σύστημα είναι ένα κατανεμημένο σύστημα (Σχήμα 1) που αποτελείται από κόμβους που είναι ομότιμοι (δηλαδή κόμβους που έχουν την ίδια λειτουργικότητα).

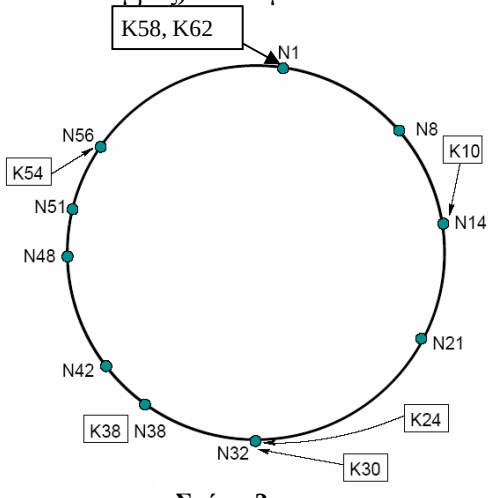


Σχήμα 1: Ένα Διομότιμο Σύστημα

Θεωρούμε ένα διομότιμο σύστημα στο οποίο το πλήθος των κόμβων που συμμετέχουν κάθε χρονική στιγμή είναι αυθαίρετο αλλά όχι μεγαλύτερο από 2^m, όπου m είναι κάποιος ακέραιος. Θεωρούμε πως κάθε κόμβος που συμμετέχει στο σύστημα έχει ένα **μοναδικό** αναγνωριστικό, δηλαδή χαρακτηρίζεται από κάποιον ακέραιο στο διάστημα από 0 έως 2^m-1. Έτσι, όλοι οι κόμβοι μπορούν να διαταχθούν, σύμφωνα με το αναγνωριστικό τους, πάνω σε έναν λογικό δακτύλιο που αναπαριστά το σύνολο τιμών {0, …, 2^m-1}. Ένα διομότιμο σύστημα εξελίσσεται δυναμικά, δηλαδή υποστηρίζει την εισαγωγή και την αποχώρηση κόμβων.

Στο Σχήμα 2 παρουσιάζεται ένα διομότιμο σύστημα για το οποίο ισχύει ότι m=6 (δηλαδή ο μέγιστος αριθμός κόμβων που μπορούν να συμμετέχουν στο σύστημα είναι 2^6 = 64). Την τρέχουσα χρονική στιγμή μόνο 10 κόμβοι συμμετέχουν στο σύστημα, οι οποίοι έχουν αναγνωριστικά 1, 8, 14, 21, 32, 38, 42, 48, 51 και 56. Οι κόμβοι αυτοί, στο σχήμα εμφανίζονται ως N1, N8, N14, N21, N32, N38, N42, N48, N51 και N56, αντίστοιχα.

Μια από τις σημαντικότερες εφαρμογές ενός συστήματος P2P είναι η κατανεμημένη αποθήκευση και διαχείριση αρχείων. Θεωρούμε πως κάθε αρχείο έχει (όπως και κάθε κόμβος) ένα μοναδικό αναγνωριστικό στο σύνολο {0,…,2^m-1}. Έστω N_{min} και N_{max} οι κόμβοι με το μικρότερο και το μεγαλύτερο, αντίστοιχα, αναγνωριστικό στο δακτύλιο. Ο κανόνας σύμφωνα με τον οποίο αποθηκεύονται τα αρχεία στους κόμβους του συστήματος είναι ο ακόλουθος:



Σχήμα 2

«(α) Κάθε κόμβος ≠ N_{min} αποθηκεύει τα αρχεία των οποίων τα αναγνωριστικά είναι μεταξύ του αναγνωριστικού του κόμβου και του αναγνωριστικού του προηγούμενου κόμβου στο δακτύλιο και (β) ο Nmin αποθηκεύει κάθε αρχείο με αναγνωριστικό K τέτοιο ώστε $N_{max} \leq K \leq 2^m$-1 ή $0 \leq K \leq N_{min}$». *(1)*

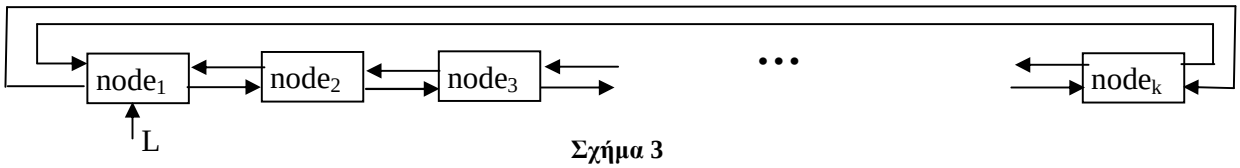
Για παράδειγμα, στο Σχήμα 2, έχουν αποθηκευτεί 7 αρχεία στο σύστημα, τα οποία έχουν αναγνωριστικά 10, 24, 30, 38, 54, 58 και

62. Το αρχείο με αναγνωριστικό 10 αποθηκεύεται στον κόμβο με αναγνωριστικό 14 (αφού το 10 είναι μεταξύ του αναγνωριστικού 8 που έχει ο προηγούμενος κόμβος του 14 στο σύστημα και του 14), τα αρχεία με αναγνωριστικά 24 και 30 αποθηκεύονται στον κόμβο με αναγνωριστικό 32, το αρχείο με αναγνωριστικό 38 αποθηκεύεται στον κόμβο με αναγνωριστικό 38, εκείνο με αναγνωριστικό 54 αποθηκεύεται στον κόμβο με αναγνωριστικό 56, ενώ τα αρχεία με αναγνωριστικά 58 και 62 αποθηκεύονται στον κόμβο με αναγνωριστικό 1.

Αν στο σύστημα εισαχθεί κόμβος με αναγνωριστικό 26, ο κόμβος 32 θα συνεχίσει να αποθηκεύει μόνο το αρχείο με αναγνωριστικό 30 ενώ εκείνο με αναγνωριστικό 24 θα αποθηκευθεί στο νέο κόμβο (δηλαδή σε εκείνο με αναγνωριστικό 26) ώστε να ισχύει η ιδιότητα (1).

Ζητείται να υλοποιηθεί ένα πρόγραμμα που περιγράφει (προσομοιώνει) τη λειτουργία ενός διομότιμου συστήματος. Πιο συγκεκριμένα, το πρόγραμμα θα πρέπει να υλοποιεί τις ακόλουθες δομές δεδομένων:

Μία **ταξινομημένη κυκλική διπλά συνδεδεμένη** λίστα στην οποία αποθηκεύονται πληροφορίες για τους κόμβους που συμμετέχουν κάθε χρονική στιγμή στο σύστημα. Η λίστα αυτή ονομάζεται λίστα κόμβων (NodeList).



Σχήμα 3

Για κάθε κόμβο n διατηρείται η ακόλουθη πληροφορία:

- id: το αναγνωριστικό του κόμβου που είναι ένας ακέραιος.
- 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.
- next: ένας δείκτης στο επόμενο στοιχείο της λίστας των κόμβων.
- prev: ένας δείκτης στο προηγούμενο στοιχείο της λίστας των κόμβων.

Η μορφή των δομών δεδομένων που χρησιμοποιεί το πρόγραμμα φαίνεται στο Σχήμα 4.

Το πρόγραμμα που θα δημιουργηθεί θα πρέπει να εκτελείται καλώντας την ακόλουθη εντολή:

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 ακόλουθων μορφών:

- J <id>: Γεγονός τύπου Join το οποίο σηματοδοτεί την εισαγωγή ενός κόμβου στο σύστημα με αναγνωριστικό <id>. Ο κόμβος πρέπει να τοποθετηθεί στη σωστή θέση της λίστας των κόμβων. Επίσης, η λίστα των αρχείων του πρέπει να αρχικοποιηθεί κατάλληλα. Πιο συγκεκριμένα, τα αρχεία που βρίσκονται στη λίστα αρχείων του επομένου κόμβου στο δακτύλιο μοιράζονται ώστε να εξακολουθήσει να ισχύει η συνθήκη (1) (δηλαδή κάθε κόμβος να έχει στη λίστα των αρχείων του εκείνα τα αρχεία για τα οποία ισχύει πως το αναγνωριστικό τους είναι μεταξύ του αναγνωριστικού του κόμβου και του αναγνωριστικού του προηγουμένου του κόμβου στο δακτύλιο). Για παράδειγμα, έστω ο κόμβος με αναγνωριστικό 50 και έστω ότι ο προηγούμενός του στη λίστα των κόμβων είναι ο 20. Έστω ότι ο 50 έχει στη λίστα των αρχείων του τα αρχεία με αναγνωριστικά 25, 33, 36, 41, 46, 49. Αν στο σύστημα εισαχθεί ο κόμβος με αναγνωριστικό 40, ο 40 είναι πλέον ο προηγούμενος του 50 και τα αρχεία με αναγνωριστικά 25, 33 και 36 πρέπει να μεταφερθούν από τη λίστα αρχείων του 50

στη λίστα αρχείων του 40. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

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

όπου <pid> και <sid> είναι τα αναγνωριστικά του προηγούμενου και του επόμενου, αντίστοιχα, του προς εισαγωγή κόμβου στη λίστα των κόμβων και fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που έχουν αποθηκευτεί στη λίστα των αρχείων του προς εισαγωγή κόμβου.

- L <id>: Γεγονός τύπου Leave το οποίο σηματοδοτεί την αποχώρηση του κόμβου με αναγνωριστικό <id> από το σύστημα. Ο κόμβος θα πρέπει να παραχωρήσει όλα τα αρχεία του στον επόμενό του κόμβο στη λίστα των κόμβων. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  L <id> DONE, PRED = <pid>, SUCC=<sid>, DOCLISTSUCC = <fid1,fid2,…,fidr>

  όπου <pid> και <sid> είναι τα αναγνωριστικά του προηγούμενου και του επόμενου, αντίστοιχα, στη λίστα των κόμβων, του κόμβου με αναγνωριστικό <id> που απεχώρησε από το σύστημα, ενώ fid1, fid2, …, fidr είναι τα αναγνωριστικά των αρχείων που είναι αποθηκευμένα στη λίστα των αρχείων του επομένου του κόμβου <id> που απεχώρησε από το σύστημα. Η λίστα αρχείων αυτή θα πρέπει να περιέχει και τα αρχεία του κόμβου <id>.

- I <id> <tp>: Γεγονός τύπου Insert το οποίο σηματοδοτεί την εισαγωγή ενός αρχείου με αναγνωριστικό <id> τύπου <tp> στο σύστημα. Το αρχείο θα πρέπει να τοποθετηθεί, βάσει του αναγνωριστικού του, στη λίστα των αρχείων του κατάλληλου κόμβου. Οι δυνατοί τύποι αρχείων είναι οι ακόλουθοι:

  - 1: αρχείο τύπου ascii
  - 2: μουσικό αρχείο
  - 3: αρχείο πολυμέσων (π.χ. video)

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

  I <id> <tp> DONE, NODEID = <nid>

  όπου <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τη λίστα αρχείων αποθηκεύτηκε το αρχείο.

- F <id>: Γεγονός τύπου Find το οποίο σηματοδοτεί την αναζήτηση ενός αρχείου με αναγνωριστικό <id> στο σύστημα. Το αρχείο θα πρέπει να ευρεθεί και να τυπωθούν ο κόμβος στον οποίο είναι αποθηκευμένο, καθώς και ο τύπος του αρχείου. Πιο συγκεκριμένα, θα πρέπει να τυπωθεί η ακόλουθη πληροφορία:

  F <id> DONE, TYPE = <tp>, NODEID = <nid>

  όπου <tp> είναι ο τύπος του αρχείου και <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τη λίστα αρχείων βρέθηκε το αρχείο.

- D <id>: Γεγονός τύπου Delete το οποίο σηματοδοτεί τη διαγραφή του αρχείου με αναγνωριστικό <id> από το σύστημα. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  D <id> DONE, NODEID = <nid>

  όπου <nid> είναι το αναγνωριστικό του κόμβου στου οποίου τη λίστα αρχείων ήταν αποθηκευμένο το αρχείο πριν τη διαγραφή του.

- P: Γεγονός τύπου Print που σηματοδοτεί την εκτύπωση της λίστας των κόμβων και για κάθε έναν από αυτούς την εκτύπωση της λίστας των αρχείων του. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  P DONE

  NODEID = <id1>, DOCLIST = <fd_1^1, fd_2^1, …, fd_{r1}^1>

  NODEID = <id2>, DOCLIST = <fd_1^2, fd_2^2, …, fd_{r2}^2>

  …

  NODEID = <idk>, DOCLIST = <fd_1^k, fd_2^k, …, fd_{rk}^k>

  NO_NODES = <number of nodes> No_FILES = <number of files>

  όπου <number of nodes> και <number of files> είναι το συνολικό πλήθος κόμβων και αρχείων, αντίστοιχα, στο σύστημα, για κάθε j, 1≤ j ≤ k, <idj> είναι το αναγνωριστικό του τρέχοντος προς εκτύπωση κόμβου και fd_i^j είναι το αναγνωριστικό του i-οστού αρχείου της λίστας αρχείων του κόμβου id_j.

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

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 μέρος του γεγονός τύπου J (Join). Για να γίνει αυτό θα πρέπει να δημιουργήσετε τον κώδικα για μια διπλά συνδεδεμένη ταξινομημένη κυκλική λίστα που θα αποτελεί τη λίστα των κόμβων. Φτιάξτε διαδικασίες DL_Insert, DL_Delete και DL_LookUp για τη διπλά συνδεδεμένη λίστα, προκειμένου να πραγματοποιείτε εισαγωγές, διαγραφές και αναζητήσεις, αντίστοιχα, στη λίστα αυτή. Το πεδίο DocL δεν είναι απαραίτητο να συμπεριληφθεί στη δομή του κόμβου σε αυτό το βήμα. Δημιουργήστε μια διαδικασία DL_Print() για την εκτύπωση των στοιχείων της λίστας η οποία θα σας βοηθήσει να ελέγξετε την ορθότητα των διαδικασιών που περιγράφονται παραπάνω. Μόνο όταν ο κώδικας του βήματος 1 εκτελείται σωστά θα πρέπει να προχωρήστε στο βήμα 2.

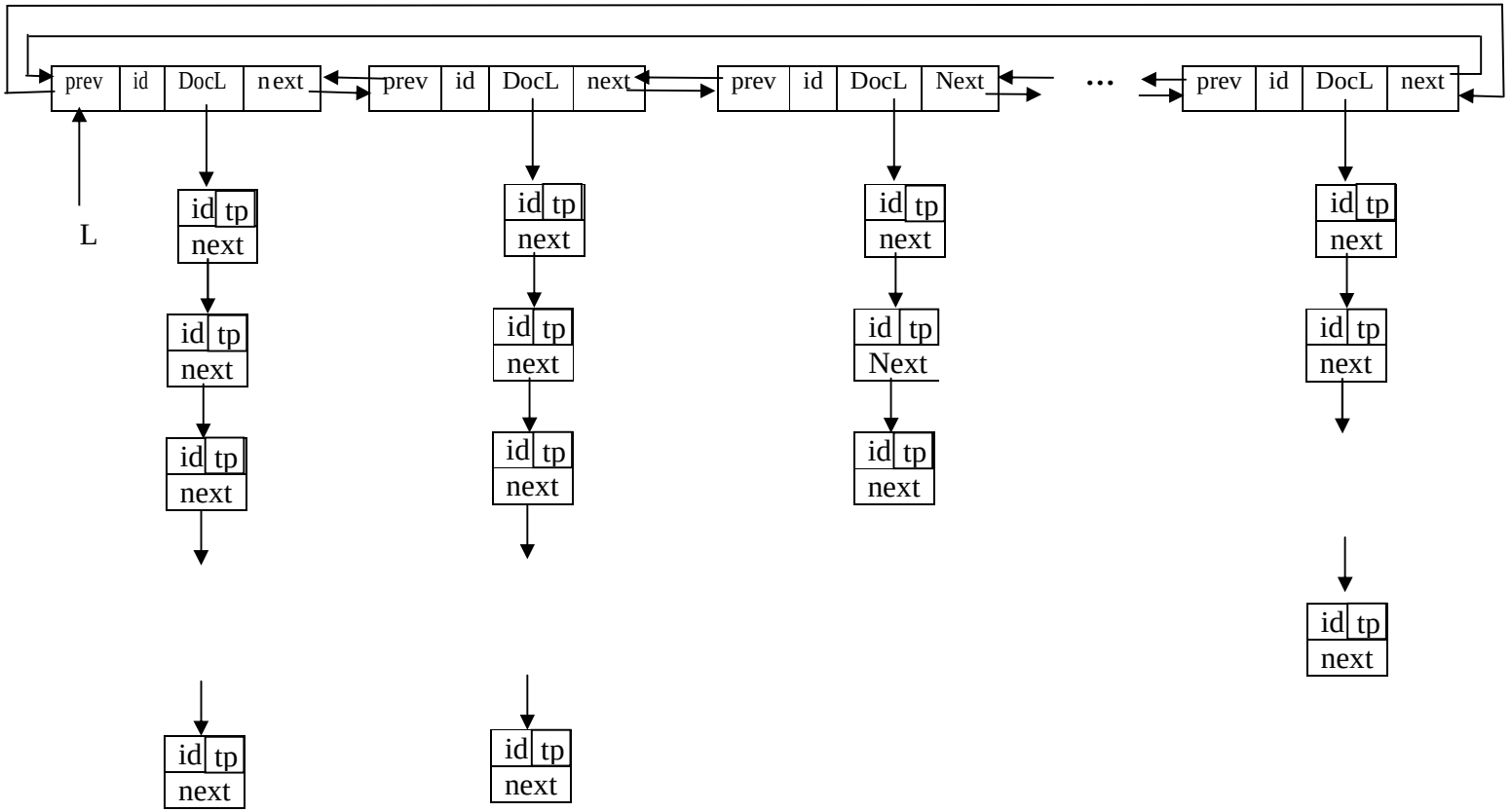
Βήμα 2: Το βήμα αυτό αποσκοπεί στην υλοποίηση της λίστας των αρχείων. Η λίστα των αρχείων ενός κόμβου μπορεί να είναι ταξινομημένη ή μη. Αποφασίστε τι από τα δύο θεωρείτε πιο βολικό. Στη συνέχεια, σε ένα νέο αρχείο, στο οποίο δεν περιέχεται ο κώδικας του βήματος 1, γράψτε κώδικα για τις λειτουργίες SL_Insert, SL_Delete και SL_LookUp μιας απλά συνδεδεμένης λίστας. Επίσης, δημιουργείστε μια διαδικασία SL_Print για την εκτύπωση των στοιχείων της λίστας. Ο κώδικας που έχετε δημιουργήσει στο βήμα 1 δεν χρησιμοποιείται καθόλου σε αυτό το βήμα (εδώ πρέπει να υλοποιήσετε μια απλά συνδεδεμένη λίστα).

Βήμα 3: Στο αρχείο κώδικα του Βήματος 2, δημιουργήστε μια συνάρτηση SL_Split() η οποία δοθέντος ενός δείκτη L στην αρχή μιας (απλά συνδεδεμένης) λίστας και ενός κλειδιού K δημιουργεί μια νέα λίστα L' με τον ακόλουθο τρόπο. Κάθε στοιχείο της L που έχει τιμή στο πεδίο id μικρότερη ή ίση του K εξάγεται από την L και τοποθετείται σε μια νέα λίστα L'. Στην L απομένουν μόνο εκείνα τα στοιχεία που έχουν στο πεδίο id τιμή μεγαλύτερη του K.

Δημιουργείστε επίσης μια συνάρτηση SL_Merge() η οποία δοθέντος δύο δεικτών L1 και L2 στα πρώτα στοιχεία δύο απλά συνδεδεμένων λιστών, συνενώνει τις δύο λίστες σε μία μεταφέροντας όλα τα στοιχεία της L1 στην L2.

Βήμα 4: Συνδυάστε τους κώδικες των αρχείων των Βημάτων 1 και 2-3. Υλοποιήστε και εξετάστε πρώτα την ορθότητα των γεγονότων τύπου Join, Insert, Find και Delete. Αρχικά, ο κώδικας σας πρέπει να εκτελείται σωστά για στατικό πλήθος κόμβων (δηλαδή όταν όλα τα Join έχουν πραγματοποιηθεί πριν από κάθε Insert, Find ή Delete). Συνδυάστε τις διαδικασίες DL_LookUp και SL_LookUp για την επεξεργασία των γεγονότων τύπου Find. Συνδυάστε επίσης τις διαδικασίες DL_Print και SL_Print για να δημιουργήσετε την διαδικασία Print η οποία θα υλοποιεί το γεγονός τύπου P.

Βήμα 5: Χρησιμοποιώντας τη συνάρτηση SL_Split παράγετε κώδικα για την περίπτωση στην οποία κόμβοι εισέρχονται δυναμικά (οποιαδήποτε χρονική στιγμή) στο σύστημα. Αυτό σημαίνει πως κάποιες λίστες αρχείων θα πρέπει να γίνονται split προκειμένου να ισχύει η (1) μετά από κάθε εισαγωγή κόμβου στο σύστημα. Υλοποιήστε τέλος γεγονότα τύπου Leave. Για να πραγματοποιηθεί αυτό θα πρέπει να τροποποιήσετε/χρησιμοποιήσετε κατάλληλα τον κώδικα των συναρτήσεων DL_Delete και SL_Merge.


prev
id
DocL
next
prev
id
DocL
next
prev
id
DocL
Next
...
prev
id
DocL
next
L
id
tp
next
Next

Σχήμα 4