Δομές Δεδομένων

1η Εργασία

2020-2021

Στα πλαίσια της εργασίας ζητείται να συγγράψετε πρόγραμμα σε C++ το οποίο θα επιλύει το πρόβλημα που ακολουθεί.

Δίνεται ένα σύνολο από k γραμμικές λίστες L_i, i = 0, 1, 2, ..., k-1. Στα πλαίσια της εργασίας θα πρέπει να χρησιμοποιήσετε την συνδεδεμένη αναπαράσταση για τις γραμμικές λίστες. Για λεπτομέρειες δείτε το Κεφ. 3.4 του βιβλίου "Δομές Δεδομένων, Αλγόριθμοι και Εφαρμογές στη C++".

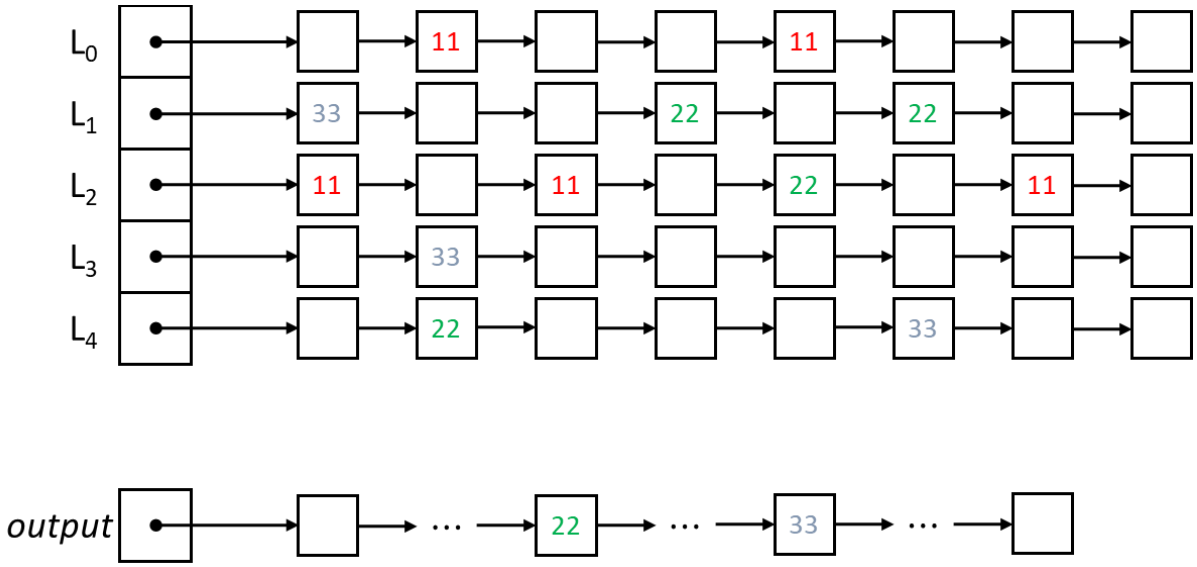
Τα δεδομένα σε κάθε κόμβο για όλες τις λίστες L_i αποτελούνται από έναν ακέραιο αριθμό. Δεν είναι απαραίτητο όλες οι λίστες να περιέχουν το ίδιο πλήθος κόμβων, ενώ επιτρέπεται να εμφανίζεται ο ίδιος αριθμός περισσότερες φορές σε κάθε λίστα. **Ωστόσο, δεν επιτρέπεται η χρήση των έτοιμων δομών δεδομένων που προσφέρει η C++ (C++ Containers).**

Σκοπός του προγράμματος που θα φτιάξετε είναι να δημιουργήσετε μια νέα λίστα *output* στην οποία θα τοποθετηθούν τα στοιχεία των L_i τα οποία εμφανίζονται σε τουλάχιστον από τις μισές λίστες. Κάθε τέτοιο στοιχείο θα πρέπει να εμφανίζεται μόνο μια φορά στην λίστα *output* και όλα τα στοιχεία της λίστας *output* θα πρέπει να είναι ταξινομημένα κατά αύξουσα σειρά.

Στο τέλος του προγράμματος οι αρχικές λίστες L_i **δεν** είναι απαραίτητο να βρίσκονται στην ίδια κατάσταση στην οποία βρίσκονταν στην αρχή του προγράμματος. Επιτρέπεται να αφαιρέσετε οσαδήποτε στοιχεία θέλετε από την κάθε λίστα L_i ή να αναδιατάξετε στοιχεία εντός κάθε λίστας L_i κατά την διάρκεια εκτέλεσης του αλγορίθμου. **Δεν επιτρέπεται ωστόσο να επεξεργαστείτε τις λίστες L_i εκ των προτέρων, π.χ. δεν επιτρέπεται να ταξινομήσετε τις λίστες στην αρχή εκτέλεσης του προγράμματος.**

Το γεγονός ότι επιτρέπεται να τροποποιήσετε τις λίστες L_i και ότι η λίστα *output* πρέπει να είναι ταξινομημένη σας δίνει την δυνατότητα να βελτιστοποιήσετε τον αλγόριθμο σας, μειώνοντας κατά το δυνατό το πλήθος των συγκρίσεων που απαιτούνται μέχρι την τελική κατασκευή της λίστας *output*. Προσπαθήστε να σκεφτείτε και να υλοποιήσετε τέτοιου είδους βελτιστοποιήσεις στον αλγόριθμο σας.

Το παρακάτω σχήμα δίνεται προς διευκρίνιση λεπτομερειών. Για παράδειγμα, αν και ο αριθμός 11 εμφανίζεται αρκετές φορές συνολικά, ωστόσο εμφανίζεται μόνο σε δύο λίστες (λιγότερες από τις μισές) και έτσι δεν πρέπει να συμπεριληφθεί στην λίστα *output*. Αντιθέτως, ο αριθμός 33 εμφανίζεται μόνο τρεις φορές, όμως εμφανίζεται σε τρεις διαφορετικές λίστες (περισσότερες από τις μισές) και έτσι πρέπει να συμπεριληφθεί στην λίστα *output*. Τέλος, παρατηρείστε πως τα στοιχεία στην λίστα *output* είναι ταξινομημένα.



Θέματα υλοποίησης

- Το πλήθος k από λίστες θα πρέπει να διαβάζεται από το πρόγραμμα σας κατά τον χρόνο εκτέλεσης. Κατά συνέπεια, η δομή που θα αποθηκεύει το σύνολο των λιστών L_i θα πρέπει να κατασκευάζεται δυναμικά κατά τον χρόνο εκτέλεσης.
- Στην συνέχεια το πρόγραμμα σας θα πρέπει να κατασκευάσει τις λίστες. Θα πρέπει για κάθε λίστα L_i να οριστεί το μέγεθος της (το πλήθος των κόμβων από το οποίο θα αποτελείται) και για κάθε κόμβο θα πρέπει να οριστεί ο αριθμός που θα περιέχει ως δεδομένο. Για τον σκοπό αυτό μπορεί να γίνει χρήση **γεννήτριας τυχαίων αριθμών**. Η C++ προσφέρει ένα σύνολο κλάσεων και μεθόδων για τον σκοπό αυτό (δείτε στο https://www.cplusplus.com/reference/random). Για να χρησιμοποιήσετε αυτές τις δυνατότητες θα πρέπει:
    - Να συμπεριλάβετε τα αρχεία κεφαλίδας <random> και <functional> στο πρόγραμμα σας:

      ```
      #include <random>
      #include <functional>
      ```

    - Να ορίσετε μια γεννήτρια τυχαίων αριθμών:

      ```
      std::default_random_engine generator;
      ```

    - Να ορίσετε τα επιτρεπτά όρια και την κατανομή για τους παραγόμενους τυχαίους αριθμούς. Με τις παρακάτω δηλώσεις ζητάμε για το μέγεθος κάθε λίστας να χρησιμοποιηθεί ομοιόμορφη κατανομή και οι παραγόμενοι αριθμοί να είναι ακέραιοι στο διάστημα [100, 200] (δηλαδή το μέγεθος κάθε λίστας να είναι από 100 έως 200 στοιχεία), ενώ για τον αριθμό που θα περιέχει κάθε κόμβος ζητάμε πάλι ομοιόμορφη κατανομή και οι παραγόμενοι αριθμοί να είναι ακέραιοι στο διάστημα [0, 50]. Είστε ελεύθεροι να αλλάξετε τα όρια αυτά:

```
std::uniform_int_distribution<int> list_size_distribution(100, 200);
std::uniform_int_distribution<int> data_element_distribution(0, 50);
```

- Προς ευκολία χρήσης μπορείτε να «δέσετε» μαζί την γεννήτρια τυχαίων αριθμών με την επιθυμητή κατανομή:

```
auto random_list_size = std::bind(list_size_distribution, generator);
auto random_element   = std::bind(data_element_distribution, generator);
```

- Κάθε φορά που θα καλείτε την `random_list_size()` και την `random_element()` θα παράγεται ένας τυχαίος αριθμός με βάση την κατανομή και τα όρια που έχετε ορίσει. Π.χ. με τις παρακάτω κλήσεις δημιουργείται ένας τυχαίος αριθμός για το μέγεθος μιας λίστας στο διάστημα [100, 200] και ένας τυχαίος αριθμός στο διάστημα [0, 50] που μπορεί να χρησιμοποιηθεί ως δεδομένο για έναν κόμβο σε μια λίστα:

```
int list_size = random_list_size();
int data_element = random_element();
```

- Όπως προαναφέρθηκε, από την στιγμή που θα δημιουργηθούν με χρήση τυχαίων αριθμών οι λίστες L_i δεν επιτρέπεται να τις επεξεργαστείτε, παρά μόνο αφού ξεκινήσει ο αλγόριθμος σας για την κατασκευή της λίστας *output*.

Παραδοτέα

Θα πρέπει να παραδοθεί ο πηγαίος κώδικας μαζί με τον εκτελέσιμο. Ιδιαίτερη βαρύτητα θα πρέπει να δοθεί στη σωστή τεκμηρίωση των προγραμμάτων σας. Θα πρέπει λοιπόν ο κώδικας σας να συνοδεύεται από ξεχωριστό κείμενο που θα παρέχει λεπτομερή περιγραφή των τεχνικών σας. Επίσης, εντός του πηγαίου κώδικα θα πρέπει να συμπεριληφθούν «πυκνά» **σχόλια ουσίας**. Η παράδοση των εργασιών θα γίνει μέσω του gunet.

Η εργασία μπορεί να εκπονηθεί από **ομάδα μέχρι δύο ατόμων αυστηρώς**.

Προθεσμία Παράδοσης

Παρασκευή 14 Μαΐου 2021