HY240 — Δομες Δεδομενων

2η Σειρά Ασκήσεων

Τρόπος Παράδοσης

Οι ασκήσεις μπορούν να παραδοθούν μόνο δια ζώσης στους βοηθούς του μαθήματος, τη **Δευτέρα, 30 Οκτωβρίου 2023, ώρα 14:00-15:00 μ.μ.** στο γραφείο τους (**Β.208** / **Β.210**). Εμπρόθεσμες παραδόσεις γίνονται επίσης δεκτές στις διαλέξεις ή και τις ώρες γραφείου των βοηθών (μετά από συννενόηση με τον υπεύθυνο βοηθό). Ασκήσεις που παραδίδονται μετά τις 15:00 μ.μ. της Δευτέρας, 30/10/2023 θεωρούνται εκπρόθεσμες. **Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή** και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα `turnin`. Για περισσότερες πληροφορίες ανατρέξτε εδώ.

Εξαμηνο:	Εαρινό (2023-24)
Πανεπιστημιο:	Πανεπιστήμιο Κρήτης
Τμημα:	Επιστήμης Υπολογιστών
Καθηγητης/τρια:	Παναγιώτα Φατούρου
Υπευθυνοι Βοηθοι:	Κατερίνα Πετράκη
Τελευταια Τροποποιηση Εκφωνησης:	25 / 10 / 2023

Άσκηση 1 **[35 μονάδες]**

a. Μια μεταφορική εταιρεία (π.χ. μια courier) αποθηκεύει πληροφορίες για τις δοσοληψίες μεταφορών που έχει πραγματοποιήσει. Το πρόγραμμα διαχείρισης δεδομένων της μεταφορικής εταιρείας χρησιμοποιεί μια δομή δεδομένων, η οποία αποθηκεύει μια εγγραφή για κάθε δοσοληψία μεταφοράς. Κάθε εγγραφή έχει ένα μοναδικό αναγνωριστικό. Ας υποθέσουμε ότι η δομή δεδομένων που χρησιμοποιεί η εταιρεία είναι μια δυναμική απλά-συνδεδεμένη λίστα, της οποίας τα στοιχεία έχουν την εξής μορφή:

```
struct courier_order {
  int id;
  int source;
  int destination;
  int dest_continent;
  char date[100];
  struct courier_order *next;
}
```

Υπάρχουν 5 ήπειροι στις οποίους δραστηριοποιείται η εταιρεία, η Ευρώπη (με αναγνωριστικό 1), η Αμερική (με αναγνωριστικό 2), η Ασία (με αναγνωριστικό 3), η Αφρική (με αναγνωριστικό 4) και η Ωκεανία (με αναγνωριστικό 5). Το πεδίο `dest_continent` στο `struct courier_order` αποθηκεύει το αναγνωριστικό της ηπείρου προς την οποία αποστέλλεται μια μεταφορά.

Θεωρήστε ότι **η λίστα είναι ταξινομημένη ως προς τα αναγνωριστικά των δοσοληψιών σε αύξουσα διάταξη**. Μετά από πολλά χρόνια λειτουργίας της μεταφορικής εταιρείας, το Διοικητικό της Συμβούλιο αποφάσισε πως θα αλλάξει ο τρόπος διαχείρισης των δεδομένων της εταιρείας, έτσι ώστε να υπάρχει μια δομή δεδομένων που θα αποθηκεύει τις πληροφορίες των δοσοληψιών μεταφορών για κάθε ήπειρο. Συνολικά επομένως θα πρέπει να υπάρχουν 5 λίστες (όσες και οι ήπειροι), η κάθε μια εκ των οποίων είναι απλά-συνδεδεμένη και ταξινομημένη ως προς το αναγνωριστικό των δοσοληψιών σε αύξουσα διάταξη.

Μετά την απόφαση του Διοικητικού Συμβουλίου, θα πρέπει, ως μέρος του προγράμματος διαχείρισης δεδομένων της εταιρείας, να υλοποιηθεί μια λειτουργία που θα επεξεργάζεται την αρχική δομή δεδομένων και θα δημιουργεί από αυτήν πέντε αντίστοιχες δομές, τέτοιες ώστε: στην πρώτη θα περιέχονται εγγραφές για δοσοληψίες που πραγματοποιήθηκαν προς Ευρώπη, στη δεύτερη θα περιέχονται οι εγγραφές που αντιστοιχούν στις δοσοληψίες που έχουν πραγματοποιηθεί προς Αμερική, κ.ο.κ.. **Περιγράψτε (μη-αναδρομικό) αλγόριθμο υλοποίησης της λειτουργίας αυτής.** Η χρονική πολυπλοκότητα του αλγορίθμου σας, θα πρέπει να είναι $\mathrm{O}(n)$, όπου n είναι το πλήθος των στοιχείων στην αρχική λίστα που περιέχει όλες τις δοσοληψίες, ανεξαρτήτως ηπείρων. [15Μ]

b. Θεωρήστε μια **διπλά-συνδεδεμένη λίστα** L.

(i) Παρουσιάστε ψευδοκώδικα για έναν μη αναδρομικό αλγόριθμο που θα ταξινομεί τη λίστα L. Ο αλγόριθμος που θα παρουσιάσετε θα πρέπει να υλοποιεί την `InsertionSort` πάνω στη λίστα. Δεν επιτρέπεται η χρήση βοηθητικών δομών για τη σχεδίαση του αλγορίθμου. [15M]

(ii) Ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας και γιατί; [5M]

Άσκηση 2 [35 μονάδες]

Έστω ότι μια βιβλιοθήκη σας παρέχει πρόσβαση σε στοίβες και ουρές χαρακτήρων. Η βιβλιοθήκη σας επιτρέπει να ορίσετε μια στοίβα (ή μια ουρά) και να καλέσετε τις 5 βασικές λειτουργίες σ' αυτήν. Για παράδειγμα, ο ορισμός μιας στοίβας (ή μιας ουράς) γίνεται γράφοντας: `Stack S;` (αντίστοιχα, `Queue Q;`). Για τη στοίβα υποστηρίζονται οι εξής λειτουργίες: (1) `void MakeEmptyStack(Stack S)`, (2) `boolean IsEmptyStack(Stack S)`, (3) `Type Top(Stack S)`, (4) `Type Pop(Stack S)`, (5) `void Push(Stack S, Type x)`. Αντίστοιχα, για την ουρά υποστηρίζονται οι λειτουργίες: (1) `void MakeEmptyQueue(Queue Q)`, (2) `boolean IsEmptyQueue(Queue Q)`, (3) `Type Front(Queue Q)`, (4) `Type Dequeue(Queue Q)`, (5) `void Enqueue(Queue Q, Type x)`, όπου `Type` μπορεί να είναι οποιοσδήποτε τύπος (`char`, `int`, `float`, `double`, etc.).

a. Παρουσιάστε ψευδοκώδικα για έναν μη-αναδρομικό αλγόριθμο, ο οποίος θα διαβάζει μια έκφραση που αποτελείται από άγκιστρα, αγκύλες και παρενθέσεις και θα ελέγχει αν το άνοιγμα των αγκίστρων, αγκυλών και παρενθέσεων είναι συμβατό με το κλείσιμο τους (δηλαδή τα αριστερά άγκιστρα, οι αριστερές αγκύλες και οι αριστερές παρενθέσεις, καθώς και η σειρά με την οποία αυτά απαντώνται στην έκφραση «ταιριάζουν» με τα δεξιά άγκιστρα, τις δεξιές αγκύλες και τις δεξιές παρενθέσεις και τη σειρά με την οποία αυτά απαντώνται). Για παράδειγμα, ο αλγόριθμος που θα σχεδιάσετε θα πρέπει να τυπώνει TRUE για την ακόλουθη έκφραση:

```
{{[(([[{{{()}}}()]{}]()()()))[][[]])]}}
```

και FALSE για την ακόλουθη έκφραση:

```
{{[(([[{{{()}}}()]{}()()()))[][[]])]}}
```

Επιτρέπεται να χρησιμοποιηθεί μόνο μία ουρά ή στοίβα για την επίλυση του προβλήματος. Επίσης ο αλγόριθμος θα πρέπει να διαβάζει μια μόνο φορά την έκφραση. Η λύση σας θα πρέπει να έχει την ακόλουθη μορφή: [8M]

```
while ((ch = getchar()) != '\n') {
  switch(ch) {
    case ch is '{':
      break;
    case ch is '}':
      break;
    ...
  }
}
```

b. Περιγράψτε αναδρομική έκδοση του αλγορίθμου που θα παρουσιάσετε στο ερώτημα a. [10M]

c. Θεωρήστε μια στοίβα S, κάθε στοιχείο της οποίας είναι ένας ακέραιος number. Παρουσιάστε ψευδοκώδικα για έναν μη αναδρομικό αλγόριθμο που θα ταξινομεί την S. Ο αλγόριθμος που θα παρουσιάσετε θα πρέπει να υλοποιεί την `InsertionSort` πάνω στην στοίβα. Για τη σχεδίαση του αλγορίθμου επιτρέπεται να χρησιμοποιήσετε δύο βοηθητικές στοίβες, επιπροσθέτως της S. [10M]

d. Θεωρήστε μια ουρά Q, κάθε στοιχείο της οποίας είναι ένας ακέραιος number και τα στοιχεία στην ουρά είναι ταξινομημένα σε αύξουσα διάταξη. Παρουσιάστε ψευδοκώδικα για έναν αλγόριθμο που θα διαγράφει όλα τα στοιχεία της Q για τα οποία ισχύει ότι η διαίρεσή τους με το 5 είναι άρτιος αριθμός. Θεωρήστε πως γνωρίζετε τον αριθμό n των στοιχείων στην Q. Ο αλγόριθμός που θα σχεδιάσετε δεν επιτρέπεται να χρησιμοποιεί καμία βοηθητική δομή. [7M]

Σημειώσεις:
- Οι αλγόριθμοι θα πρέπει να χρησιμοποιούν **μόνο τις δομές δεδομένων από τη βιβλιοθήκη που καθορίζονται από την εκφώνηση**, καθώς και ενδεχόμενα κάποιες βοηθητικές μεταβλητές.
- Δεν επιτρέπεται η χρήση πινάκων ή άλλων δομών δεδομένων που δεν ανήκουν στη βιβλιοθήκη για την αποθήκευση ή την επεξεργασία των αλφαριθμητικών ή των ακολουθιών χαρακτήρων.
- Δεν γνωρίζετε αν οι στοίβες και οι ουρές που παρέχει η βιβλιοθήκη υλοποιούνται με στατικό ή με δυναμικό τρόπο.

Άσκηση 3 **[30 μονάδες]**

a. Σε ένα πάρτυ γενεθλίων έχει αποφασιστεί μια συγκεκριμένη playlist με τραγούδια τα οποία είναι αποθηκευμένα σε μια απλά-συνδεδεμένη λίστα L, σε φθίνουσα διάταξη ως προς τη χρονική διάρκεια κάθε τραγουδιού (ώστε να ακουστούν πρώτα τα τραγούδια με περισσότερη διάρκεια).

(i) Παρουσιάστε ψευδοκώδικα για έναν μη-αναδρομικό αλγόριθμο, ο οποίος θα ταξινομεί τη λίστα σε αύξουσα διάταξη (ως προς τη χρονική διάρκεια κάθε τραγουδιού). Ο αλγόριθμος που θα σχεδιάσετε δεν θα πρέπει να χρησιμοποιεί καμία επιπρόσθετη βοηθητική δομή (πίνακα, λίστα, ουρά, στοίβα). Μπορεί ωστόσο να χρησιμοποιεί βοηθητικές μεταβλητές, πχ δείκτες, κλπ. Ο αλγόριθμος σας θα πρέπει να έχει χρονική πολύπλοκότητα $\mathrm{O}(n)$, όπου n είναι το πλήθος στοιχείων στην L. [10M]

(ii) Παρουσιάστε αναδρομική έκδοση του αλγορίθμου που παρουσιάσατε στο ερώτημα a. [10M]

b. Θεωρήστε μια διπλά-συνδεδεμένη ταξινομημένη λίστα L. Θεωρήστε ότι η L έχει άρτιο αριθμό στοιχείων και συγκεκριμένα ο αριθμός αυτός είναι $2k$, όπου k είναι κάποιος ακέραιος. Θεωρήστε ότι τα πρώτα k στοιχεία της L είναι ταξινομημένα σε αύξουσα διάταξη, ενώ τα τελευταία k στοιχεία της L είναι ταξινομημένα σε φθίνουσα διάταξη. Οι ταξινομήσεις είναι ως προς το μοναδικό αναγνωριστικό `id` του κάθε κόμβου. Καθένα από τα δύο μισά μέρη της L, που απαρτίζεται από k στοιχεία, μπορεί να περιέχει στοιχεία, τα οποία είναι άλλα μεγαλύτερα και άλλα μικρότερα από εκείνα που υπάρχουν στο άλλο μισό. Ζητείται ψευδοκώδικας για έναν αλγόριθμο που θα δημιουργεί μια διπλά-συνδεδεμένη λίστα DL, η οποία θα είναι ταξινομημένη σε φθίνουσα διάταξη ως προς το αναγνωριστικό `id`, και θα περιέχει όλα τα στοιχεία της L. Η λίστα L θα πρέπει να παραμείνει αναλλοίωτη κατά την εκτέλεση του αλγορίθμου. Ο αλγόριθμος θα πρέπει να εκτελείται σε χρόνο $\mathrm{O}(n)$, όπου n είναι το πλήθος στοιχείων της L. Θεωρήστε ότι ο αλγόριθμος γνωρίζει το k (πχ το δέχεται ως παράμετρο). [10M]