ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Δομές Δεδομένων - Εργασία 1

Τμήμα Πληροφορικής

Φθινοπωρινό Εξάμηνο 2017-2018

Διδάσκων: Ε. Μαρκάκης

Ουρές: Υλοποιήσεις ΑΤΔ και εφαρμογές

Σκοπός της εργασίας είναι η εξοικείωση με βασικούς αφηρημένους τύπους δεδομένων όπως οι ουρές FIFO και οι γενικεύσεις τους. Η εργασία αποτελείται από υλοποιήσεις ΑΤΔ (Μερη Α, Γ και Δ) καθώς και 1 εφαρμογή (Μέρος Β). Διαβάστε προσεκτικά την εκφώνηση και τα ζητούμενα της εργασίας.

Μέρος Α. Να υλοποιήσετε την διεπαφή CharDoubleEndedQueue για *ουρές,* οι οποίες χειρίζονται χαρακτήρες, δηλαδή δεδομένα τύπου char, και στις οποίες μπορεί να γίνεται εισαγωγή και διαγραφή στοιχείων και από τα 2 άκρα τους. Το αρχείο CharDoubleEndedQueue.java που περιέχει την δήλωση της διεπαφής είναι διαθέσιμο στο φάκελο 'Εγγραφα/Εργασίες στο eclass.

Οδηγίες υλοποίησης:

- Η κλάση σας **πρέπει να λέγεται** CharDoubleEndedQueueImpl.
- Η υλοποίηση θα πρέπει να γίνει χρησιμοποιώντας διπλά συνδεδεμένη λίστα.
- Κάθε μέθοδος εισαγωγής ή εξαγωγής στοιχείου θα πρέπει να ολοκληρώνεται σε χρόνο O(1), δηλαδή σε χρόνο ανεξάρτητο από τον αριθμό των αντικειμένων που είναι μέσα στην ουρά. Ομοίως, η μέθοδος size θα πρέπει να εκτελείται σε O(1).
- Όταν η ουρά είναι άδεια, οι μέθοδοι που διαβάζουν από την δομή θα πρέπει να πετάνε εξαίρεση τύπου NoSuchElementException. Η εξαίρεση NoSuchElementException ανήκει στην core βιβλιοθήκη της Java. Κάντε την import από το πακέτο **java.util**.
- Μπορείτε είτε να βασιστείτε στον κώδικα του εργαστηρίου και να τροποποιήσετε κατάλληλα τη λίστα μονής σύνδεσης που παρουσιάστηκε στο εργαστήριο 2 είτε να γράψετε εξ' ολοκλήρου τη δική σας λίστα είτε να χρησιμοποιήσετε μόνο αντικείμενα τύπου Node μέσα στην κλάση της ουράς. Για να αποκτήσετε καλύτερη εξοικείωση προτείνουμε να ξεκινήσετε χωρίς τον κώδικα του εργαστηρίου και να γράψετε τις δικές

σας κλάσεις (σίγουρα δεν θα χάσετε μονάδες όμως αν χρησιμοποιήσετε κάτι που έχετε δει στο εργαστήριο).

- **Προαιρετικά:** μπορείτε να κάνετε την υλοποίηση σας με χρήση generics για να μπορείτε να χειρίζεστε ουρές με οποιοδήποτε τύπο αντικειμένων. Υπάρχει ένα 10% bonus σε όσους χρησιμοποιήσουν generics για τα Μέρη Α και Γ της εργασίας. Αν κάνετε χρήση generics, επιτρέπεται να τροποποιήσετε κατάλληλα τα interfaces που σας δίνονται για να δουλεύουν για οποιονδήποτε τύπο δεδομένων.
- **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας, στοίβας, ουράς, από την βιβλιοθήκη της Java** (π.χ. Vector, ArrayList κλπ).

Μέρος Β. Γράψτε ένα πρόγραμμα-πελάτη για να λύσετε το παρακάτω πρόβλημα χρησιμοποιώντας την υλοποίηση από το Μέρος Α.

Μία ακολουθία DNA είναι ένα αλφαριθμητικό (String) που αποτελείται από τα γράμματα A, T, C, G. Τα γράμματα αυτά αντιστοιχούν στα νουκλεοτίδια Αδενίνη, Θυμίνη, Κυτοσίνη, Γουανίνη. Τα νουκλεοτίδια Α και Τ είναι συμπληρωματικά μεταξύ τους και θα λέμε ότι το Α είναι το συμπλήρωμα του Τ και αντιστρόφως. Ομοίως το C είναι το συμπλήρωμα του G και το G το συμπλήρωμα του C. Μία ακολουθία DNA ονομάζεται *Watson-Crick complemented palindrome* (συμπληρωματικά παλίνδρομη) αν, όταν αντικαταστήσουμε κάθε νουκλεοτίδιο με το συμπλήρωμά του και μετά αντιστρέψουμε την σειρά, τότε παίρνουμε την ίδια ακολουθία με την αρχική. Για παράδειγμα το string AAAACGTTTT είναι Watson-Crick complemented palindrome. Αντιθέτως το ATAATA δεν είναι (παρατηρήστε ότι η έννοια αυτή είναι διαφορετική από το να είναι απλά παλίνδρομη μια ακολουθία). Οι ακολουθίες DNA με αυτή την ιδιότητα έχουν σημαντικούς βιολογικούς ρόλους. Για παράδειγμα τα καρκινικά κύτταρα συχνά μεγαλώνουν τα γονίδιά τους δημιουργώντας τέτοιου είδους παλίνδρομα DNA.

Γράψτε ένα πρόγραμμα, το οποίο διαβάζει από την είσοδο μια ακολουθία χαρακτήρων που αναπαριστά DNA ακολουθία και αποφασίζει αν είναι Watson-Crick complemented palindrome. Η main σας θα πρέπει να διαβάζει την ακολουθία DNA από το standard input, να ελέγχει αν είναι έγκυρη (π.χ. δεν περιέχει κενά ενδιάμεσα ή μη αποδεκτούς χαρακτήρες, μη κεφαλαία γράμματα), και να εκτυπώνει το σχετικό αποτέλεσμα. Αν η ακολουθία δεν είναι έγκυρη το πρόγραμμα πρέπει απλά να τερματίζει και να τυπώνει ένα μήνυμα λάθους.

Επιπλέον παραδείγματα από Watson-Crick παλινδρόμηση

Ναι: "ATATATAT", και "" (το κενό string ικανοποιεί την ιδιότητα).
Όχι: "AAAA", "AAAAGTTTT", "ATAATA", "ZZZZ" και "AaTT".

Οδηγίες υλοποίησης:

- Το πρόγραμμα σας **πρέπει να λέγεται** DNAPalindrome.java.
- Η υλοποίηση σας εσωτερικά θα πρέπει να χρησιμοποιεί μόνο 1 αντικείμενο ουράς με διπλά άκρα από το Μέρος Α.

Μέρος Γ. Στο φάκελο 'Εγγραφα/Εργασίες δίνεται το αρχείο CharQueue.java που περιέχει την δήλωση της διεπαφής για μια ουρά FIFO που χειρίζεται στοιχεία τύπου char. Δημιουργήστε μία υλοποίηση του ΑΤΔ CharQueue.

Οδηγίες υλοποίησης:

- Η κλάση σας **πρέπει να λέγεται** CharQueueImpl.
- Η υλοποίηση σας θα πρέπει να γίνει χρησιμοποιώντας μια λίστα μονής σύνδεσης.

- Οι λειτουργίες εισαγωγής και εξαγωγής θα πρέπει να γίνονται σε O(1) και γενικότερα ισχύουν και εδώ όλες οι οδηγίες που αναγράφονται για το Μέρος Α (εκτός από τη χρήση διπλά συνδεδεμένης λίστας).

Μέρος Δ. Έστω ότι θέλετε να υλοποιήσετε την διεπαφή για μια ουρά FIFO, όπως στο ερώτημα Γ, έτσι ώστε να υποστηρίζεται επιπλέον και η μέθοδος: `char min()`, η οποία επιστρέφει τον λεξικογραφικά μικρότερο χαρακτήρα που βρίσκεται στην ουρά (δεν θα τον αφαιρεί από την ουρά). Ας ονομάσουμε CharQueueWithMin τη νέα διεπαφή, η οποία προκύπτει από την διεπαφή του Μέρους Γ, προσθέτοντας τη νέα αυτή μέθοδο. Δημιουργήστε μία υλοποίηση του ΑΤΔ CharQueueWithMin χρησιμοποιώντας τα προηγούμενα ερωτήματα της άσκησης. Ένας τρόπος για να γίνει αυτό θα ήταν όταν εκτελείται η `min()` να σαρώσετε απλώς όλη την ουρά και να βρείτε το ελάχιστο στοιχείο. Όμως αυτό στην πράξη δεν είναι πολύ αποτελεσματικό ειδικά όταν αρχίζει και μεγαλώνει πολύ η ουρά (χρειάζεται ***πάντα*** χρόνο O(N), ανεξάρτητα από το περιεχόμενο της ουράς). Αντί αυτού θα φτιάξετε μία διαφορετική υλοποίηση της διεπαφής, η οποία κατά μέσο όρο έχει καλύτερη πολυπλοκότητα, σύμφωνα με τις παρακάτω οδηγίες.

Οδηγίες υλοποίησης:

- Η κλάση θα λέγεται CharQueueWithMinImpl.
- Υπάρχουν διάφορες επιλογές στη Java για να ελέγξετε 2 χαρακτήρες και να δείτε ποιος είναι λεξικογραφικά μικρότερος (ενδεικτικά, μπορείτε να χρησιμοποιήσετε την μέθοδο compareTo από την κλάση περιτύλιξης Character).
- Η υλοποίησή σας θα πρέπει να χρησιμοποιεί 1 αντικείμενο ουράς FIFO από το ερώτημα Γ και 1 αντικείμενο ουράς με διπλά άκρα από το ερώτημα Α. Έστω F η ουρά FIFO και D η ουρά με τα διπλά άκρα.
    - Το σκεπτικό είναι ότι η F θα αναπαριστά το περιεχόμενο της δομής, όπως και στο Μέρος Γ.
    - Όμως, όταν εκτελείται η μέθοδος put της κλάσης CharQueueWithMinImpl, θα πρέπει να γίνεται εισαγωγή και στην F και στην D. Στην F θα γίνεται κανονική εισαγωγή μέσω της F.put. Στην D θα γίνεται εισαγωγή στην αρχή της με την D.addFirst, αφού όμως πρώτα φροντίσετε να διαγράψετε (όταν είναι απαραίτητο) από την D κάποια στοιχεία (σκεφτείτε ποια).
    - Η μέθοδος get της κλάσης CharQueueWithMinImpl θα πρέπει να αφαιρεί το παλιότερο στοιχείο από την F αλλά και από την D (αν δεν έχει αφαιρεθεί ήδη στην D από προηγούμενες διαγραφές).
    - Τέλος, χρησιμοποιώντας όλα τα παραπάνω, θα πρέπει να υλοποιήσετε τη μέθοδο min σε χρόνο O(1), χωρίς να χρειάζεται σάρωση όλων των στοιχείων της δομής.

Οδηγίες Παράδοσης

Η εργασία σας θα πρέπει να μην έχει συντακτικά λάθη και να μπορεί να μεταγλωττίζεται. Εργασίες που δεν μεταγλωττίζονται χάνουν το 50% της συνολικής αξίας.

Η εργασία θα αποτελείται από:

1. Τον πηγαίο κώδικα (source code). Τοποθετήστε σε ένα φάκελο με όνομα **src** τα αρχεία java που έχετε φτιάξει. Ενδεικτικά θα πρέπει να περιέχει (χρησιμοποιήστε τα όνοματα των κλάσεων όπως ακριβώς δίνονται στην εκφώνηση):
    a. Τις διεπαφές CharQueue, και CharDoubleEndedQueue, οι οποίες είναι ήδη διαθέσιμες στο eclass.
    b. Όλες τις υλοποιήσεις των διεπαφών από τα Μέρη Α, Γ, και Δ.
    c. Το πρόγραμμα DNAPalindrome.java από το Μέρος Β.

Επιπλέον, φροντίστε να συμπεριλάβετε όποια άλλα αρχεία πηγαίου κώδικα φτιάξατε και απαιτούνται για να μεταγλωττίζεται η εργασία σας. Φροντίστε επίσης να προσθέσετε επεξηγηματικά σχόλια όπου κρίνετε απαραίτητο στον κώδικά σας.

2. Γράψτε μία σύντομη αναφορά σε pdf αρχείο (όχι Word ή txt!) με όνομα project1-report.pdf, στην οποία θα αναφερθείτε στα εξής:
    a. Εξηγήστε συνοπτικά πώς υλοποιήσατε τις διεπαφές στα μέρη Α και Γ (άνω όριο 1 σελίδα).
    b. Για το μέρος Β, εξηγήστε πώς χρησιμοποιήσατε την υλοποίηση από το μέρος Α για να φτιάξετε το πρόγραμμα που ζητείται (άνω όριο 2 σελίδες).
    c. Για το Μέρος Δ, εξηγήστε αναλυτικά πώς χρησιμοποιήσατε την ουρά FIFO από το Μέρος Γ, και την ουρά με διπλά άκρα από το Μέρος Α για την υλοποίησή σας (άνω όριο 2 σελίδες).

Όλα τα παραπάνω αρχεία θα πρέπει να μπουν σε ένα αρχείο zip. Το όνομα που θα δώσετε στο αρχείο αυτό θα είναι ο αριθμός μητρώου σας πχ. 3030056_3030066.zip ή 3030056.zip (αν δεν είστε σε ομάδα). Στη συνέχεια, θα υποβάλλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τους 2 φοιτητές μιας ομάδας.

Η προθεσμία παράδοσης της εργασίας είναι Παρασκευή, 24 Νοεμβρίου 2017 και ώρα 23:59.