Αναφορα Παραδοσης Εργασιας 1 στο Μαθημα των Δομων Δεδομενων (INF231)

Σασσαλος Ρηγας (3220178)
Λαζανα Ευγενια (3220104)

Τμήμα Πληροφορικής / Οικονομικο Πανεπιστημιο Αθηνών
Διδάσκουσα: Κάτια Παπακωνσταντινοπούλου
Έτος: 2023

ΥΛΟΠΟΙΗΣΗ ΔΙΕΠΑΦΗΣ ΜΕΡΟΣ Α:

Στο μέρος Α εργαστήκαμε αρχικά δημιουργώντας τις απαραίτητες μεθόδους (isEmpty, addFirst, removeFirst, addLast, removeLast, getFirst, getLast, printQueue, size) σύμφωνα με το αρχείο που δόθηκε ως κατεύθυνση για την εργασία (StringDoubleEndedQueue.java).

Στις μεθόδους προσθήκης κόμβων στην ουρά χρησιμοποιούμε πάντα τον έλεγχο isEmpty για να επιβεβαιώσουμε οτι έχει κόμβους. Αν έχει κόμβους μεταφέρουμε το head ή το tail αντίστοιχα και θέτουμε καινούργιο next και previous κόμβο με την χρήση των μεθόδων της κλάσης Node (setNext, setPrevious) που δημιουργήσαμε ώστε να ειναι διπλά συνδεδεμένη ουρά. Αν η ουρά τελικά είναι κενή θέτουμε ως head και tail τον κόμβο που παμε να προσθέσουμε.

Όσον αφορα τις μεθόδους αφαίρεσης κόμβων απο την ουρά χρησιμοποιούμε προφανώς τον έλεγχο isEmpty για να αποφύγουμε σφάλμα (NoSuchElementException()). Αν δεν ειναι άδεια ελέγχουμε μήπως ο κόμβος που προσπαθούμε να αφαιρέσουμε ειναι ο τελευταίος στην ουρά. Αν είναι όντως ο τελευταίος μετατρεπουμε και το head και το tail σε null. Στην περίπτωση που δεν ισχύει κανένα απο τα δύο προηγούμενα θετουμε το next ή το previous του head ή tail αντίστοιχα με null και επιστρέφουμε τα δεδομένα του κόμβου που αφαιρέσαμε.

Στις μεθόδους getFirst και getLast απλά επιστρέφουμε το πρώτο ή τελευταίο στοιχείο αντίστοιχα ελέγχοντας με την isEmpty αν η ουρά ειναι άδεια.

Η printQueue χρησιμοποιεί μια while και την μέθοδο getItem της κλάσης Node για να εκτυπώσει όλα τα στοιχεία της ουράς.

Τέλος, η size λόγω ανάγκης διατήρησης της πολυπλοκότητας στο O(1) υλοποιήθηκε με έναν απλο counter που ενσωματώσαμε στις μεθόδους προσθήκης και αφαίρεσης κόμβων.

Όλο το μέρος Α εκτός απο την μέθοδο printQueue έχει πολυπλοκότητα O(1) διότι η κάθε εντολή εκτελείται μόνο μια φορά και δεν χρησιμοποιείται καμία επανάληψη.

ΥΛΟΠΟΙΗΣΗ ΜΕΡΟΥΣ Β (PREFIX ΤΟ INFIX):

Στο μέρος Β εργαστήκαμε ως εξής:

Στην main μέθοδο δημιουργήσαμε εναν scanner και ζητήσαμε απο τον χρήστη μια προθεματική έκφραση. Διαγράφοντας όλα τα κενά που μπορεί να πρόσθεσε ο χρήστης και αφού δημιουργήσουμε ενα αντικείμενο της κλάσης PrefixToInfix καλούμε την μέθοδο convert που έχουμε υλοποιήσει στην ιδια κλαση.

Η κλάση convert είναι η βασική κλάση της υλοποιησής μας. Ξεκινάμε αρχικά με την μέθοδο clear ώστε να σιγουρευτούμε οτι η ουρά ειναι άδεια. Στην συνέχεια, διαχωρίζουμε την προθεματική έκφραση που λάβαμε απο τον χρήστη (prefix) και αποθηκεύουμε τα στοιχεία ένα-ένα σε έναν πίνακα.

Χρησιμοποιούμε την μέθοδο isValid για να ελέγξουμε κατά πόσο η έκφραση ειναι έγκυρα διατυπωμένη και αφού επιβεβαιώσουμε οτι είναι σωστή δημιουργούμε ενα αντικείμενο της ουράς που υλοποιήθηκε στο μέρος Α και προσθέτουμε όλα τα στοιχεία του πίνακα στο τέλος της ουράς.

Αρχικοποιούμε μια μεταβλητή infix (String) με το κενο και ξεκινάμε επαναληπτικά να προσπελάυνουμε τα στοιχεία και να ελέγχουμε αν το στοιχείο item είναι σύμβολο πράξης με την μέθοδο isOperator που επιστρέφει true αν το item ειναι +, -, * ή / και false αν είναι αριθμός.

Αν είναι σύμβολο πράξης αφαιρούμε δύο στοιχεία απο την ουρά, τα αποθηκεύουμε στις μεταβλητές operand1 και operand2 και δομούμε την έκφραση που προσθέτουμε στην αρχή της ουράς.

Αν το item ΔΕΝ είναι σύμβολο πράξης τοτε το προσθέτουμε απλά στην αρχή της ουράς. Αφού τερματίσει η παραπάνω επανάληψη επιστρέφουμε το πρώτο στοιχείο της ουράς που είναι και η έκφραση σε ενθεματική μορφή.

Αν τελίκα το isValid επιστρέψει false τότε εκτυπώνουμε στον χρήστη μήνυμα ‘Invalid prefix expression’ και τερματίζουμε το πρόγραμμα.

ΥΛΟΠΟΙΗΣΗ ΜΕΡΟΥΣ Γ (DNA PALINDROME):

Το τρίτο μέρος της εργασίας υλοποιήθηκε ως εξής:

Στη main μέθοδο δημιουργήσαμε εναν scanner και ζητήσαμε απο τον χρήστη μια ακολουθία DNA αποθηκεύοντας την στην μεταβλήτη ‘dna’. Δημιουργήσαμε ενα αντικείμενο της DNAPalindrome κλάσης και αφού αρχικοποιήσαμε ενα flag με true ξεκινάμε μια επανάληψη οσο το μήκος του dna και ελέγχουμε αν η ακολουθία που μας έδωσε ο χρήστης είναι έγκυρη (αν δηλαδή περιέχει μόνο A, T, C και G, χωρίς κενά, μικρά γράμματα, αριθμούς και σύμβολα) και αν δεν ειναι θέτουμε το flag σε false, σταματάμε την επανάληψη και εκτυπώνουμε μήνυμα στον χρήστη ‘Invalid DNA Sequence’.

Αν το flag ειναι true καλούμε την μέθοδο isPalindrome.

Η μέθοδος isPalindrome είναι η βασική μέθοδος της υλοποιησής μας. Ξεκινάει με το να διαχωρίζει την μεταβλητή dna ανα γράμμα (νουκλεοτίδιο) και να την αποθηκεύει σε έναν πίνακα String (dnaArray). Επίσης, δημιουργεί ενα αντικείμενο της υλοποιήσης της ουράς για να χρησιμοποιηθεί στο πρόγραμμα. Στη συνέχεια, ξεκινάει μια επανάληψη που προσπελαύνει τον πίνακα dnaArray, ελέγχει ποιό απο τα τέσσερα νουκλεοτίδια ειναι στην θεση i, το μετατρέπει στο συμπλήρωμα του και το αποθηκεύει στο τέλος της ουράς. Έτσι, έχουμε στην ουρά την συμπληρωματική μορφή της ακολουθίας που μας δόθηκε απο τον χρήστη.

Άφου τελειώσει αυτή η διαδικασία αρχικοποιούμε μια μεταβλητή reverse τυπου String και με μια while ελέγχοντας αν η ουρά είναι άδεια σε κάθε επανάληψη προσθέτουμε τα στοιχεία της ουράς απο το πίσω άκρο της στην μεταβλητή reverse.

Τέλος, ελεγχουμε αν η ακολουθία που μας έδωσε ο χρήστης ταιριάζει με την αντίστροφη συμπληρωματική που δημιουργήσαμε και αν είναι ίδιες επιστρέφουμε true στην main. Αλλιώς, επιστρέφουμε false.

Αφού επιστραφεί το Boolean argument απο την μέθοδο isPalindrome αν ειναι true τοτε εκτυπώνουμε μήνυμα στον χρήστη ‘Palindrome’. Αλλίως, εκτυπώνουμε ‘Not a palindrome’.