ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 1η Εργασία 2023-2024

ΚΡΗΜΙΤΣΑΣ ΤΡΥΦΩΝ 3220258

ΚΩΝΣΤΑΝΤΙΝΟΣ ΖΗΒΑΛΑΣ 3210051

ΑΝΑΦΟΡΑ ΠΑΡΑΔΟΣΗΣ

Για την υλοποιήση του πρώτου ερωτήματος χρησιμοποιούμε τη κλάση Node με ορίσματα data , next , prev , όπου κρατάμε τα δεδομένα και δείχνουμε στον προηγούμενο και τον επόμενο κόμβο. Αρχικά ορίσαμε δύο μεταβλητές για τα άκρα (head και tail) της λίστας καθώς και ένα μετρητή που αυξάνεται η μειώνεται ανάλογα με τη μέθοδο που πραγματοποιείται (π.χ. αυξάνεται με την addFirst() ενώ μειώνεται με την removeLast()). Χρησιμοποιούμε τις μέθοδους: isEmpty() για να ελέγχουμε αν υπάρχουν στοιχεία στη λίστα.

Οι μέθοδοι εισαγωγής και εξαγωγής πραγματοποιούνται σε χρόνο O(1) καθώς δεν υπάρχει κάποια μορφή σάρωσης των στοιχείων της λίστας με χρήση επαναληπτικών μεθόδων , αλλά επιλέγεται ποιο στοιχείο θα εισαχθεί/διαγραφεί χρησιμοποιώντας τα δύο άκρα της λίστας τα οποία είναι ήδη ορισμένα. addFirst() και addLast() που προσθέτουν τον κόμβο που δέχονται ως όρισμα στην αρχή ή στο τέλος αντίστοιχα. removeFirst() και removeLast() που διαγράφουν τον κόμβο που βρίσκεται στην αρχή ή στο τέλος αντίστοιχα και επιστρέφουν τα δεδομένα του. getFirst() και getLast() επιστρέφουν τα δεδομένα της αρχής και τους τέλους της λίστας. printQueue εκτυπώνει τα δεδομένα της λίστας μας size() επιστρέφει ακέραια μεταβλητή και συγκεκριμένα τον μετρητή count που είναι η ακέραια μεταβλητή που εκφράζει το ακριβές μέγεθος της λίστας.

Οπότε πάλι αποφεύγεται η σάρωση της τελικής λίστας και έτσι ο χρόνος της είναι επίσης O(1).

Β Ο κώδικας δέχεται σαν είσοδο ένα όρισμα από τον χρήστη το οποίο αν είναι σε μορφή prefix είναι έγκυρη τη δέχεται και τη μετατρέπει σε μορφή infix, αν δε είναι έγκυρη τυπώνει μήνυμα λάθους. Αρχικά, ζητάει από τον χρήστη να εισάγει μια προθεματική παράσταση. Δημιουργεί μια διπλή ουρά χρησιμοποιώντας την υλοποίηση της (`StringDoubleEndedQueueImpl`) που υλοποιήσαμε στο πρώτο ερώτημα για να αποθηκεύσει τα στοιχεία της παράστασης. και διατρέχει την παράσταση από το τέλος προς την αρχή. Ελέγχει αν κάθε χαρακτήρας είναι τελεστής. Αν είναι, αφαιρεί δύο στοιχεία από την ουρά, τα συνενώνει με τον τελεστή και προσθέτει το αποτέλεσμα πίσω στην ουρά. Αν ο χαρακτήρας δεν είναι τελεστής, τότε πρέπει να είναι ένας αριθμός, ο οποίος προστίθεται στην ουρά. Ελέγχουμε αν η ουρά δεν περιέχει ακριβώς ένα στοιχείο στο τέλος, και στην περίπτωση που συμβαίναι αυτό τότε η παράσταση δεν είναι έγκυρη και εμφανίζεται ένα μήνυμα λάθους.

Τέλος, εμφανίζεται η ενθεματική παράσταση που αποθηκεύτηκε στην ουρά.

C. Αυτός ο κώδικας Java ελέγχει αν μια αλυσίδα DNA είναι μια ακολουθία Watson-Crick, που σημαίνει ότι είναι παλίνδρομη όταν αντικατασταθούν τα νουκλεοτίδια με τα συμπληρωματικά τους. Ζητάει από τον χρήστη να εισάγει μια αλυσίδα DNA. και μετατρέπει την αλυσίδα σε έναν πίνακα χαρακτήρων. Δημιουργεί μια διπλή ουρά χρησιμοποώντας την υλοποίηση της (`StringDoubleEndedQueueImpl`) που υλοποιήσαμε στο πρώτο ερώτημα για να αποθηκεύσει τα συμπληρωματικά νουκλεοτίδια.

Διατρέχει την αλυσίδα και προσθέτει το συμπληρωματικό νουκλεοτίδιο στην λίστα για κάθε νουκλεοτίδιο στην αλυσίδα. Αν βρει έναν χαρακτήρα που δεν είναι νουκλεοτίδιο ή αν υπάρχει κενό μεταξύ δύο νουκλεοτιδίων, εμφανίζει ένα μήνυμα λάθους και τερματίζει το πρόγραμμα. Συνενώνει όλα τα στοιχεία της ουράς σε μια συμπληρωματική αλυσίδα καλώντας τη μέθοδο removeFirst() που επιστρέφει και διαγράφει το πρώτο στοιχείο της ουράς.

Συγκρίνει την αρχική αλυσίδα με τη συμπληρωματική. Αν είναι ίδιες, τότε η αλυσίδα είναι μια ακολουθία Watson-Crick και εμφανίζει το αντίστοιχο μήνυμα. Αν δεν είναι ίδιες, εμφανίζει ένα μήνυμα ότι η αλυσίδα δεν είναι ακολουθία Watson-Crick. Η πολυπλοκότητα του αλγορίθμου είναι O(2N) δηλαδή γενικά O(N) αφού διατρέχει δυο for loop N στοιχείων