ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF INFORMATICS

8/12/2020

Δομές Δεδομένων

Εργασία 1 - Αναφορά παράδοσης

Υπεύθυνος Καθηγητής: Ευάγγελος Μαρκάκης

Χρήστος Μουταφτσής (p3190127)
Ιωάννα – Μαρία Παπακωνσταντίνου (p3190161)

Αθήνα

Ακαδημαϊκό Έτος 2020-2021

Χειμερινό Εξάμηνο

Μέρος a

Όλες οι λειτουργίες εισαγωγής/εξαγωγής και η μέθοδος size είναι σε O(1). Πιο συγκεκριμένα:

- Η μέθοδος **addFirst** προσθέτει ένα στοιχείο (κόμβο) στην αρχή της λίστας και αυτό γίνεται head (δεν υπάρχουν ψευδοκόμβοι) ενώ ταυτόχρονα το ως τώρα head γίνεται head.next (όταν υπάρχει τουλάχιστον ένα στοιχείο).

- Η μέθοδος **removeFirst** αφαιρεί το πρώτο στοιχείο της λίστας, το οποίο ονομάζεται head κάθε φορά. Ταυτόχρονα, όταν η λίστα έχει ήδη ένα στοιχείο, ως head τίθεται το ως τώρα επόμενο στοιχείο (δηλαδή το head.next).

- Η μέθοδος **addLast** προσθέτει ένα στοιχείο (κόμβο) στο τέλος της λίστας και αυτό γίνεται tail (δεν υπάρχουν ψευδοκόμβοι) ενώ ταυτόχρονα το ως τώρα tail γίνεται tail.prev (όταν υπάρχει τουλάχιστον ένα στοιχείο).

- Η μέθοδος **removeLast** αφαιρεί το τελευταίο στοιχείο της λίστας, το οποίο ονομάζεται tail κάθε φορά. Ταυτόχρονα, όταν η λίστα έχει ήδη ένα στοιχείο, ως tail τίθεται το ως τώρα προηγούμενο στοιχείο (δηλαδή το tail.prev).

- Η μέθοδος **getFirst** εμφανίζει το πρώτο στοιχείο της λίστας, το οποίο ονομάζεται head κάθε φορά, χωρίς να το αφαιρεί από την λίστα.

- Η μέθοδος **getLast** εμφανίζει το τελευταίο στοιχείο της λίστας, το οποίο ονομάζεται tail κάθε φορά, χωρίς να το αφαιρεί από την λίστα.

- Η μέθοδος **size** επιστρέφει το μέγεθος της λίστας(int size), το οποίο υπολογίζει αυξάνοντας ή μειώνοντας κάθε φορά την μεταβλητή size στις μεθόδους addFirst, addLast και removeFirst, removeLast αντίστοιχα.

Γενικά σε όλες τις μεθόδους οι εντολές που υπάρχουν είναι O(1) (δεν υπάρχουν βρόγχοι) άρα και οι μέθοδοι είναι O(1), δηλαδή εκτελούνται σε χρόνο ανεξάρτητο από τον αριθμό των αντικειμένων που μπορεί να είναι μέσα στην ουρά.

Μέρος b

Στο μέρος Β, και συγκεκριμένα στο PostfixToInfix.java διαβάζεται από τον χρήστη μια μεταθεματική παράσταση (postfix) σε String, ελέγχεται για την εγκυρότητά της και αν είναι έγκυρη τοποθετείται στη διπλά συνδεδεμένη λίστα list και τέλος εμφανίζεται στον χρήστη η αντίστοιχη ενθεματική παράσταση (infix). Αν δεν είναι έγκυρη εμφανίζεται το μήνυμα λάθους The expression is not valid. Επίσης στην main υπάρχουν οι κατάλληλες εντολές import και ένα try - catch block.

Αναφορικά με την εγκυρότητα ελέγχονται τα εξής:

- Τα στοιχεία που περιέχονται στην μεταθεματική παράσταση θα πρέπει να είναι μόνο ακέραιοι αριθμοί μεταξύ του 1 και του 9 ($\mathbf{x} \in [1,9]$).
- Η μεταθεματική παράσταση θα πρέπει τερματίζει με τελεστή και τα δύο πρώτα στοιχεία της να είναι τελεστέοι (εδώ ακέραιοι από το 1 μέχρι και το 9).
- Το πλήθος των τελεστέων θα πρέπει να είναι ίσο με το πλήθος των τελεστών συν ένα (πλήθος τελεστέων = πλήθος τελεστών + 1).
- Επειδή με τα παραπάνω δεν ελέγχονται περιπτώσεις όπως η 23*/65- (δηλαδή περιπτώσεις που τηρούνται τα παραπάνω αλλά οι τελεστές δεν είναι σε σωστά σημεία) δημιουργούμε έναν μετρητή total που αρχικοποιείται με μηδέν και λειτουργεί ως εξής:
  - ❖ Όταν συναντά αριθμό (τελεστέο) αυξάνει κατά 1 και
  - ❖ Όταν συναντά τελεστή μειώνεται κατά δύο και στη συνέχεια αυξάνεται κατά ένα.

    Αν ο μετρητής total είναι στο τέλος ίσος με ένα και δεν έγινε ποτέ μηδέν η παράσταση είναι έγκυρη.

Αναφορικά με την εισαγωγή στη διπλά συνδεδεμένη λίστα list γίνονται τα εξής:

- Ελέγχεται αν το κάθε στοιχείο (δηλαδή το elm) είναι τελεστής ή

τελεστέος.

- Αν είναι τελεστέος προστίθεται στο τέλος της λίστας με list.addLast(elm);

- Αν είναι τελεστής (+,-,*,/) αποθηκεύονται τα δύο τελευταία στοιχεία της λίστας στις μεταβλητές hf και hf1 και αφαιρούνται από την λίστα με τις μεθόδους getLast και removeLast. Στη συνέχεια κατασκευάζεται ένα String με παρενθέσεις, το hf, το hf1 και το elm και προστίθεται στο τέλος της λίστας με το addLast. Τέλος εμφανίζεται η λίστα στον χρήστη.

Μέρος c

Στο Μέρος Γ, και συγκεκριμένα στο DNAPalindrome.java εισάγεται από τον χρήστη μια String ακολουθία νουκλεοτιδίων DNA και σε πρώτο βήμα ελέγχεται για την εγκυρότητα της. Εάν είναι έγκυρη, τότε κάθε χαρακτήρας αυτής της String ακολουθίας εισάγεται μέσα σε ένα αντικείμενο ουράς (queue) από το Μέρος Α. Εάν δεν είναι έγκυρη, τυπώνεται μήνυμα λάθους, συγκεκριμένα Invalid input. Έπειτα, δημιουργείται μια νέα String ακολουθία (newqueue) η οποία είναι αρχικά κενή και τελικά γίνεται ίση με το αντεστραμμένο συμπλήρωμα της αρχικής ακολουθίας. Τέλος, γίνεται έλεγχος για το αν η newqueue ταυτίζεται με την αρχική ακολουθία, και αν αυτό ισχύει τότε τυπώνεται το σχετικό μήνυμα, δηλαδή ότι η ακολουθία νουκλεοτιδίων DNA που εισήχθη από τον χρήστη είναι πράγματι μία Watson-Crick συμπληρωματικά παλίνδρομη ακολουθία. Εάν δεν ισχύει αυτό, τότε τυπώνεται ότι η εν λόγω ακολουθία δεν είναι Watson-Crick συμπληρωματικά παλίνδρομη. Επιπλέον, στο πρόγραμμα υπάρχουν οι απαραίτητες εντολές import και στην main ένα try-catch block.

Αναφορικά με τον έλεγχο εγκυρότητας γίνεται το εξής:

- Για να είναι η εισαγόμενη String ακολουθία μία έγκυρη DNA ακολουθία, πρέπει ο κάθε χαρακτήρας της να είναι κάποιος από τους ακόλουθους κεφαλαίους λατινικούς: A, T, C, G (οποιοσδήποτε άλλος χαρακτήρας, όπως μη κεφαλαία ελληνικά ή λατινικά γράμματα, αριθμοί, σύμβολα και ενδιάμεσα κενά καθιστούν την ακολουθία μη έγκυρη). Επιπλέον, το κενό string ικανοποιεί την ιδιότητα, μπορεί να θεωρηθεί δηλαδή έγκυρη DNA ακολουθία.

Αναφορικά με τον έλεγχο για το αν είναι συμπληρωματικά παλίνδρομη γίνεται το εξής:

- Αν δεν είναι έγκυρη η ακολουθία DNA, φυσικά δεν είναι και συμπληρωματικά παλίνδρομη.
- Αν είναι έγκυρη η ακολουθία DNA, προχωράμε στον έλεγχο για το αν είναι και Watson-Crick συμπληρωματικά παλίνδρομη. Στην περίπτωση που το μήκος της είναι ίσο με 0, θεωρείται συμπληρωματικά παλίνδρομη. Αν όμως είναι ίσο με 1, δεν ικανοποιείται η ιδιότητα. Τέλος, αν το μήκος της είναι μεγαλύτερο

του 1, τότε χτίζεται το συμπλήρωμά της και ταυτόχρονα αντιστρέφεται. Εάν το αντεστραμμένο συμπλήρωμα της ακολουθίας ταυτίζεται με την αρχική μορφή αυτής, τότε αυτή θεωρείται συμπληρωματικά παλίνδρομη.

<u>Αναφορικά με τις απαιτήσεις πολυπλοκότητας:</u>

- Έστω ότι εισάγεται από τον χρήστη μια ακολουθία DNA με N νουκλεοτίδια. Στο πρόγραμμα υπάρχουν 2 loop τύπου for, τα οποία ως γνωστό εκτελούνται σε χρόνο O(N) το καθένα. Όλες οι υπόλοιπες εντολές του προγράμματος εκτελούνται σε σταθερό χρόνο, έστω c. Οπότε, σχετικά με τον συνολικό χρόνο εκτέλεσης του προγράμματος, έχουμε ότι:

O(N) + O(N) + O(c) = O(N + N + c) = O(2N + c) = O(N), καθώς:
2N + c <= N, για κάθε N >= c.

Επομένως, το πρόγραμμα μας πράγματι εκτελείται σε χρόνο O(N).

- Για την επεξεργασία της ουράς χρησιμοποιείται, όπως ζητείται, μόνο 1 loop, τύπου for.

Μέρος d

- Η υλοποίηση γίνεται με χρήση generics.

- Στο πρόγραμμα **PostfixToInfix.java** η μεταθεματική παράσταση θα πρέπει να δίνεται χωρίς κενά μεταξύ των στοιχείων (δηλαδή 42+ όχι 4 2 +) γιατί αλλιώς εμφανίζεται μήνυμα λάθους

  όπως και στο πρόγραμμα **DNAPalindrome.java** (ATAT όχι A T A T) μόνο με αγγλικούς κεφαλαίους χαρακτήρες (δηλαδή πχ αγγλικό A όχι ελληνικό).

  Ο *κενός χαρακτήρας* δίνεται πατώντας *μόνο enter* στο πληκτρολόγιο.

  Αναλυτικές λεπτομέρειες για το πώς γίνονται οι έλεγχοι δίνονται **στις παραπάνω σελίδες.**

- *Όλα* τα μέρη **μπορούν να τρέξουν** και σε cmd και σε Eclipse.

- Τα δύο προγράμματα *τερματίζονται μετά την επεξεργασία* κάθε παράστασης που δίνεται. Δεν συνεχίζουν να τρέχουν μέχρι ο χρήστης να αποφασίσει ότι τερματίζουν.