Δομές Δεδομένων

Εργασία 1

Λυδία Τασουλή- Νόνικα, 3180182

Ορέστης Παπαδόπουλος, 3190158

Μέρος Α και Γ:

Την εργασία την υλοποιήσαμε με χρήση Generics ώστε να μην υπάρχει πρόβλημα με τη συμβατότητα των τύπων των δεδομένων καθώς και με σκοπό την εξοικείωση με μια πιο γενικευμένη μέθοδο υλοποίησης τέτοιου είδους αλγορίθμων. Για αρχή, φτιάξαμε μια κλάση Node και για το Α' μέρος αλλά και για το Γ' μέρος με δυο constructor ένα για την FIFO και ένα για την LIFO αντίστοιχα. Τις StringStackImpl, StringQueueWithOnePointer και StringQueueImpl τις υλοποιήσαμε σύμφωνα με τα interface που δόθηκαν. Για το Α' μέρος η υλοποίηση των μεθόδων έγινε με βάση των κώδικα των διαλέξεων και των φροντιστηρίων. Ουσιαστικά είναι υλοποιημένες με λίστα μονής σύνδεσης και η ουρά και η στοίβα.

Για το Γ μέρος χρησιμοποιήσαμε κυκλική λίστα μονής σύνδεσης με σκοπό να έχουμε μόνο έναν pointer. Στη συγκεκριμένη περίπτωση είναι το pointer = rear; και ουσιαστικά το rear είναι το τελευταίο στοιχείο της ουράς δηλαδή rear = tail;, το οποίο όμως δείχνει στο πρώτο στοιχείο οπότε έχουμε ότι το head = rear.next; με αυτό το σκεπτικό υλοποιήσαμε το Γ μέρος και τις μεθόδους του ώστε να λειτουργεί σαν την ουρά στο Α' μέρος αλλά με τη χρήση ενός δείκτη. Επίσης, οι υλοποιήσεις είναι παρόμοιες με την StringQueueImpl απλά προσαρμοσμένες στα δεδομένα τις StringQueueWithOnePointer.

Μέρος Β:

Για το μέρος Β ξεκινάμε διαβάζουμε το αρχείο με τη μέθοδο BufferedReader και διαβάζοντας την πρώτη γραμμή αποθηκεύουμε τα m x n για να φτιάξουμε τον πίνακα, αντίστοιχα με το διάβασμα της δεύτερης γραμμής αποθηκεύουμε στα int x, y; τις συντεταγμένες της εισόδου. Ύστερα φτιάχνουμε ένα πίνακα χαρακτήρων και τον αρχικοποιούμε με τα m,n char[][] details = new char[m][n]; και ύστερα καλούμαι την StringStackImpl η οποία είναι τύπου Point(εξήγηση παρκάτω). Μετά εισάγουμε τα δεδομένα του text στον πίνακα καθώς και κάνουμε έλεγχο για τα στοιχεία εισόδου αν είναι σωστά πχ για τις Στήλες αν ισούται με n και αν υπάρχει είσοδος μια, καμία η πολλές. Για το σκοπό της εργασίας έχουμε φτιάξει μια κλάση point() που για κάθε details[m][n] κρατάει τις συντεταγμένες x,y και το value= 0 ή 1. Επίσης έχουμε φτιάξει ένα πίνακα Boolean[][] visited; που για κάθε δεδομένο αποθηκεύει true αν το συγκεκριμένο δεδομένο το έχουμε ελέγξει ήδη. Το πρόγραμμα λοιπόν λειτουργεί ως εξής ξεκινάμε κάνοντας push

της είσοδο για να μην είναι άδεια η στοίβα, μετά την κάνουμε pop() και ελέγχουμε για κάθε γείτονα της (πάνω, κάτω, δεξιά, αριστερά) αν είναι εντός των ορίων του πίνακα και αν visited() = false . Αν αυτά είναι true τότε ελέγχουμε αν η value = 0 αν είναι 0 κάνουμε push αλλιώς κάνουμε το visited() = true ώστε να μην το ξανά ελέγξει και το αφαιρούμε εντελώς από τη στοίβα. Αυτό συμβαίνει μέχρι η στοίβα να είναι άδεια η να βρει έξοδο που αυτό το ελέγχουμε με το αν υπάρχει 0 στα άκρα και δεν είναι η είσοδος. Ο αλγόριθμος ουσιαστικά κάνει backtracking διότι όταν βρίσκεται σε αδιέξοδο μαρκάρει όλους τους γείτονες που το value=1 σαν visited και ύστερα ψάχνει τα value=0 που δεν έχει κάνει visited και έτσι διαλέγει άλλο μονοπάτι.