Αλγόριθμοι

Προγραμματιστική εργασία 1

Δημήτρης Φιλίππου p3160253

Εισαγωγή

Η δομή του project καθώς και ο τρόπος με τον οποίο γίνονται compile τα αρχεία java είναι σχεδιασμένα να τρέχουν εύκολα και γρήγορα σε μηχανήματα Mac (UNIX) χωρίς αυτό να σημαίνει ότι δεν τρέχουν σε περιβάλλον Windows. Συγκεκριμένα όσο αφορά τη δομή του project, υπάρχει ο φάκελος "src" ο οποίος περιέχει όλο τον πηγαίο κώδικα, ο φάκελος data που περιέχει δεδομένα απαραίτητα για την τροφοδοσία των προγραμμάτων και ο φάκελος build που δημιουργείται αυτόματα από το compiler script. Το αρχείο "compiler.sh" (compiler script) δεν κάνει τίποτα άλλο παρά να δημιουργεί τον φάκελο build, να κάνει compile όλο τον φάκελο "src" και να μετακινεί τα compiled class αρχεία εκεί. Το αρχείο "shortcuts.sh" περιέχει σύντομα μονοπάτια για γρήγορη εκτέλεση σε περιβάλλον UNIX. Για παράδειγμα για να τρέξει το πρόγραμμα σε περιβάλλον UNIX χρησιμοποίησα τις παρακάτω εντολές.

1. source shortcuts.sh
2. compile
3. one

Η τρίτη εντολή τρέχει την πρώτη άσκηση και η εντολή "two" τρέχει την δεύτερη άσκηση αντίστοιχα. Σε μηχανήματα Windows το compile μπορεί να γίνει κανονικά στον φάκελο "src". Στο project υπάρχουν τα εξής αρχεία:

1. One.java
2. QuickSort.java
3. Result.java
4. Splitter.java
5. Two.java
6. Utilities.java

Τα αρχεία "One" και "Two" αποτελούν την πρώτη και την δεύτερη υλοποίηση των ασκήσεων αντίστοιχα ενώ όλα τα υπόλοιπα αρχεία είναι βοηθητικά και χρησιμοποιούνται αποκλειστικά για την καλύτερη οργάνωση του project και πιο καθαρό κώδικα.

Άσκηση 1

Στην πρώτη άσκηση, η λύση του προβλήματος προσεγγίστηκε και υλοποιήθηκε σε χρόνο O(logn) με τη μέθοδο της διάσπασης σε επιμέρους στοιχεία. Εφόσον γνωρίζω ότι υπάρχει είσοδος ταξινομημένων στοιχείων, τότε θα ορίζω ως pivot την μέση της λίστας και θα γίνεται αναδρομικά η κλήση έτσι ώστε να βρεθεί το πρώτο και το τελευταίο στοιχείο σε μία λίστα. Η υλοποίηση σε κώδικα βρίσκεται στο αρχείο "Splitter.java" στο οποίο υπάρχουν οι εξής μέθοδοι:

Method	Scope	Returns	Complexity
searchFirstLast	public	Result	$O(logn)$
first	private	Integer	$O(logn)$
last	private	Integer	$O(logn)$
toIntArray	private	Integer []	$O(n)$

Η μέθοδος "searchFirstLast" έχει ως είσοδο μία λίστα ακεραίων αριθμών επομένως για να λειτουργήσει σωστά το πρόγραμμα, γίνεται μετατροπή σε array από ακέραιους. Αυτή η μετατροπή γίνεται σε γραμμικό χρόνο όμως δεν είναι μέρος του αλγορίθμου οπότε δεν αποτελεί ιδιαίτερη ανησυχία. Αφού έχουμε έτοιμο το array, δημιουργούνται 2 λογικές μεταβλητές οι οποίες ελέγχουν μονάχα μία φορά εάν μπορούμε να δώσουμε άμεση απάντηση σε $O(1)$ χωρίς να τρέξει ο αναδρομικός αλγόριθμος. Συγκεκριμένα ο πρώτος έλεγχος λέγεται "edge match" και ελέγχει εάν οι άκρες του array έχουν ίδια δεδομένα, και εάν αυτό ισχύει, τότε η απάντηση του προβλήματος θα είναι το πρώτο στοιχείο του array και το τελευταίο. Ο δεύτερος έλεγχος ονομάζεται "is empty" και ο ρόλος του είναι να κοιτάει εάν το array είναι άδειο, όπου δημιουργεί ένα νέο error λέγοντας στον χρήστη το πρόβλημα. Η συνάρτηση "first" και "last" είναι και οι δύο $O(n)$ οπότε $O(n) + O(n) = O(n)$.

Άσκηση 2

Στην δεύτερη άσκηση υλοποιήθηκε η παραλλαγή της QuickSort χρησιμοποιώντας 3 μέρη. Η άσκηση είναι υλοποιημένη στο αρχείο QuickSort.java το οποίο περιέχει μία public μέθοδο η οποία καλεί τις private για να γίνει η ταξινόμηση με ασφάλεια. By default η συνάρτηση δέχεται και ταξινομεί ολόκληρο array και όχι μέρος αυτού ου array πράγμα που μπορεί να αλλάξει με minor tweaks. Τέλος υπάρχει η συνάρτηση swap η οποία ασυμπτωτικά είναι $O(1)$