Algorithms
Programming exercise 1

Κωνσταντίνος Νικολούτσος
p3170122

1 Ασκηση

1.1 Πολυπλοκοτητα χρόνου

Για την επίλυση της συγκεκριμένης άσκησης χρησιμοποιήθηκε η γνωστή σε ολους μας *binary search.* Πιο συγκεκριμένα, η πολυπλοκοτητα καλύτερης,χειρότερης και μέσης περίπτωσης ειναι : $\Theta(\log n)$.
Πιο αναλυτικά η πολυπλοκότητα ειναι αυτη διοτι αφου εχουμε κανει 3 *binary searches* μεχρι το τελος, εχουμε $\Theta(3 \log n) := \Theta(\log n)$.

1.2 Ιδέα υλοποίησης

Αρχικά βρίσκουμε με μια απλη binary search μια εμφανιση στοιχειου που ψαχνουμε. Επειτα, καλουμε τις μεθοδους *findFirstOccurrence(), findLastOccurrence()* οι οποιες βρισκουν την αριστεροτερη και την δεξιοτερη εμφανιση του ζητουμενου αριθμου.
Ετσι λοιπον στο τελος του αλγορίθμου εχουμε τα index/offset της πρώτης και τελευταιας εμφανισης του ζητουμενου αριθμου. Αξιζει να σημειωθεί μάλιστα οτι στην περίπτωση που ψαχνουμε αριθμου που δεν υπαρχει, ο αλγοριθμος επιστρεφει την σταθερα *ITEM_NOT_FOUND* η οποια ειναι ιση με -1.

1.3 Συμπεράσματα

΄Οταν έχουμε τα στοιχεία ταξινομημένα σε εναν πίνακα τοτε πρέπει να εκμεταλευόμαστε αυτην την ιδιότητα οσο μπορούμε. Για παράδειγμα οπως είδαμε και σε αυτην την άσκηση ειναι "κακό" να κανουμε linear search με πολυπλοκότητα χειρότερης περίπτωσης $\mathcal{O}(n)$ αφου μπορουμε να το κανουμε σε $\mathcal{O}(\log n)$.

2 Ασκηση

2.1 Πολυπλοκοτητα χρόνου

Στην μικρή παραλλαγή αυτη της κανονικής *quicksort* η πολυπλοκότητα χρόνου που προκύπτει ειναι:

• **καλύτερη περίπτωση:** $\Omega(n \log n)$

• **μέση περίπτωση:** $\Theta(n \log n)$

• **χειρότερη περίπτωση:** $\mathcal{O}(n^2)$

Αυτο μπορεί να δειχθεί εύκολα χρησιμοποιώντας το *master theorem* που είδαμε στο μάθημα.

2.2 Ιδέα υλοποίησης

Η κεντρική ιδεα ειναι αρκετά ίδια με της κανονικής *quicksort* αλλα αντί να χωρίζουμε τα στοιχεία σε δυο groups(μεγαλυτερα/μικροτερα του *Pivot*) τωρα τα σπάμε σε 3 ομαδες των μικρότερων,ισων και μεγαλύτερων.
Φυσικά το στοιχείου του recursion δεν λείπει απο τον αλγοριθμο αυτο αφου πανω σε αυτον στηρίζεται ολη η υλοποίηση. Τελος, η επιλογή του pivot γινεται ψευδοτυχαια με την βοήθεια του *java.util.Random* . Για περισσοτερα ανατρέξτε στο source code!

2.3 Συμπεράσματα

Ο αλγοριθμος ειναι αναδρομικος και “γρηγορος” .Το ποσο γρηγορος βεβαια ειναι σχετικο,γιατι αν του δωσεις εναν πινακα που ειναι ”αναποδα” ταξινομημενος, τοτε γινεται αργος.