## *ΑΛΓΟΡΙΘΜΟΙ – 1η Σειρά Προγραμματιστικών Ασκήσεων*
## *Μαρίνα Ντόγκα (3160119)*

## ΆΣΚΗΣΗ 1.1

Στην άσκηση αυτή ζητείται να βρεθούν οι θέσεις της πρώτης και τελευταίας εμφάνισης ενός αριθμού x με χρήση αλγορίθμου που θα εκτελείται σε χρόνο O(logn).

Ο αλγόριθμος θα βασιστεί στην δυαδική αναζήτηση. Για τον λόγο αυτό δημιουργούμε τις μεθόδους firstOccurence() και lastOccurence() όπου και οι δύο παίρνουν ως παραμέτρους τον ταξινομημένο πίνακα array, τα άκρα L και H του πίνακα, τον αριθμό x για τον οποίο ψάχνουμε την πρώτη και τελευταία εμφάνιση του στον πίνακα και το μέγεθος του πίνακα N. Οι δύο αυτές μέθοδοι θα υλοποιηθούν στην κλάση DQ_util.

Η μέθοδος firstOccurence() θα βρει την πρώτη φορά που εμφανίζεται ο αριθμός x. Έτσι αρχικά εφόσον δεν έχει αναζητηθεί όλες οι θέσεις του πίνακα με βάση τα άκρα H και L (αν δηλαδή δεν ισχύει H < L) προχωράμε στην υλοποίηση του αλγορίθμου. Αρχικά βρίσκουμε τον δείκτη που αντιστοιχεί στην τιμή της διαμέσου και στην συνέχεια εκτελούμε ελέγχους με βάση αυτή την πληροφορία. Οι έλεγχοι είναι οι εξής:

- Αν ο δείκτης είναι 0 ή το x είναι μεγαλύτερο από την προηγούμενη τιμή της διαμέσου και (με την προϋπόθεση ότι ισχύει το ένα εκ των δύο προηγουμένων) το x ισούται με την διάμεσο τότε επιστρέφουμε τον δείκτη.
- Αν το x είναι μεγαλύτερο από την διάμεσο τότε γίνεται αναδρομική κλήση της μεθόδου στις θέσεις [δείκτης+1, H].
- Αν το x είναι μικρότερο από την διάμεσο τότε γίνεται αναδρομική κλήση της μεθόδου στις θέσεις [L, δείκτης-1].

Η μέθοδος lastOccurence() θα βρει την τελευταία φορά που εμφανίζεται ο αριθμός x. Έχει την ίδια λειτουργία με την προηγούμενη μέθοδο με την εξής διαφορά:

- Αν ο δείκτης ισούται με το (μέγεθος του πίνακα – 1) ή αν το x είναι μικρότερο από την επόμενη τιμή της διαμέσου και (με την προϋπόθεση ότι ισχύει το ένα εκ των δύο προηγουμένων) το x ισούται με την διάμεσο τότε επιστρέφουμε τον δείκτη.

Η πολυπλοκότητα του αλγορίθμου είναι O(logn) διότι κάθε φορά θα πρέπει να διαιρέσουμε τον πίνακα στα δύο (N/2) k φορές. Στην περίπτωση που η τιμή x δεν είναι η διάμεσος θα ψάξουμε στο αντίστοιχο μισό του πίνακα ([L, δείκτης-1] ή [δείκτης, H]) και στην συνέχεια θα ξαναχωρίσουμε τον υποπίνακα στα 2 και θα ψάξουμε ανάλογα. Αυτό γίνεται με λογαριθμικό τρόπο. Για N στοιχεία έχουμε ότι:

$$n \times \left(\frac{1}{2}\right)^k = 1 \Leftrightarrow \frac{n^k \times 1}{2^k} = 2^k \Leftrightarrow n = 2^k \Leftrightarrow \log_2 n = k$$

από όπου και προκύπτει ο αρχικός μας ισχυρισμός ότι ο αλγόριθμος εκτελείται σε λογαριθμικό χρόνο.

## **ΑΣΚΗΣΗ 1.2**

Στην άσκηση αυτή ζητείται μια παραλλαγή της μεθόδου ταξινόμησης Quicksort που βασίζεται στην τριμερή διαμέριση και στην επιλογή τυχαίου pivot.

Στην κλάση Quicksort_util θα υλοποιηθεί ο αλγόριθμος αυτός και οι μέθοδοι που πρέπει να υλοποιηθούν είναι οι:

- quicksort(): παίρνει ως παραμέτρους τον πίνακα και το διάστημα στο οποίο θα γίνει η ταξινόμηση (δείκτες L και H). Χρησιμοποιείται για την εκτέλεση της ταξινόμησης.
- partition(): ίδιες παράμετροι με την quicksort. Χρησιμοποιείται για την τριμερή διαμέριση του πίνακα με την χρήση του τυχαίου του pivot.
- swap(): παίρνει ως παραμέτρους τον πίνακα και τις θέσεις με βάση των οποίων θα γίνει η ανταλλαγή των στοιχείων.

Η μέθοδος quicksort() χρησιμοποιεί αρχικά την μέθοδο partition για να κάνει την τριμερή διαμέριση και θα επιστρέψει το τυχαίο pivot με το οποίο θα εκτελεστεί αναδρομικά η ταξινόμηση quicksort.

Η μέθοδος partition() αρχικά θα επιλέξει το τυχαίο pivot βρίσκοντας έναν τυχαίο δείκτη με την χρήση της Random. Η τιμή αυτή με χρήση της swap θα πάει στην πρώτη θέση του πίνακα και με βάση αυτόν θα γίνει η ταξινόμηση χρησιμοποιώντας τριμερή διαμέριση. Δημιουργούμε, λοιπόν, έναν μετρητή i που θα ξεκινάει από την επόμενη θέση του pivot. Όσο ο μετρητής αυτός δεν φτάσει το όριο H θα γίνει κατάλληλη ανταλλαγή στοιχείων αν το pivot είναι διάφορο του στοιχείου του πίνακα στην θέση i και αύξηση του μετρητή i αν ισχύει η ισότητα array[i] = pivot. Για την ανισότητα γίνονται τα εξής:

- Αν array[i] < pivot τότε γίνεται αντιμετάθεση των γειτονικών στοιχείων του πίνακα
- Αν array[i] > pivot τότε γίνεται αντιμετάθεση του στοιχείου που βρίσκεται στην θέση I με αυτή που βρίσκεται στην θέση πριν το όριο H (H--).

Ο αλγόριθμος εκτελείται σε χρόνο Θ(nlogn) όπως η κλασική μέθοδος quicksort στην μέση περίπτωση και με Θ($n^2$) χρόνο στην χειρότερη περίπτωση. Για την εύρεση του O(nlogn) θα πρέπει να κοιτάξουμε την εκτέλεση της μεθόδου quicksort(). Η partition() δηλαδή η τριμερής διαμέριση θα εκτελεστεί σε γραμμικό χρόνο (O(n)) εφόσον διατρέχουμε όλο τον πίνακα ώστε να γίνουν οι κατάλληλες αντιμεταθέσεις. Η επιλογή του τυχαίου pivot και οι αντιμεταθέσεις γίνονται σε σταθερό χρόνο και εκτελούνται n φορές. Οι 2 αναδρομικές κλήσεις της μεθόδου quicksort εκτελούνται σε υποπίνακα μεγέθους n/2 του αρχικού πίνακα. Αυτο γίνεται σε Θ(logn) χρόνο. Αυτό συνδυασμένο με το ότι οι διαμερίσεις εκτελούνται σε Θ(n) καταλήγει στο ότι ο αλγόριθμος θα εκτελεστεί σε Θ(nlogn) χρόνο.