Εργασία Δομών Δεδομένων

Χαράλαμπος Μυσιρλίδης – ΑΜ 0949

Μεταπτυχιακό ΕΤΥ

# Εισαγωγή Θέμα 1ο

Στο συγκεκριμένο θέμα έπρεπε να υλοποιήσουμε τους αλγόριθμους ταξινόμηση αλλά και αυτούς της αναζήτησης που μας ζητάει το πρώτο θέμα.

Οι αλγόριθμοι αναζήτησης είναι οι

* BinarySearch
* InterpolationSearch

Ενώ για τους αλγορίθμους ταξινόμησης έχουμε τους:

* QuickSort
* MergeSort
* **InsertionSort**

Ο τελευταίο αποτελεί επιλογή δική μου σαν αλγόριθμος ταξινόμησης από το βιβλίο. Και αυτό γιατί έχει μεγαλύτερη πολυπλοκότητα από τους άλλους ενώ παράλληλα είναι εύκολος στην υλοποίηση.

Επιπλέον οι συγκεκριμένοι αλγόριθμοι ελέγχθηκαν με την χρήση είτε τυχαίων πινάκων είτε από ένα σύνολο ακεραίων μέσα από ένα αρχείο. Στην πρώτη περίπτωση δημιουργούσαμε τυχαίους πίνακες και τους ταξινομούσαμε αντίστροφα έτσι ώστε να αποτελούν η χειρότερη περίπτωση για τους αλγορίθμους QuickSort – InsertionSort μιας και ο πρώτος έχει σαν χειρότερη περίπτωση O(n2) ενώ καλύτερη O(nlogn). Από την άλλη η InsertionSort έχει καλύτερη περίπτωση O(n) αλλά αυτό ισχύει μόνο στην περίπτωση που είναι ήδη ταξινομημένος ο πίνακας μας και στην πραγματικότητα η μέση περίπτωση είναι O(n2) και η • MergeSort εγγυάτε O(nlogn) για όλες τις περιπτώσεις. Τέλος για να είμαστε σίγουροι και για τους δυο αυτούς αλγορίθμους οι μετρήσεις των συγκρίσεων αλλά και του χρόνου έγινε με πίνακες αντίστροφα ταξινομημένους.

# Απόδοση αλγορίθμων ταξινόμηση

Όλοι οι πίνακες που ακολουθούν είναι ήμι-λογαριθμικοί (μόνο Y-axis logarithmic). Υπάρχει μια περίπτωση που οι συγκρίσεις είναι πάρα πολλές και χρειάστηκε να κάνουμε και διαίρεση με χιλιάδα πίνακας 2ος κατά σειρά (αναφέρει πλάγια thousands).

Έγινε μια προσπάθεια να βρούμε το λάθος αλλά και την τυπική απόκλιση για να είμαστε στατιστικά σωστοί στα δείγματα που θα πάρουμε αλλά παρατηρήσαμε ότι με δείγματα της τάξης των δεκάδων στους τυχαίους πίνακες είχαμε αρκετά μεγάλο λάθος και τα άνω και κάτω όρια ήταν αρκετά μεγάλα. Σε αυτή την περίπτωση πείραμα δείγματα με μερικές δεκάδες και τα παρουσιάζω κάτω χωρίς τα άνω και κάτω όρια αλλά παρά μόνο με την μέση τιμή.

## Απόδοση από τυχαίου πίνακες

Όπως αναφέραμε πιο πριν η σύγκριση των παρακάτω αλγορίθμων οι πίνακες που κάναμε αναπαραγωγή είναι πίνακες χειρότερης περίπτωσης για τους παρακάτω αλγορίθμους. Για την InsertionSort & QuickSort αυτό είναι ο αντίστροφα ανεστραμμένος πίνακας.

Όπως ήταν λογικό την καλύτερη απόδοση την έχει ο αλγόριθμος MergeSort. Ενώ οι άλλοι δύο αλγόριθμοι λόγο τετραγωνικής πολυπλοκότητας στην χειρότερη περίπτωση παρατηρούμε ότι έχουν κατά πολύ μεγαλύτερο χρόνο εκτέλεσης ενώ παράλληλα αυτός ο χρόνος έχει και γραμμική εξάρτηση (εκτός φυσικά από την αρχή).

Την χειρότερη απόδοση την σε αυτή την περίπτωση έχει η InsertionSort και αυτό γιατί έχει την μέση περίπτωση O(n2). Γενικότερα η InsertionSort κρίνετε από μεγάλο αριθμό συγκρίσεων κατά την ταξινόμηση των στοιχείων ενός πίνακα. Από την άλλη η MergeSort εκτός από την χρήση βοηθητικής μνήμης (auxiliary memory) κάνει χρήση και αρκετών συγκρίσεων μεγάλη διαφορά από αυτή της InsertionSort. Τέλος η QuickSort παρουσιάζει τον μικρότερο αριθμό συγκρίσεων με διαφορά σε όλες τις περιπτώσεις. Αυτό που παρατηρείτε όπως και στον προηγούμενο πίνακα είναι ότι η σχέση παραμένει γραμμική.

Τέλος να υποσημειώσουμε αν πριν από κάθε ταξινόμηση η QuickSort έκανε Shuffle στις τιμές του πίνακα τότε η απόδοση του αλγορίθμου θα ήταν πάλι O(nlogn) αλλά με ένα μικρό overhead από το shuffle.

## Απόδοση από αρχείο

Στον παρακάτω πίνακα παρατηρούμε ότι ο αλγόριθμος με σταθερά καλύτερη απόδοση είναι αυτός του InsertionSort πιθανά γιατί οι πίνακες μας να είναι σχετικά ταξινομημένοι, έτσι παρατηρείτε η σταθερή απόδοση του αλγορίθμου. Όπως είναι εμφανές η απόδοση του αλγορίθμου QuickSort είναι καλύτερη από αυτή της MergeSort.

Ο αριθμός των συγκρίσεων έχει εκθετική αύξηση για όλες τις περιπτώσεις ενώ ο αριθμός των συγκρίσεων πλέον είναι μεγαλύτερος για αυτό της MergeSort ενώ η InsertionSort έχει κοντά ίδιο αριθμό με αυτό της QuickSort.