Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Δομές Δεδομένων και Αλγόριθμοι

Αναφορά 1ης άσκησης
Ομάδα 10

1)

	1a	1b	2a	2b	3a	3b
insert()	O(1)	O(N)	O(1)	O(N)	O(1)	O(N)
delete()	O(N)	O(N)	O(N)	O(N)	O(N)	O(N)
search()	O(N)	O(N)	O(N)	O(N)	O(N)	O(logN)

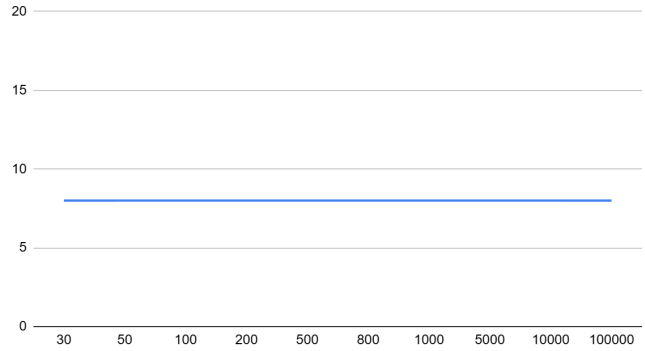
Για τις εισαγωγές στοιχείων σε μη ταξινομημένες δομές η πολυπλοκότητα θα ισούται με O(1) αφού το στοιχείο απλώς θα εισαχθεί στο τέλος της λίστας. Άρα για τα 1a, 2a, 3a, ισχύει O(1). Επιπλέον για την δομή 1a ισχύει ότι η πολυπλοκότητα για το delete και το search ισούται με O(N) αφού για να διαγράψουμε ή να επιστρέψουμε κάποιο στοιχείο πρέπει να διασχίσουμε την λίστα. Για την δομή 1b η πολυπλοκότητα της αναζήτησης και της διαγραφής είναι η ίδια με την 1a, αλλά για την εισαγωγή η πολυπλοκότητα είναι O(N) αφού για να γίνει η εισαγωγή στο σωστό σημείο πρέπει πάλι να διασχίσουμε τη λίστα. Αντίστοιχα, για την δομή 2a και 2b ισχύουν τα ίδια. Για την δομή 3a, η διαγραφή και η αναζήτηση θα έχουν O(N) πολυπλοκότητα αφού πάλι γίνεται διάσχιση της λίστας. Τέλος, όσον αφορά την υλοποίηση 3b, και οι 3 λειτουργίες χρησιμοποιούν δυαδική αναζήτηση (binary search) αλλά η διαφορά είναι ότι για insert και delete, μετά το binary search έχουμε μετατόπιση στοιχείων (shifts). Επομένως οι συνολικές πολυπλοκότητες για insert, delete θα είναι O (logN+N) = O (N), ενώ για την search, που απλά χρησιμοποιεί τη binary search θα έχουμε λογαριθμική πολυπλοκότητα O (logN).

2)
i**nsert() 2a:** Ο αριθμός των πράξεων παραμένει σταθερός για κάθε N, δείχνοντας σταθερή πολυπλοκότητα, ενώ ο χρόνος εκτέλεσης φαίνεται να μειώνεται ελαφρώς καθώς αυξάνεται το N.

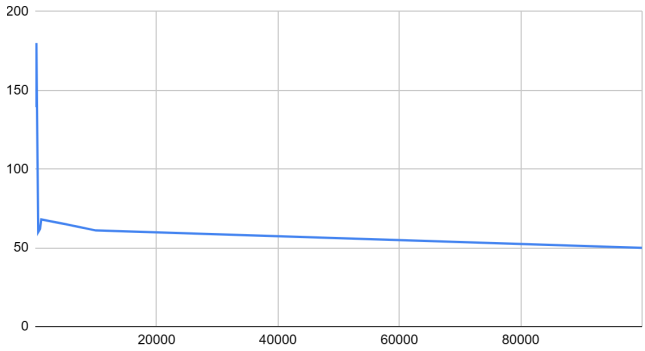
search() 2b: Οι πράξεις και ο χρόνος εκτέλεσης αυξάνονται γραμμικά, όπως άλλωστε περιμέναμε και από τη θεωρία.

search() 3b: Η μορφή της καμπύλης για τον αριθμό των πράξεων δείχνει μια πολύ μικρή σταδιακή αύξηση, που συνάδει με λογαριθμική πολυπλοκότητα.

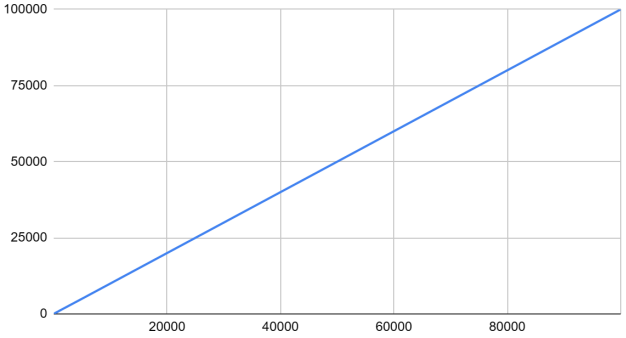
insert() 2a operations



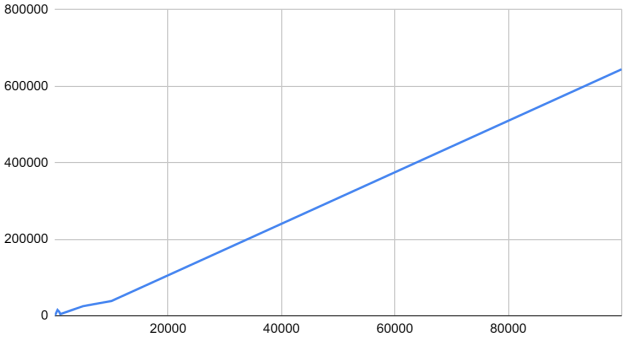
insert() 2a time



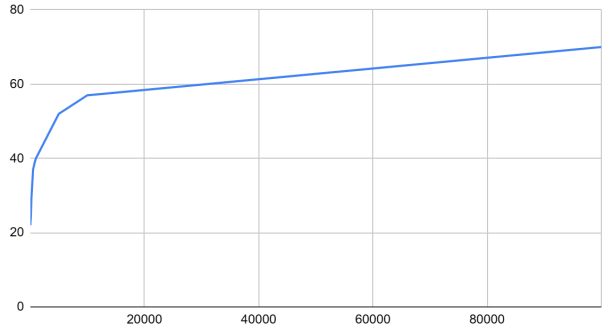
search() 2b operations



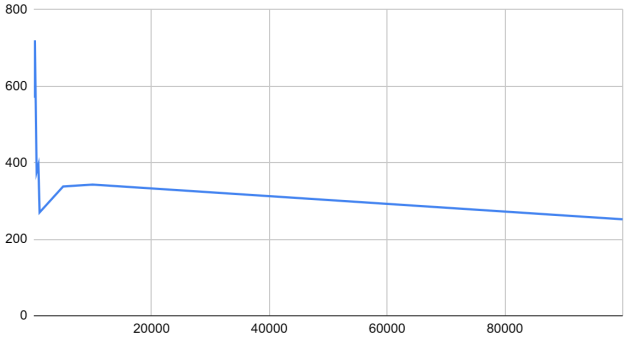
search() 2b time



search() 3b operations



search() 3b time



3) Ερωτήσεις

1. Ναι, με βάση τη θεωρία μας, στη θεωρία αλγορίθμων και των δομών δεδομένων, είναι συνηθισμένο οι χρόνοι εκτέλεσης να συμβαδίζουν με το πλήθος των πράξεων που εκτελούνται. Πράγματι, τόσο στις γραμμικές όσο και στη λογαριθμική πολυπλοκότητα, οι τιμές πράξεων και χρόνων εκτέλεσης ταυτίζονται όπως επίσης και για πολυπλοκότητα O(1) που έχουμε μικρό αριθμό πράξεων και σύντομο χρόνο εκτέλεσης.

2.

	1a	1b	2a	2b	3a	3b
insert()	O(N)	O(N)	O(N)	O(N)	O(N)	O(N)
delete()	O(N)	O(N)	O(N)	O(N)	O(N)	O(N)
search()	O(N)	O(N)	O(N)	O(N)	O(N)	O(logN)

Η διαφορά που θα υπάρχει στους πίνακες είναι στο insert των μη ταξινομημένων λιστών αφού αν επιβάλλεται να υπάρχουν μοναδικά κλειδιά, πρέπει να γίνει έλεγχος κατά το insert, δηλαδή διάσχιση της λίστας για να μην επιτραπεί εισαγωγή duplicate στοιχείου. Έτσι θα έχουμε γραμμική πολυπλοκότητα, στις περιπτώσεις που είχαμε O(1).

3.

Αρχικά, συγκρίνοντας τις δομές 1a και 2a, παρατηρούμε ότι με τη χρήση του στατικού δισδιάστατου πίνακα (2α), οι πράξεις γίνονται ταχύτερα, ιδίως στις λειτουργίες Delete και Search που έχουμε γραμμική πολυπλοκότητα. Για το Insert, οι χρόνοι θα είναι ελάχιστοι και στις 2 δομές.

Συγκρίνοντας τα 1b και 2b, βλέπουμε ότι οι χρόνοι εκτέλεσης των πράξεων είναι παρόμιοι. Απο τη θεωρία μας γνωρίζουμε ότι οι στατικές δομές συνήθως είναι ταχύτερες, ωστόσο οι δυναμικές δομές παρέχουν σημαντικά οφέλη ως προς την ευελιξία, άρα εν τέλει ενώ περιμέναμε η δομή 2b να είναι η ταχύτερη λόγω του ότι συνδυάζει τα οφέλη του στατικού πίνακα με την ταχύτητα της ταξινόμησης, στην πράξη παρατηρούμε ότι η απόδοση των υλοποιήσεων εξαρτάται από διάφορους παράγοντες.