1η Εργαστηριακή Άσκηση Δομών Δεδομένων
Διονύσιος Θεοδόσης
Α.Μ:321/2015066

Ζήτημα 1ο

Άσκηση 2:

1ος Πίνακας:

Συμπλήρωση Πίνακα σε μη ταξινομημένο πίνακα σειριακής αναζήτησης:

Μέγεθος πίνακα N	Μέσος αριθμός συγκρίσεων	Μέσος χρόνος αναζήτησης
5	3	0.000000
147	104	0.000000
200000000	39211	0.000090
45687456	75688	0.000170
125125	104555	0.000230

Συμπλήρωση Πίνακα σε ταξινομημένο πίνακα σειριακής αναζήτησης:

Μέγεθος πίνακα N	Μέσος αριθμός συγκρίσεων	Μέσος χρόνος αναζήτησης
5	3	0.000400
147	13	0.000550
200000000	173	0.000380
45687456	168	0.000520
125125	169	0.000570

Συμπλήρωση Πίνακα σε ταξινομημένο πίνακα σειριακής αναζήτησης:

Μέγεθος πίνακα N	Μέσος αριθμός συγκρίσεων	Μέσος χρόνος αναζήτησης
5	2	0.000000
147	8	0.000000
200000000	28	0.000000
45687456	27	0.000000
125125	17	0.000000

Βλέποντας κιόλας ότι από την συμπλήρωση του πίνακα, ότι δεν είναι και η καλύτερη μέθοδος να συγκρίνουμε την χρονική πολυπλοκότητα με τον χρόνο εκτέλεσης αφού παίζει ρόλο ο υπολογιστής στον οποίο εκτελείτε και των άλλων παράλληλων διαδικασιών που αυτός εκτελεί στον συγκεκριμένο χρόνο. Αυτό μόνο που φαίνεται καθαρά είναι ότι η δυαδική αναζήτηση είναι πολύ ποιο γρήγορη από την σειριακή όπως και περιμέναμε. Για αυτό τον λόγο όμως υπολογίζουμε την πολυπλοκότητα με την χρήση του μεγάλου Ο.

2ος Πίνακας:

Αλγόριθμος Αναζήτησης	Πολυπλοκότητα Χειρίστης Περίπτωσης
Σειριακή αναζήτηση σε μη ταξινομημένο πίνακα N στοιχείων	O(n)
Σειριακή αναζήτηση σε ταξινομημένο πίνακα N στοιχείων	O(n)
Δυαδική αναζήτηση σε πίνακα N στοιχείων	O(log n)

Έτσι μπορούμε να πούμε ότι η σειριακή αναζήτηση σε ταξινομημένο πίνακα ίσως γίνει πιο γρήγορα από τον μη ταξινομημένο αλλά και πάλι στην χειρότερη περίπτωση θα είναι το ίδιο. Ενώ αυτό που βλέπουμε σίγουρα ότι με την δυαδική σε ταξινομημένο είναι πιο γρήγορη.

Ζήτημα 3ο

Άσκηση 1:

Α)

1η υλοποίηση:

Ανήκει στην κατηγορία πολυπλοκότητας O(n^2). Γιατί η πρώτη for θα είναι για n φορές, όπως επίσης και η δεύτερη for και ας έχει διαφορετική δήλωση οι φορές θα είναι οι ίδιες μετά η if έχει χρόνο πολυπλοκότητας 1 μονάδα όπως και οι άλλες εντολές κ τέλος η printφ και αυτή 1 χρόνο. Άρα n*n*(1*3)+1=4*n^2+1 ,αλλά επειδή στην πολυπλοκότητα δεν μας ενδιαφέρουν οι σταθερές έχουμε ν^2. Επομένως O(n^2).

2η υλοποίηση:

Ανήκει στην κατηγορία πολυπλοκότητας O(n). Γιατί η πρώτη γραμμή κρατάει μία μονάδα χρόνου. Η for θα εκτελεστεί n φορές, η if 1 μονάδα όπως και η εντολή μέσα της και τέλος η printf 1 μονάδα. Άρα 1+n*(1*1)+1=1+n+1=n+2. Αλλά επειδή όπως είπαμε δεν μας ενδιαφέρουν οι σταθερές η πολυπλοκότητα είναι: O(n).

Β)

Η ποιο αποδοτική υλοποίηση είναι η δεύτερη αφού έχει μικρότερη πολυπλοκότητα χρόνου. Αυτό φαίνεται από την ανάλυση στην α ερώτηση.

Άσκηση 2:

Η χρονική πολυπλοκότητα της function 3 θα είναι O(n^3). Διότι πρώτα θα εκτέλεση την function 1 χρονικής πολυπλοκότητας O(n^2) και μετά την function 2 με χρονική πολυπλοκότητα O(n^3). Και αφού είναι η μία μετά από την άλλη, κρατάμε αυτή με τον μεγαλύτερο O.

Γιατί: function1 +function2 =n^2+n^3 και κρατάμε το μεγαλύτερο.