ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ & ΑΡΧΕΙΩΝ

Μιχαηλίδης Στέργιος
2020030080

Α’ Μέρος

Sources:
Διάλεξη *btrees.pptx*
https://www.youtube.com/watch?v=V-ayKf1CWQE

Για την εύρεση της τάξης ενός κόμβου του B+ δέντρου στον δίσκο, λύνω την ανίσωση:

b >= (n-1)*4 [bytes] + n*4 [bytes]

ως προς n.

Όπου:
n → η τάξη του MST για storage outside nodes (n → integer)
b → το μέγεθος σελίδας δίσκου.

Στην παραπάνω υλοποίηση, ο υπολογισμός της τάξης γίνεται εσωτερικά.

Για b = 128 bytes:

n <= (b + 4) / 8 = 136 / 12 = 16,5
Άρα n = 16.

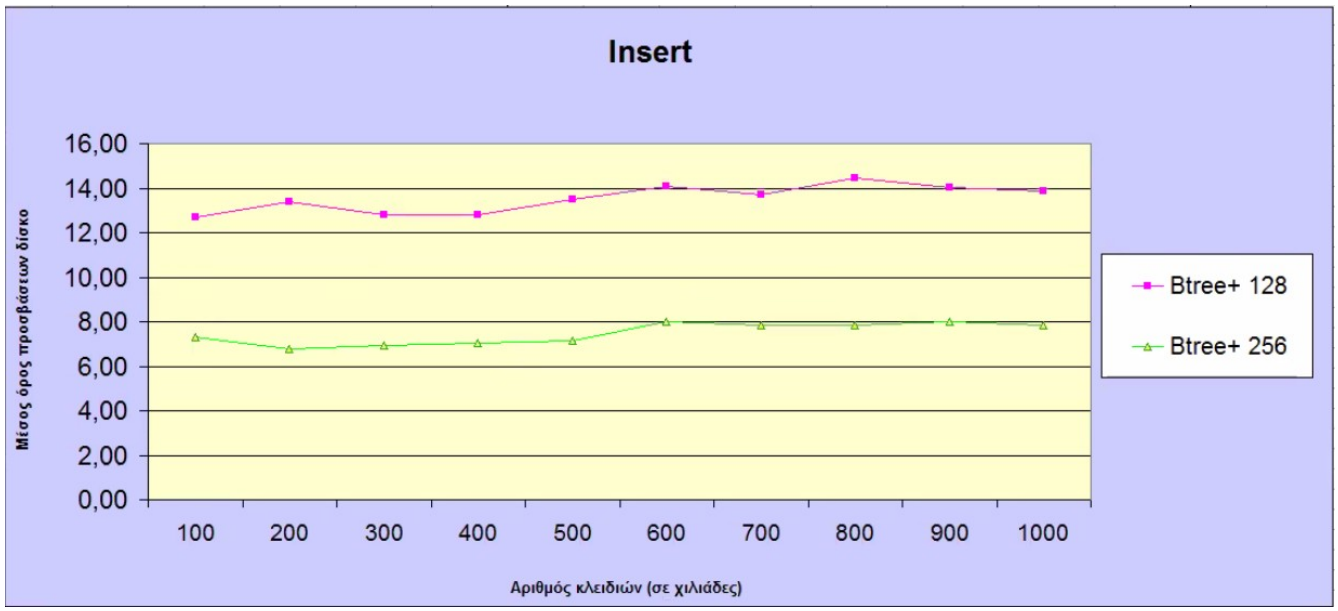
Ομοίως, για b = 256 bytes:

n = 32

Στην περίπτωση του B-tree στον δίσκο η τάξη υπολογίζεται μέσω της ανίσωσης:
b >= (n-1)*4 [bytes] + n*4 [bytes] + (n-1)*sizeof(info)

(ανίσωση τάξης για δέντρο με storage inside nodes.)

Στα παρακάτω διαγράμματα , οι πραγματικές μετρήσεις του αρχείου app_console_output.txt έχουν διαιρεθεί διά 100 έτσι ώστε να απεικονίζουν τον μέσω αριθμό προσβάσεων στον δίσκο ανά ***μία*** μόνο προσθήκη/διαγραφή/αναζήτηση:

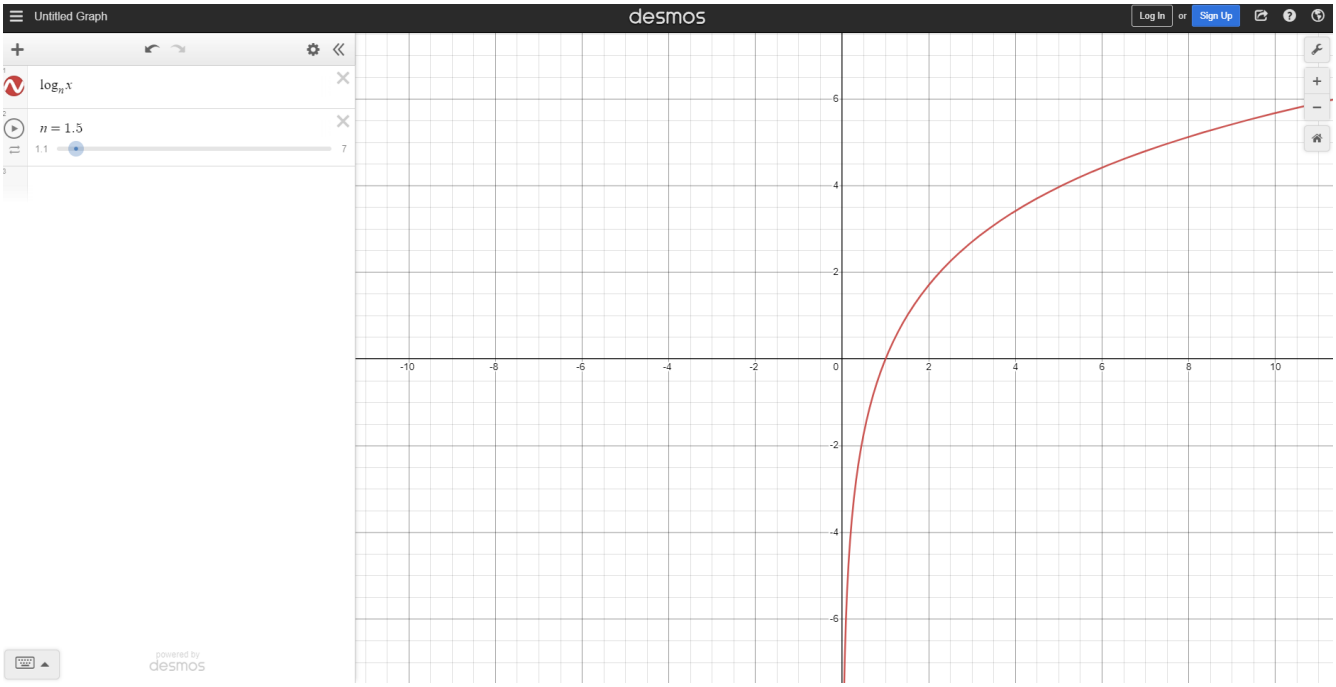


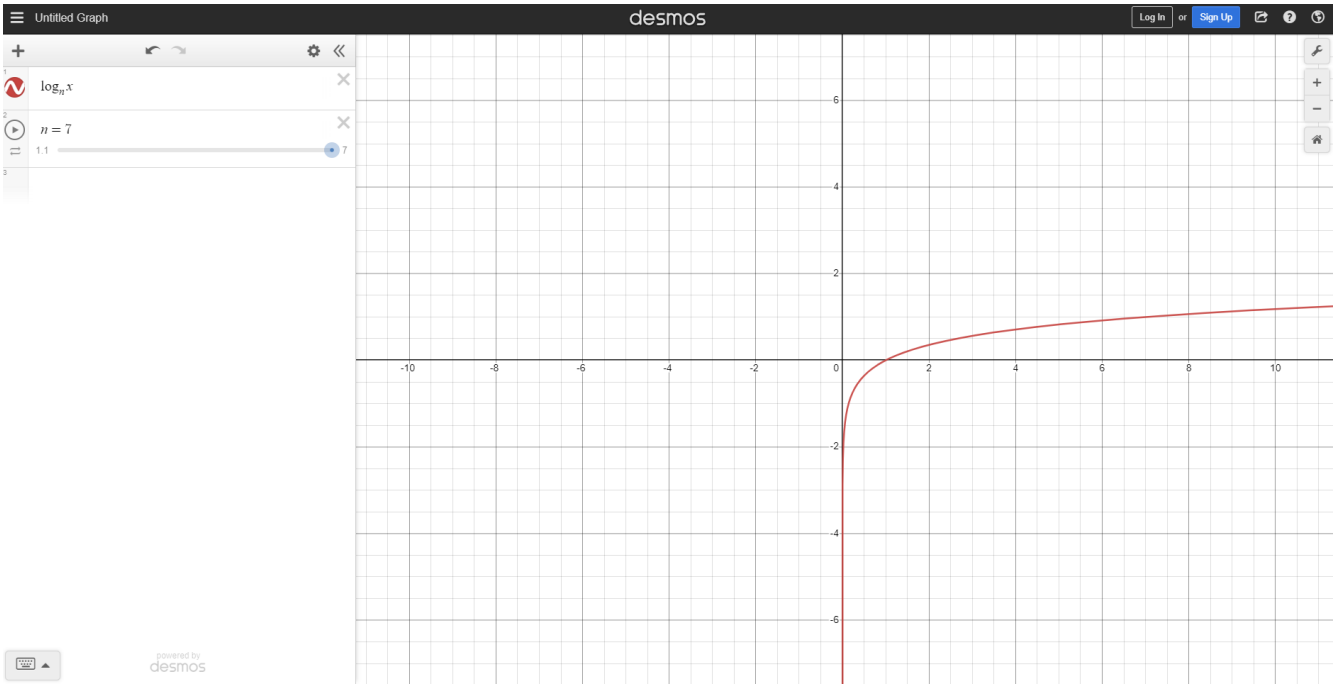
Όπως γνωρίζουμε , η πολυπλοκότητα όλων των μεθόδων του B+ tree είναι O ($\log_n N$) όπου n η τάξη (order) του δέντρου. Αυτό ισχύει τόσο στην συνήθη όσο και στην χειρότερη περίπτωση.

Sources:
https://en.wikipedia.org/wiki/B%2B_tree ,
https://www.cs.helsinki.fi/u/mluukkai/tirak2010/B-tree.pdf,

Παρατηρώ ότι για μεγαλύτερο disc data page b , έχω μικρότερο αριθμό προσβάσεων. Αυτό οφείλεται στην φύση της συνάρτησης $y = \log_n N$. Για μεγαλύτερα n , το πλάτος της y μικραίνει.

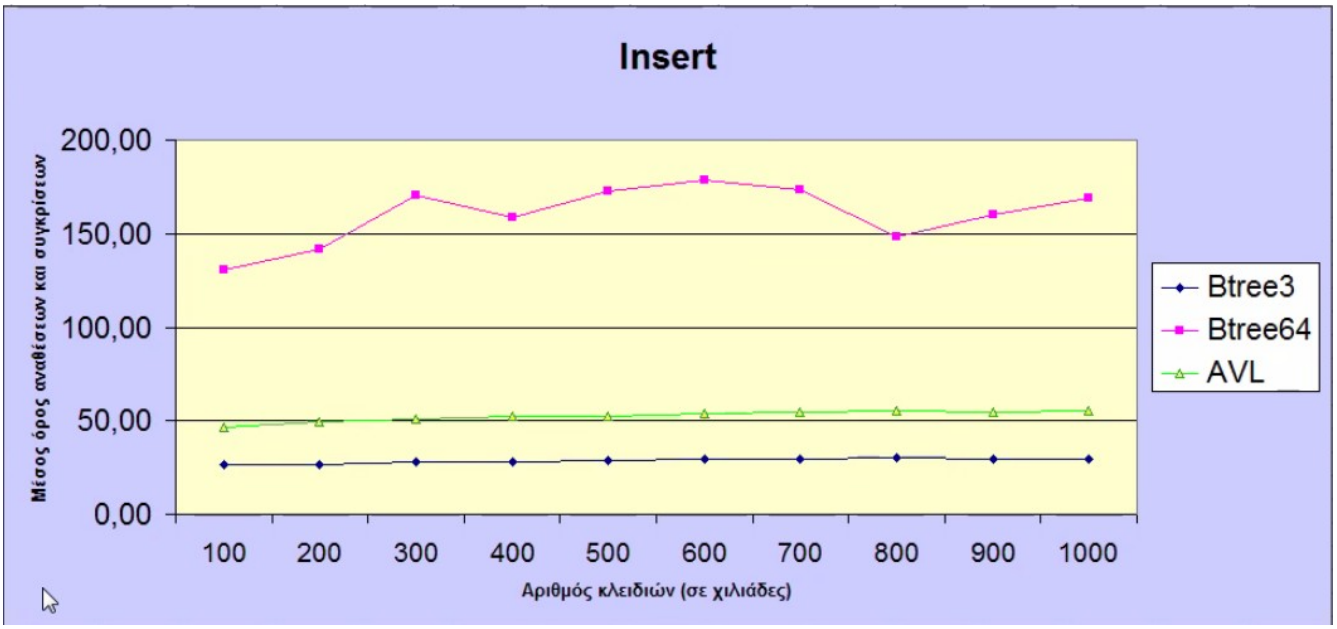
Πρακτικά , όσο μεγαλύτερο το b , μεγαλώνει και η τάξη n. Άρα έχω περισσότερα children per node , επομένως το ολικό ύψος του δέντρου μειώνεται δραστικά.Για αυτόν τον λόγο , μειώνονται και οι ολικές προσβάσεις.

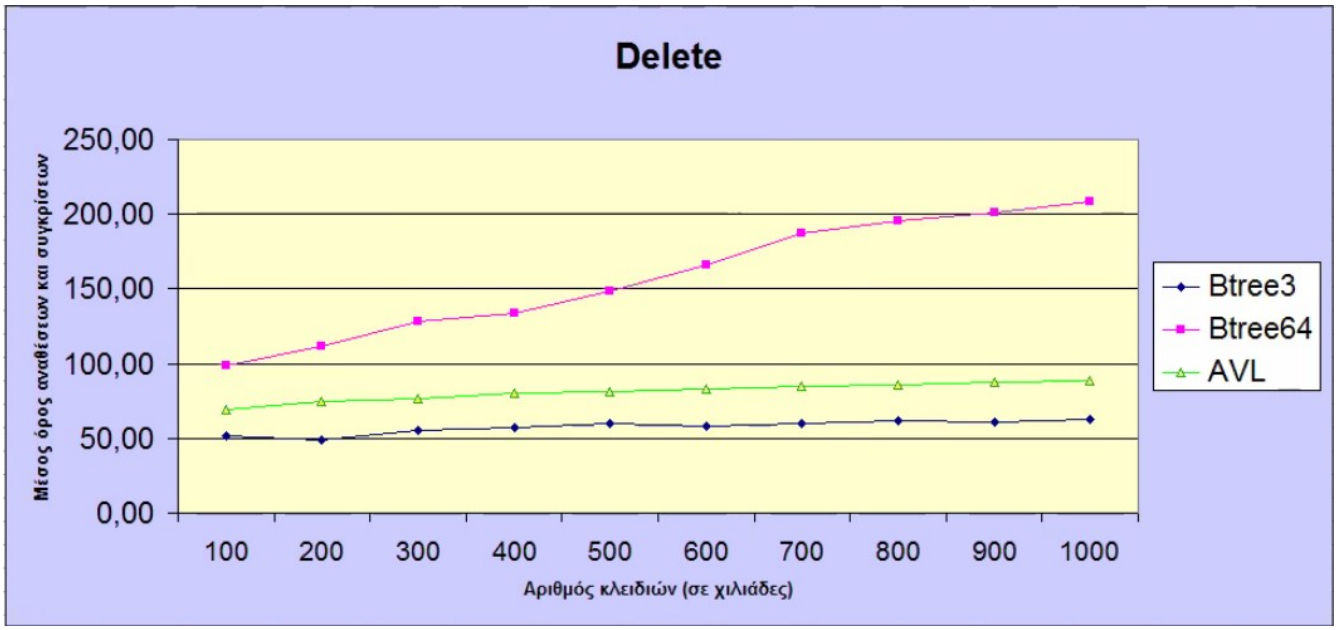

Untitled Graph
desmos
Log In
or
Sign Up
$\log_n x$
$n = 1.5$
1.1
7
powered by
desmos

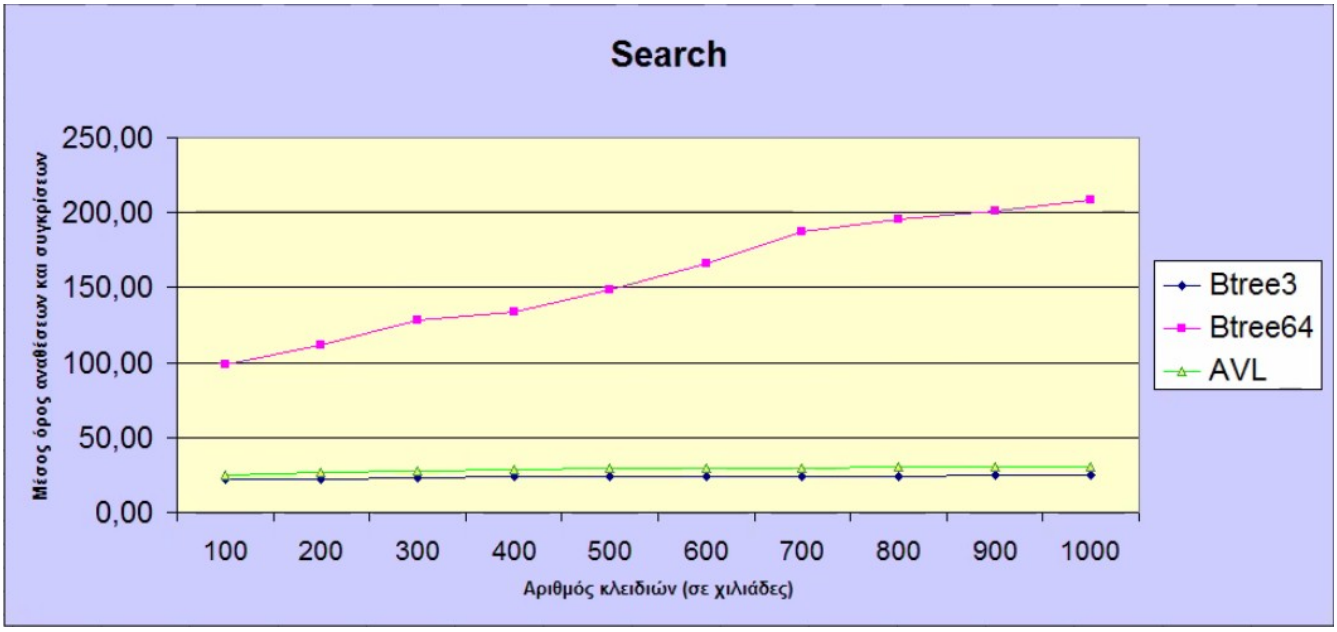

Untitled Graph
desmos
Log In
or
Sign Up
$\log_n x$
$n = 7$
1.1
7
powered by
desmos

Β' Μέρος

Sources:
http://www.csl.mtu.edu/cs2321/www/newLectures/23_IO_Cost_and_B-Trees.html
Βιβλίο *Δομές Δεδομένων & Αλγορίθμων σε JAVA* σελίδες 447 – 454,
https://en.wikipedia.org/wiki/AVL_tree
Βιβλίο *Δομές Δεδομένων & Αλγορίθμων σε JAVA* σελίδες 682 – 686,
https://en.wikipedia.org/wiki/B-tree
Διαλέξεις *btrees.pptx* & *treesmain.pptx*







Όλες οι μέθοδοι έχουν λογαριθμική πολυπλοκότητα.

Τα B trees στην κεντρική μνήμη τα οποία είναι υψηλής τάξης πάσχουν από περισσότερες συγκρίσεις, οι οποίες απαιτούνται τόσο στην εισαγωγή, όσο και στην διαγραφή/αναζήτηση τυχαίου κλειδιού.

Σε αντίθεση με την υλοποίηση του B Tree στον δίσκο όπου το χαμηλότερο ύψος του δέντρου μας ωφελεί (λιγότερα disc accesses) , στην υλοποίηση στην κεντρική μνήμη το μικρότερο ύψος οδηγεί σε πολύ περισσότερες συγκρίσεις εσωτερικά των κόμβων (serial access of linked list).

Στις περιπτώσεις του AVL και B Tree τάξης 3 η απόδοση είναι σχετικά παρόμοια.

Γενική παρατήρηση:

Η σχέση μεταξύ πολυπλοκότητας της search & delete είναι εμφανής , καθώς η delete πρέπει πρώτα να εκτελέσει την search. Σε περίπτωση που βρεθεί το κλειδί και τρέχει η διαδικασία διαγραφής, επιπρόσθετες αναθέσεις και συγκρίσεις λαμβάνουν χώρα έτσι ώστε να εξασφαλισθεί πως η δομή παραμένει balanced (ισχύει για όλες τις παραπάνω).