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

Αναφορά 2ης άσκησης
Ομάδα 10

1)

	BST	AVL	BTree100	BTree600	Linear40	Linear1000
Insert()	O(N)	O(logN)	O(logN)	O(logN)	O(N)	O(N)
Search time()	O(N)	O(logN)	O(logN)	O(logN)	O(N)	O(N)
Search levels()	O(N)	O(logN)	O(logN)	O(logN)	O(N)	O(N)

Binary Search Tree (BST)

Για το BST, η πολυπλοκότητα στη χειρότερη περίπτωση είναι O(N) αφού αν το δέντρο δεν είναι ισορροπημένο θα χρειαστεί διάσχιση όλου του δέντρου με αποτέλεσμα η πολυπλοκότητα να είναι γραμμική. Αυτό ισχύει για τα insert,search times και για τα search levels. Σε αντίθεση με το worst case, η πολυπλοκότητα για το Average case είναι O(logN).

AVL Tree

Για δέντρο AVL, η πολυπλοκότητα χειρότερης περίπτωσης είναι O(logN) αφού λόγω της ισορροπίας του δέντρου, τόσο η εισαγωγή όσο και η αναζήτηση θα απαιτούν το πολύ O(logN) περιστροφές/συγκρισεις. Επίσης το ύψος του δέντρου θα είναι O(logN), οπότε θα έχουμε την ίδια πολυπλοκότητα και για τα search levels. Στο average case, η πολυπλοκότητα θα είναι πάλι O(logN) σε ένα AVL Tree.

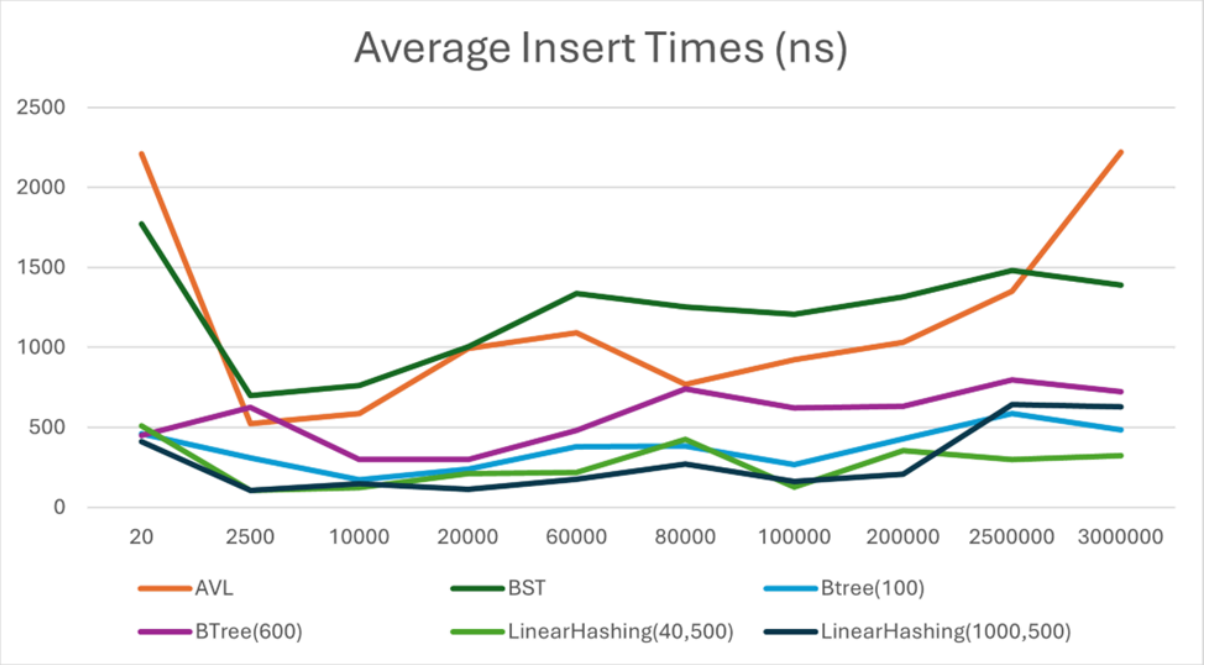
BTree

Το BTree με βάση τις δομικές του ιδιότητες, θα είναι πάντα ισορροπημένο και θα έχει λογαριθμικό ύψος , οπότε η πολυπλοκότητα χειρότερης περίπτωσης θα είναι O(logN) για όλα. Ίδια πολυπλοκότητα έχουμε και αν ενδιαφερόμαστε για το average case.

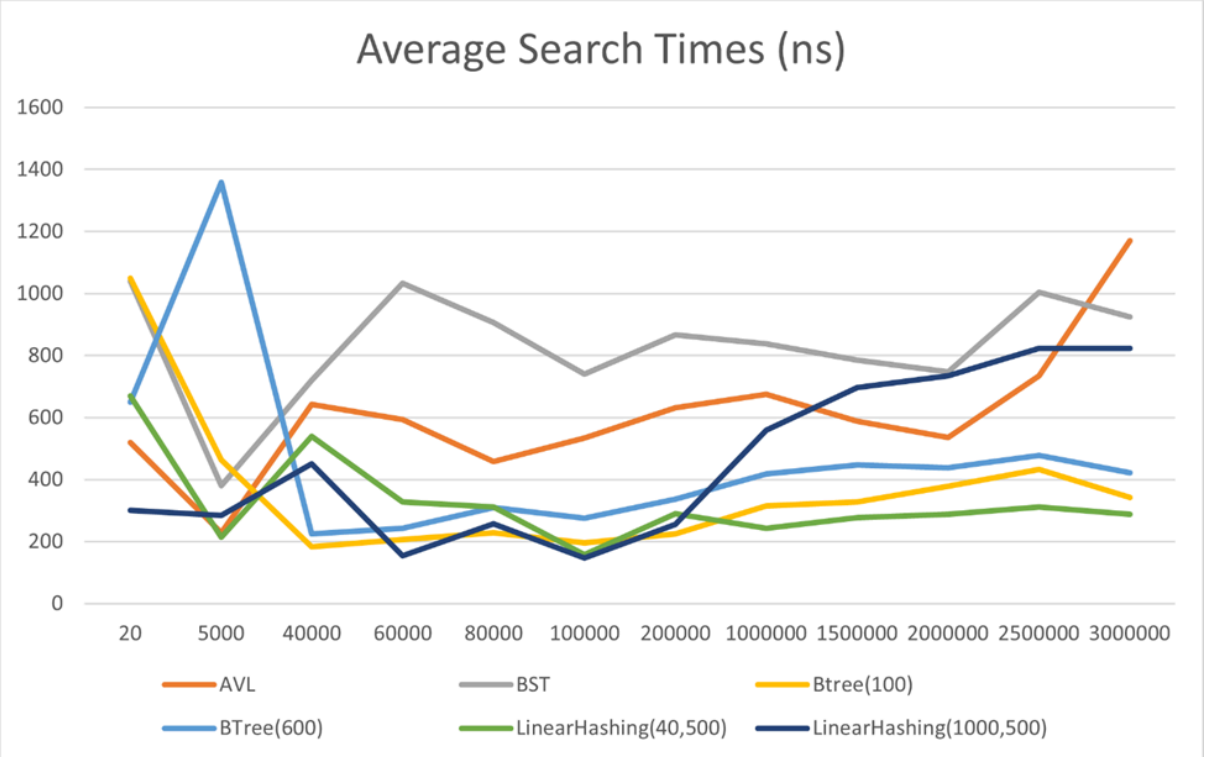
Linear Hashing

Όσον αφορά το Linear Hashing στο average case, η πολυπλοκότητα που συναντάμε είναι O(1). Παρ'ολα αυτά υπάρχει το ενδεχόμενο να έχουμε συγκρούσεις οπότε η πολυπλοκότητα χειρότερης περίπτωσης θα είναι O(N) αφού ίσως χρειαστεί σάρωση/αναδιάταξη/πρόσβαση σε όλα τα στοιχεία.

2) Για το 2ο ερώτημα, υλοποιήσαμε τα διαγράμματα που μας ζητήθηκαν. Χρειάστηκαν ομαλοποίηση κάποιες τιμές - ειδικά για μικρότερα N, εφόσον χρησιμοποιήσαμε ένα γράφημα για όλες μας τις δομές.

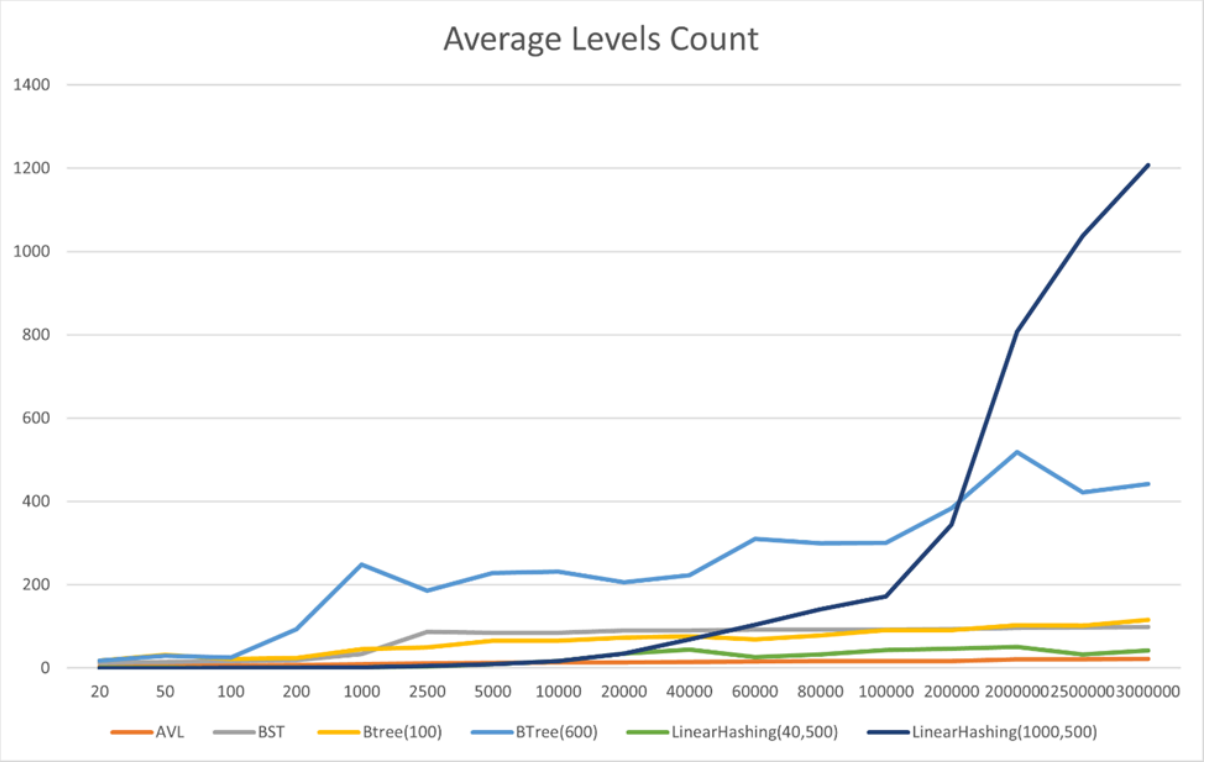


Οι χρόνοι εισαγωγής στο AVL Tree φαίνεται να συμφωνούν με την θεωρητική πολυπλοκότητα O(logN). Για το BST Tree, οι χρόνοι εισαγωγής φαίνονται να ακολουθούν περισσότερο την πολυπλοκότητα του average case του δέντρου O(logN) παρά αυτή της χειρότερης περίπτωσης O(N). Για τα B-Tree οι χρόνοι εισαγωγής ταυτίζονται με τη θεωρητική πολυπλοκότητα O((logN). Για το Linear Hashing, οι πρακτικοί χρόνοι εισαγωγής μπορεί να είναι πολύ μικρότεροι από την θεωρητική πολυπλοκότητα χειρότερης περίπτωσης O(N) για μικρές τιμές του N, αλλά εντοπίζουμε κάποιες αυξήσεις για μεγαλύτερα N, υποδεικνύοντας τις επιπτώσεις των αναδιατάξεων και συγκρούσεων.



Οι χρόνοι αναζήτησης στο AVL Tree φαίνεται να συμφωνούν με την θεωρητική πολυπλοκότητα O(logN). Οι χρόνοι αναζήτησης για το BST Tree και εδώ ταυτίζονται περισσότερο με την πολυπλοκότητα average case του δέντρου, που είναι O(logN). Για το B-Tree, οι μετρήσεις που πήραμε αντικατοπτρίζουν σε καλό βαθμό την θεωρητική πολυπλοκότητα, που είναι O(logN). Οι χρόνοι αναζήτησης στο Linear Hashing (40,500) και Linear Hashing (1000,500) για τις περισσότερες τιμές του N πλησιάζουν τη θεωρητική πολυπλοκότητα μέσης περίπτωσης O(1).

3) Για το 3ο ερώτημα, δημιουργήσαμε γράφημα με τα επίπεδα αναζήτησης για όλες τις υλοποιήσεις. Λόγω αρκετά μεγάλης διαφοράς μεταξύ των αποτελεσμάτων των υλοποιήσεων και επειδή έπρεπε να αναπαρασταθούν στο ίδιο γράφημα, δεν είναι πάντα ξεκάθαρη η πολυπλοκότητα μέσα από τις καμπύλες. Γι αυτό τον λόγο έχουμε παραθέσει τις μετρήσεις που πήραμε, στο τέλος της αναφοράς.



Ο μέσος όρος επιπέδων για το AVL Tree ανταποκρίνονται στη θεωρητική πολυπλοκότητα, αφού η καμπύλη είναι λογαριθμική. Για το BST, ενώ παρατηρούμε ότι οι τιμές αυξάνονται γραμμικά, η αύξηση δεν είναι τόσο μεγάλη, οπότε η πολυπλοκότητα ταιριάζει περισσότερο με το average case της δομής, που είναι O(logN). Για το B-Tree με order 100 η καμπύλη του μέσου όρου επιπέδων πρόσβασης ταιριάζει με τη θεωρητική μας πολυπλοκότητα O(logN). Αντίθετα, το B-Tree με order 600 φαίνεται να αυξάνεται πιο γραμμικά. Στην περίπτωση του Linear Hashing (40,500), η πολυπλοκότητα ανταποκρίνεται στη θεωρητική πολυπλοκότητα του average case O(1), ειδικά για μικρότερα N. Στο Linear Hashing(1000,500), η πολυπλοκότητα, ανταποκρίνεται περισσότερο σε αυτή της χειρότερης περίπτωσης O(N), ειδικά όσο μεγαλώνουν οι τιμές των N.

4) Παρατηρούμε ότι η απόδοση του AVL δέντρου είναι καλύτερη σε σχέση με το BST ειδικά στην περίπτωση που τα δεδομένα είναι μικρότερα και δεν είναι τυχαία. Αυτό οφείλεται στο γεγονός ότι το AVL δέντρο διατηρεί την ισορροπία με αποτέλεσμα οι χρόνοι να μην επηρεάζονται από τη σειρά που εισάγονται τα δεδομένα.

5) Μεταξύ των B-Tree για διαφορετικά orders, παρατηρούμε ότι οι χρόνοι για insert και search είναι παρόμοιοι. Όσον αφορά τα επίπεδα πρόσβασης, για το μεγαλύτερο order έχουμε αρκετά περισσότερα, το οποίο μπορεί να οφείλεται σε μη βέλτιστη χρήση του διαθέσιμου χώρου σε κάθε κόμβο, συχνές διασπάσεις και ανακατατάξεις.

6) Και σε αυτή την περίπτωση παρατηρούμε ότι έχουμε παρόμοιους χρόνους για τα διαφορετικά orders του Linear Hashing, ενώ παρατηρούμε κάποια αύξηση στους χρόνους εισαγωγής για Linear Hashing(100). Ο μέσος όρος επιπέδων για το Linear Hashing (1000,500) είναι υψηλότερος από το Linear Hashing (40,500) για μεγάλες τιμές του N.

7) Αν συγκρίνουμε το BST με βάση τις μετρήσεις μας με τις άλλες δομές, παρατηρούμε ότι δεν υπερτερεί σε τίποτα σε σχέση με τις άλλες. Γενικά, τα πλεονεκτήματα του BST, είναι η απλότητα στην υλοποίηση του, σε σχέση με άλλες δομές, ότι απαιτεί λιγότερο χώρο στη μνήμη και επίσης σε κάποιες εφαρμογές που μπορεί να έχουμε σπάνιες εισαγωγές που δεν επηρεάζουν σημαντικά την ισορροπία του δέντρου, μπορεί να είναι αποδοτικό.

8) Η κατάλληλη δομή θα ήταν η BTree με order 600 αφού έτσι μπορούμε να έχουμε μεγάλο αριθμό κλειδιών σε κάθε πρόσβαση που κάνουμε στον δίσκο μειώνοντας τον συνολικό αριθμό των προσβάσεων που χρειάζονται. Τέλος η ισορροπία της δομής BTree μας εξασφαλίζει ότι για την αναζήτηση ενός κλειδιού απαιτείται λογαριθμική πολυπλοκότητα. Πρακτικά όμως είδαμε ότι τα επίπεδα πρόσβασης για το BTree με order 100 ήταν λιγότερα, οπότε με βάση τις μετρήσεις μας, αυτή θα ήταν η επιλογή μας.

9) Για τη RAM θα προτιμούσαμε τις δομές AVL Tree και BST Tree και ειδικά την δομή AVL Tree αφού η ισορροπία που διατηρείται στο δέντρο την καθιστά ιδανική δομή. Όσον αφορά το δίσκο η δομή που θα επιλέγαμε θα ήταν η BTree(100), καθώς προσφέρει καλό συμβιβασμό μεταξύ του αριθμού των κλειδιών ανά κόμβο και του αριθμού των επιπέδων προσπέλασης, διατηρώντας υψηλή αποδοτικότητα στις αναζητήσεις.

Μετρήσεις

Παρακάτω ακολουθούν οι μετρήσεις που πήραμε.

Starting tests...

Generating
20...50...100...200...1000...2500...5000...10000...20000...40000...60000...80000...100000...200000...1000000...1500000...2000000...2500000...3000000...
Testing
20...50...100...200...1000...2500...5000...10000...20000...40000...60000...80000...100000...200000...1000000...1500000...2000000...2500000...3000000...
Average Insert Times (ns)

N\Structure	AVLTree	BSTree	BTree (100)	BTree (600)	LinearHashing (40,500)	LinearHashing(1000,500)
N=20	2210.00	1770.00	460.00	450.00	510.00	410.00
N=50	2420.00	350.00	750.00	780.00	380.00	360.00
N=100	840.00	410.00	730.00	1200.00	730.00	310.00
N=200	3330.00	6530.00	660.00	490.00	300.00	290.00
N=1000	1496.00	374.00	400.00	932.00	422.00	120.00
N=2500	522.00	697.00	310.00	625.00	107.00	104.00
N=5000	579.00	663.00	1280.00	4590.00	124.00	120.00
N=10000	586.00	760.00	174.00	298.00	122.00	146.00
N=20000	992.00	1002.00	239.00	298.00	211.00	111.00
N=40000	966.00	1671.00	409.00	397.00	224.00	163.00
N=60000	1092.00	1337.00	380.00	481.00	217.00	175.00
N=80000	770.00	1251.00	382.00	740.00	424.00	269.00
N=100000	922.00	1208.00	268.00	622.00	127.00	163.00
N=200000	1032.00	1316.00	428.00	633.00	355.00	206.00
N=1000000	956.00	1254.00	435.00	724.00	44037.00	474.00
N=1500000	1006.00	1318.00	446.00	711.00	320.00	769.00
N=2000000	946.00	1303.00	446.00	747.00	90578.00	606.00
N=2500000	1350.00	1480.00	586.00	796.00	297.00	643.00
N=3000000	2221.00	1388.00	483.00	724.00	323.00	627.00

Average Search Times (ns)

N\Structure	AVLTree	BSTree	BTree (100)	BTree (600)	LinearHashing (40,500)	LinearHashing(1000,500)
N=20	520.00	1040.00	1050.00	650.00	670.00	300.00
N=50	170.00	250.00	1050.00	900.00	260.00	250.00
N=100	170.00	260.00	1000.00	740.00	250.00	240.00
N=200	180.00	300.00	890.00	2560.00	320.00	450.00
N=1000	222.00	198.00	1822.00	2220.00	296.00	302.00
N=2500	350.00	311.00	419.00	1079.00	277.00	263.00
N=5000	228.00	380.00	464.00	1358.00	213.00	284.00
N=10000	283.00	413.00	501.00	1923.00	228.00	145.00
N=20000	336.00	458.00	215.00	150.00	357.00	243.00
N=40000	643.00	721.00	184.00	224.00	539.00	450.00
N=60000	593.00	1033.00	206.00	243.00	327.00	154.00
N=80000	458.00	906.00	229.00	309.00	312.00	257.00
N=100000	534.00	740.00	195.00	276.00	158.00	147.00
N=200000	631.00	866.00	224.00	337.00	290.00	255.00
N=1000000	675.00	838.00	316.00	419.00	243.00	560.00
N=1500000	588.00	786.00	327.00	448.00	278.00	697.00
N=2000000	536.00	748.00	378.00	438.00	288.00	735.00
N=2500000	734.00	1004.00	432.00	478.00	311.00	823.00
N=3000000	1170.00	925.00	342.00	422.00	288.00	824.00

Average Levels Count

N\Structure	AVLTree	BSTree	BTree (100)	BTree (600)	LinearHashing (40,500)	LinearHashing(1000,500)
N=20	4.40	12.10	18.20	17.90	0.40	0.10
N=50	5.60	14.70	31.60	29.30	0.50	0.20
N=100	6.00	17.00	22.20	24.80	0.20	0.30
N=200	7.60	18.40	24.50	92.80	0.40	0.50
N=1000	9.76	33.68	45.78	248.86	1.80	1.82
N=2500	10.95	86.38	49.89	186.29	4.19	4.65
N=5000	12.04	84.71	66.10	228.35	8.77	9.21
N=10000	13.59	85.06	65.16	231.31	17.05	16.24
N=20000	14.00	89.97	73.37	205.93	34.92	35.08
N=40000	14.94	89.91	76.60	223.12	44.34	68.50
N=60000	15.82	91.80	69.23	310.18	26.52	104.13
N=80000	16.26	91.90	78.72	299.07	32.91	140.65
N=100000	16.73	92.40	90.77	300.71	43.07	172.32
N=200000	16.99	93.48	90.66	383.37	46.16	344.41
N=1000000	19.78	97.71	95.98	511.68	52.85	800.60
N=1500000	20.68	97.77	122.41	475.84	41.65	1128.09
N=2000000	20.62	96.71	103.37	518.89	51.04	808.16
N=2500000	20.99	97.70	102.25	422.16	33.18	1037.43
N=3000000	21.64	98.72	115.49	441.52	42.60	1207.03