HW 231 Trie Assignement

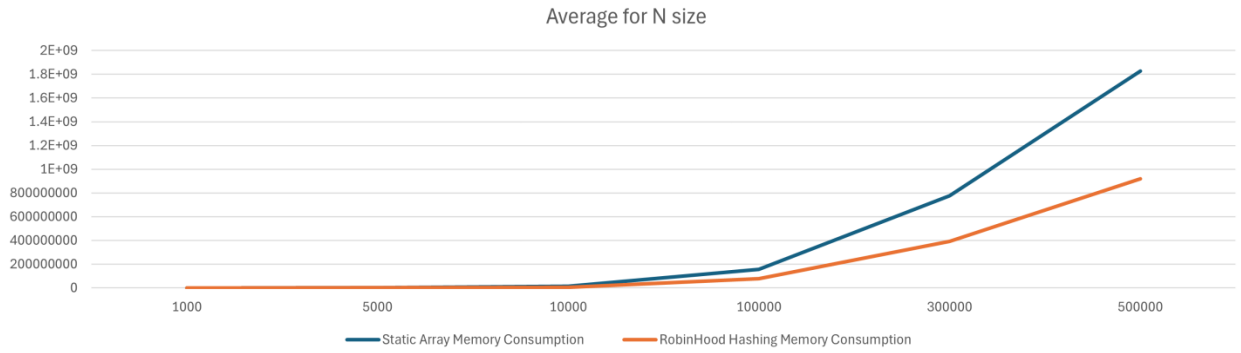
Συγγραφείς : Μαρίνος Αντωνίου και Κωνσταντίνος Θεοφίλου

Ομάδα : 43

Πρώτού ξεκινήσουμε με την παρουσίαση των γραφικών παραστάσεων για την κατανόηση της χρήσης μνήμης , θεωρούμε σημαντικό να εξηγήσουμε πώς έχουν παρθεί οι κάθε μετρήσεις για κάθε κλάση.

Η μνήμη με την χρήση στατικού πίνακα υπολογίζεται προσθέτοντας 104 bytes για τον πίνακα παιδιών (26 pointers), 12 bytes για τις μεταβλητές του κόμβου (wordLength, importance, metadata), και την αναδρομική μνήμη των παιδιών του κόμβου. Προστίθενται +4 bytes ως meta data για δίκαιη σύγκριση με το Robin Hood hashing , εφόσων υπολογίζουμε σε O(1) την θέση που μπαίνει το στοιχείο στον στατικό πίνακα , πράγμα που δεν συμβαίνει στην υλοποίηση με το Robin Hood Hashing.

Η μνήμη με την χρήση του πίνακα Robin Hood υπολογίζεται αθροίζοντας τις μνήμες που καταλαμβάνουν τα αντικείμενα TrieNode, οι αντίστοιχοι πίνακες κατακερματισμού RobinHoodHashing και τα μεμονωμένα στοιχεία αυτών των πινάκων. Κάθε TrieNode χρησιμοποιεί 4 bytes για το πεδίο wordLength και υπολογίζει αναδρομικά τη μνήμη για τα παιδιά του, τα οποία αποθηκεύονται σε αντικείμενα RobinHoodHashing. Η δομή RobinHoodHashing περιλαμβάνει 12 bytes για δεδομένα όπως η χωρητικότητα και το μέγεθος, μνήμη για τον πίνακα κατακερματισμού (capacity * 16 bytes για κάθε θέση, ακόμα και αν είναι κενή) και επιπλέον μνήμη για τα μη κενά αντικείμενα Element, το καθένα από τα οποία καταλαμβάνει 14 bytes (για το χαρακτήρα-κλειδί, το μήκος ανίχνευσης και την αναφορά σε παιδί TrieNode). Το πρόγραμμα επαναλαμβάνει αναδρομικά τον υπολογισμό σε όλα τα TrieNode της δομής, υπολογίζοντας τη μνήμη για κάθε κόμβο, τον πίνακα κατακερματισμού του και τα παιδιά του, εξασφαλίζοντας μια ακριβή εκτίμηση της συνολικής μνήμης που χρησιμοποιείται από την trie.



SIZE N TESTS	Test 1		Test 2	
Word Count (and size N)	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio
1000 (N = 3)	335940	177524	338140	178976
5000 (N = 5)	3307040	1709954	3297800	1703300
10000 (N = 8)	12787280	6520150	12807740	6534138
100000 (N = 10)	155694220	78820278	155672000	78806278
300000 (N = 15)	777536320	392519164	777586700	392550712
500000 (N = 20)	1828519660	919383068	1828465760	919329658

Test 3		Test 4	
Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio
334180	176690	335060	176956
3307920	1711780	3297360	1704026
12797180	6527276	12811920	6534966
155657920	78801702	155633940	78789914
777588240	392546706	777557880	392528884
1828424180	919323052	1828424620	919323558

Test5		Test Averages		Ratio
Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumption	RobinHood Hashing Memory Consumption	Static / Robinhood
333740	176584	335412	177346	1.891285961
3301760	1706386	3302376	1707089.2	1.934506996
12815660	6536680	12803956	6530642	1.96059683
155663860	78808074	155664388	78805249.2	1.975304813
777554140	392532252	777564656	392535543.6	1.980877066
1828494360	919366178	1828465716	919345102.8	1.988878508

Η γραφική παράσταση συγκρίνει τη μνήμη που καταναλώνεται από το Static Array (μπλε γραμμή) και το Robin Hood Hashing (πορτοκαλί γραμμή) καθώς αυξάνεται ο αριθμός των λέξεων και το μέγεθος . Παρατηρούμε ότι το Static Array καταναλώνει σημαντικά περισσότερη μνήμη, με τη διαφορά να αυξάνεται εκθετικά καθώς μεγαλώνει το N. Οι πίνακες παρουσιάζουν τα αποτελέσματα από πειράματα (Tests 1-5) για διαφορετικά N και τον μέσο όρο κατανάλωσης μνήμης, όπου το Static Array είναι σταθερά σχεδόν διπλάσιας κατανάλωσης σε σχέση με το

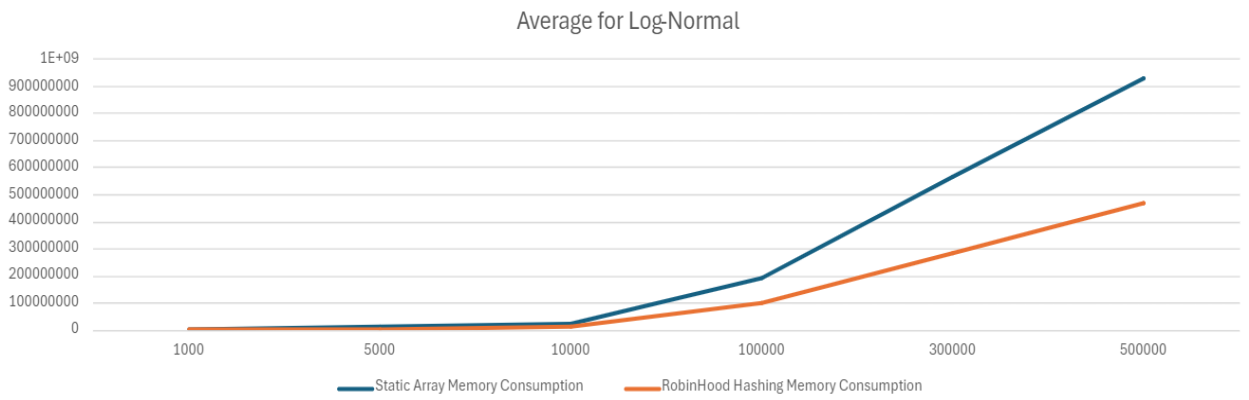
Robin Hood Hashing. Το πηλίκο "Static / Robin Hood" κυμαίνεται από 1.89 έως 1.99, δείχνοντας την υψηλή αποδοτικότητα του Robin Hood Hashing. Η πορτοκαλί γραμμή παραμένει πολύ πιο χαμηλή, αποδεικνύοντας ότι το Robin Hood Hashing διαχειρίζεται τη μνήμη πολύ πιο αποδοτικά, ιδιαίτερα για μεγάλα σύνολα δεδομένων. Συμπερασματικά, το Robin Hood Hashing είναι η βέλτιστη επιλογή για εφαρμογές που απαιτούν αποτελεσματική χρήση της μνήμης.

Αναφορικά με το πείραμα που πρέπει να χρησιμοποιήσουμε υπάρχουσα κατανομή :

Στο πείραμα χρησιμοποιήθηκε η log-normal κατανομή για την παραγωγή λέξεων με μήκη που αντικατοπτρίζουν ρεαλιστικές γλωσσικές δομές. Σύμφωνα με την έρευνα "Efficient Generation of Random Sequences" (https://arxiv.org/pdf/1709.01712), ιδιαίτερα στις σελίδες 4-6, η log-normal κατανομή παράγει κυρίως μικρές τιμές με ορισμένες μεγάλες αποκλίσεις, γεγονός που την καθιστά κατάλληλη για μήκη λέξεων. Οι παράμετροι της κατανομής καθορίστηκαν με βάση τη μελέτη "On the Log-Normality of Natural Language Phenomena" (https://arxiv.org/pdf/1207.2334), στις σελίδες 2-3, όπου η χρήση του μ (μέση τιμή του log) και σ (διακύμανση) αναλύεται για δεδομένα γλώσσας. Οι τιμές μ_min=1.5, μ_max=3.0, σ_min=0.3, σ_max=0.7 επιλέχθηκαν τυχαία σε κάθε δοκιμή, εξασφαλίζοντας μέσο μήκος λέξεων 3-30 χαρακτήρες με ρεαλιστική διακύμανση, έτσι ώστε να εμφανίζονται κατανεμημένα σε λογικά μεγέθη οι λέξεις, με τις περισσότερες που παράγονται να είναι μεταξύ 8-12 χαρακτήρων.

Word	Mu	Sigma	Word Length	Log-Normal PDF
jgyipgtkzpw	2.354225663	0.362769053	11	0.001678284
lfjtceepsdqisnaramto	2.875152227	0.228877928	20	1.96277E-06
ahsqnfzlbiejhpaodchtf	2.823535332	0.221038941	21	1.001E-06
plslqqgntwq	2.786265308	0.270714224	11	0.001678284
rkhkkidz	2.653655912	0.542517977	8	0.01972846
kqozegqnjgpuiwgznow	2.634653565	0.3256906	19	3.90532E-06
klni	1.817673646	0.524134004	4	0.277516435
qnexr	2.141028292	0.550189535	5	0.177965855
voi	1.034326478	0.335881945	3	0.278276863
zsqfdz	1.749386223	0.231182607	6	0.09292761
ptjjdxh	1.638955371	0.590227105	7	0.043870831
huxz	1.114562375	0.260933408	4	0.277516435
tkslvp	1.486525467	0.475937017	6	0.09292761
eswzgfd	2.423256196	0.55627658	7	0.043870831
zczwznjwjc	2.442865067	0.427410148	10	0.00381069
yshmphpghlyvrpxsybkifhshlxgkma	2.528710568	0.581917832	30	4.15914E-09
snc	1.10930428	0.240067362	3	0.278276863
obi	1.110997092	0.434907706	3	0.278276863
wesgiyheztgb	1.916125754	0.415507126	12	0.000746468
bja	1.461884626	0.40471323	3	0.278276863
tvigljccgxfsuftizy	3.044655563	0.243919043	18	7.88871E-06
turkmewyhczdh	2.599950223	0.257233991	13	0.000336365
loftrbgzrhsgwlaszwvnpjvnqzxbob	3.319253995	0.335305717	30	4.15914E-09
cmmeicbxumweaxsqw	2.786042777	0.251047223	17	1.61846E-05
hvtknkimpj	1.256499535	0.553960038	10	0.00381069
lqwycdelse	2.123340216	0.578920687	10	0.00381069
jwynljv	2.001263806	0.359815272	7	0.043870831
tkwhsqjlmvhhpaturkrpfkfjuqye	2.376748655	0.443206301	28	1.29745E-08
gvm	1.372446473	0.360398823	3	0.278276863
hwdxofetyfaowv	2.436021796	0.341800824	14	0.000153827
pko	1.412167866	0.295439884	3	0.278276863
rcz	1.084217945	0.2041515	3	0.278276863
aralixr	1.279190367	0.514632309	7	0.043870831

Πρίν να εμφανίσουμε την γραφική , θεωρήσαμε σημαντικό να παρουσιάσουμε ένα απόσπασμα του δείγματος και να το σχολιάσουμε .Το δείγμα που δημιουργήθηκε από τη γεννήτρια τυχαίων λέξεων με βάση τη λογαριθμοκανονική κατανομή (log-normal) δείχνει μια αναμενόμενη κατανομή των μηκών λέξεων, με τις τιμές να κυμαίνονται από 4 έως 30 χαρακτήρες. Οι παράμετροι μ και σ καθορίζουν τη μέση τιμή και τη διασπορά του λογαρίθμου των λέξεων, με μεγαλύτερες τιμές να αντιστοιχούν σε μεγαλύτερα μήκη λέξεων. Οι πιθανότητες εμφάνισης, όπως φαίνεται από το δείγμα αυτό, είναι υψηλότερες για λέξεις μεσαίου μήκους (7–15 χαρακτήρες), ενώ μειώνονται για πολύ μικρές ή πολύ μεγάλες λέξεις, κάτι που είναι χαρακτηριστικό της λογαριθμοκανονικής κατανομής. Συνολικά, το δείγμα επιβεβαιώνει τη σύνδεση των στατιστικών παραμέτρων με την κατανομή των λέξεων, διατηρώντας μια λογική ισορροπία μεταξύ μήκους και πιθανότητας εμφάνισης. Ο μέσος όρος μήκους των λέξεων σε αυτό το δείγμα είναι περίπου 11.97 χαρακτήρες



LOG-NORMAL DISTRIBUTION TESTS	Test 1		Test 2	
Word Count	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio
1000	2304940	1162632	2382160	1200890
5000	10939720	5525892	10555380	5334060
10000	20576820	10416864	21059940	10659584
100000	194520920	98024786	194685700	98105726
300000	565252820	285965820	565051960	285860350
500000	930173860	469699456	929567760	469385888

Test 3		Test 4	
Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio
2196480	1107666	2212760	1116030
10584640	5347154	10805080	5458974
20940040	10597760	21194360	10726954
193736840	97631590	194890960	98195108
568299380	287496520	565555320	286109916
930812520	470023520	932842680	471025892

Test5		Test Averages		Ratio
Static Array Memory Consumptio	RobinHood Hashing Memory Consumptio	Static Array Memory Consumption	RobinHood Hashing Memory Consumption	Static / Robinhood
2236080	1128010	2266484	1143046	1.982846529
10646240	5378690	10706212	5408954	1.979349797
21072040	10663974	20968640	10613027	1.975745431
194810000	98162894	194528884	98024021	1.984502191
567132280	286915840	566258352	286469689	1.976678069
930068920	469635602	930693148	469954072	1.980391711

Η γραφική παράσταση συγκρίνει τη μνήμη που καταναλώνουν οι δομές Static Array (μπλε γραμμή) και Robin Hood Hashing (πορτοκαλί γραμμή) κατά την εφαρμογή της log-normal κατανομής για τη δημιουργία δεδομένων. Τα αποτελέσματα δείχνουν ότι το Static Array καταναλώνει σταθερά περισσότερη μνήμη από το Robin Hood Hashing, με τη διαφορά να αυξάνεται όσο μεγαλώνει ο αριθμός των λέξεων. Οι πίνακες αποκαλύπτουν ότι το πηλίκο "Static / Robin Hood" κυμαίνεται μεταξύ 1.97 και 1.98, αποδεικνύοντας ότι το Robin Hood Hashing είναι σχεδόν διπλάσια πιο αποδοτικό. Το Static Array εμφανίζει υψηλή κατανάλωση μνήμης, ιδιαίτερα για μεγαλύτερα σύνολα δεδομένων (500.000 λέξεις), ενώ το

Robin Hood Hashing κλιμακώνεται πιο ομαλά. Τα δεδομένα επιβεβαιώνουν τη σημαντική εξοικονόμηση μνήμης που προσφέρει το Robin Hood Hashing, καθιστώντας το ιδανική επιλογή για μεγάλης κλίμακας εφαρμογές αποθήκευσης δεδομένων. Συνολικά, το πείραμα υπογραμμίζει την αποδοτικότητα της δομής Robin Hood Hashing για συστήματα με αυστηρούς περιορισμούς μνήμης.