Same Length 3:

In this section we generated dictionaries of size n = 1k, 5k, 10k, 100k, 250k and 500k.

Word size = 3.

We run each dictionary using the TrieNode with static array and the TrieNode with the RobinHood Hashing table.

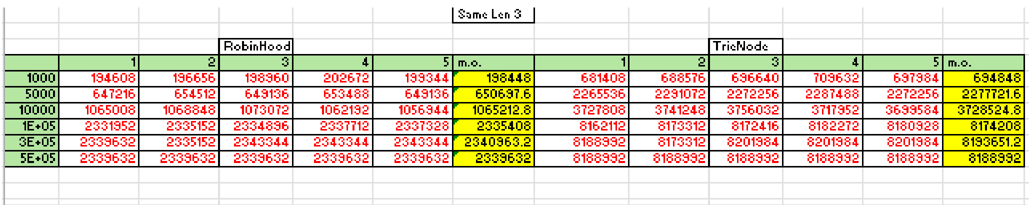
We generated 5 dictionaries for each n and calculated the average of each n for each TrieNode.

The Graph below shows two lines, the orange being the static and the blue being the RobinHood.

The Static table creates a 26 static array length of children while the RobinHood hashing creates a table of 5, then if 90% of the array is used it rehashes to 11 then to 19 and then to 29.

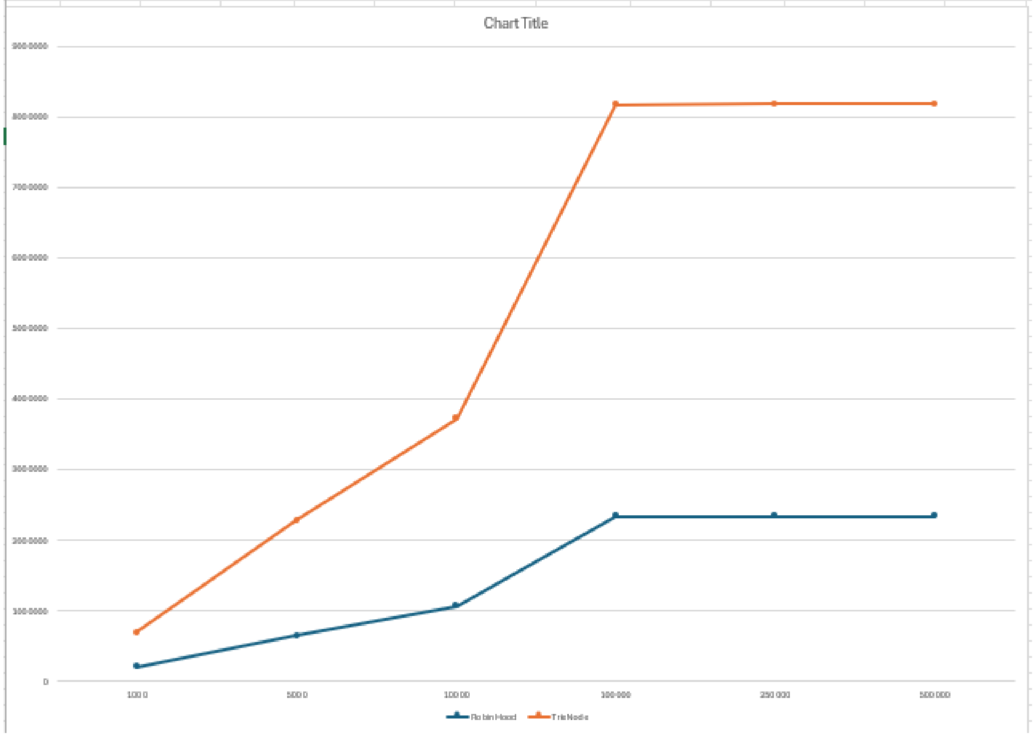
As shown below, while the dictionary length gets bigger, the static table utilizes more useless space as it always generates 26 space tables and the memory utilized is being increased exponentially.

Also after 17,576 words every possible word has being inserted so every other word is a duplicate thus no more space is being utilized.



Same Len 3

			RobinHood						TrieNode			
	1	2	3	4	5	m.o.	1	2	3	4	5	m.o.
1000	194608	196656	198960	202672	199344	198448	681408	688576	696640	709632	697984	694848
5000	647216	654512	649136	653488	649136	650697.6	2265536	2291072	2272256	2287488	2272256	2277721.6
10000	1065008	1068848	1073072	1062192	1056944	1065212.8	3727808	3741248	3756032	3717952	3699584	3728524.8
1E+05	2331952	2335152	2334896	2337712	2337328	2335408	8162112	8173312	8172416	8182272	8180928	8174208
3E+05	2339632	2335152	2343344	2343344	2343344	2340963.2	8188992	8173312	8201984	8201984	8201984	8193651.2
5E+05	2339632	2339632	2339632	2339632	2339632	2339632	8188992	8188992	8188992	8188992	8188992	8188992



Same Length 5:

In this section we generated dictionaries of size n = 1k, 5k, 10k, 100k, 250k and 500k.

Word size = 5.

We run each dictionary using the TrieNode with static array and the TrieNode with the RobinHood Hashing table.

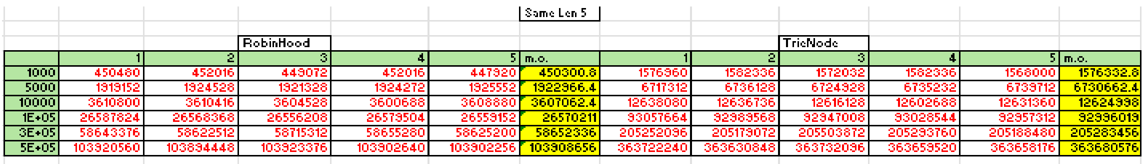
We generated 5 dictionaries for each n and calculated the average of each n for each TrieNode.

The Graph below shows two lines, the blue being the static and the green being the RobinHood.

The Static table creates a 26 static array of children while the RobinHood hashing creates a table of 5, then if 90% of the array is used it rehashes to 11 then to 19 and then to 29.

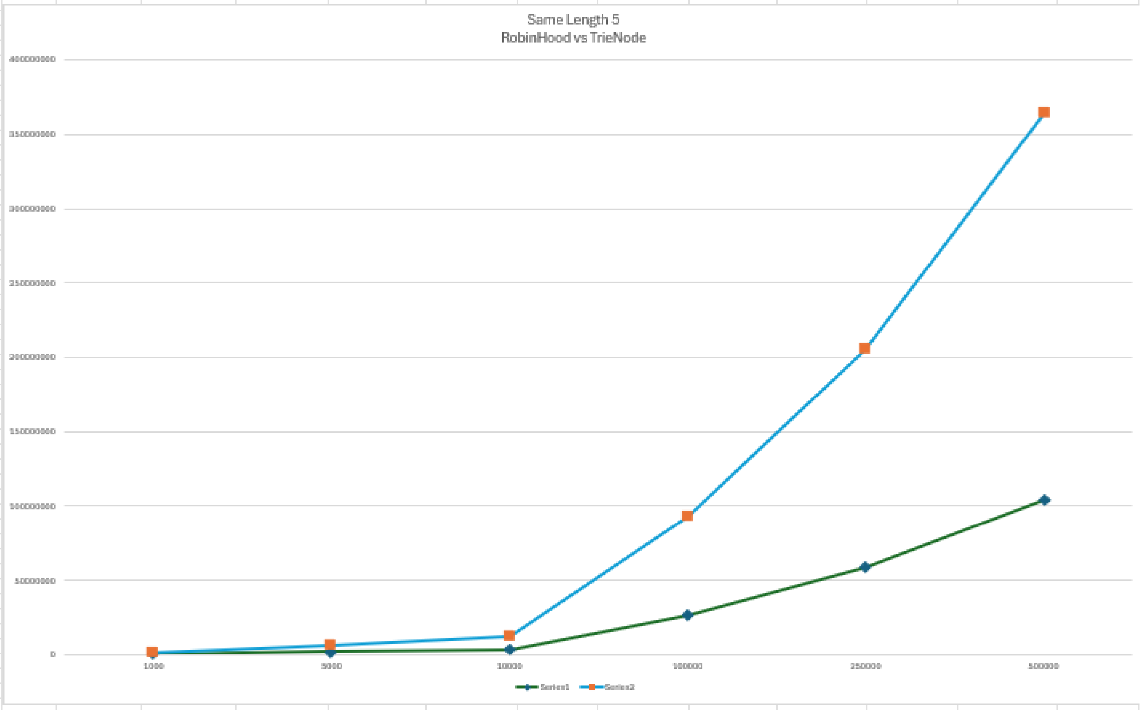
s shown below, while the dictionary length gets bigger, the static table utilizes more useless space as it always generates 26 space tables and the memory utilized is being increased exponentially.

The RobinHood Hashing method is more efficient memory-wise due to its rehashing techniques.



Same Len 5

			RobinHood						TrieNode			
	1	2	3	4	5	m.o.	1	2	3	4	5	m.o.
1000	450480	452016	449072	452016	447920	450300.8	1576960	1582336	1572032	1582336	1568000	1576332.8
5000	1919152	1924528	1921328	1924272	1925552	1922966.4	6717312	6736128	6724928	6735232	6739712	6730662.4
10000	3610800	3610416	3604528	3600688	3608880	3607062.4	12638080	12636736	12616128	12602688	12631360	12624998
1E+05	26587824	26568368	26556208	26579504	26559152	26570211	93057664	92989568	92947008	93028544	92957312	92996019
3E+05	58643376	58622512	58715312	58655280	58625200	58652336	205252096	205179072	205503872	205293760	205188480	205283456
5E+05	103920560	103894448	103923376	103902640	103902256	103908656	363722240	363630848	363732096	363659520	363658176	363680576



Same Length 7:

In this section we generated dictionaries of size n = 1k, 5k, 10k, 100k, 250k and 500k.

Word size = 7.

We run each dictionary using the TrieNode with static array and the TrieNode with the RobinHood Hashing table.

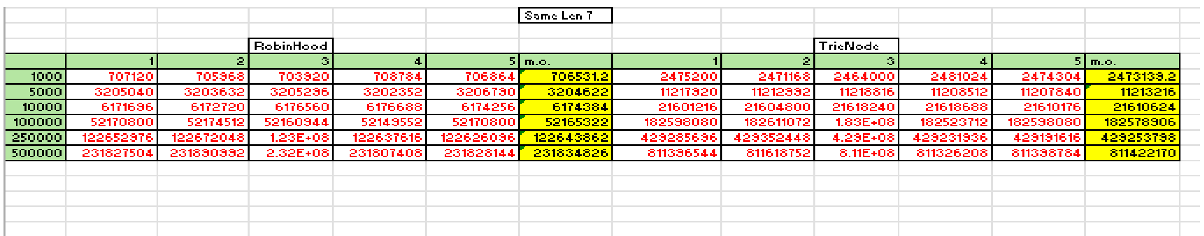
We generated 5 dictionaries for each n and calculated the average of each n for each TrieNode.

The Graph below shows two lines, the green being the static and the orange being the RobinHood.

The Static table creates a 26 static array of children while the RobinHood hashing creates a table of 5, then if 90% of the array is used it rehashes to 11 then to 19 and then to 29.

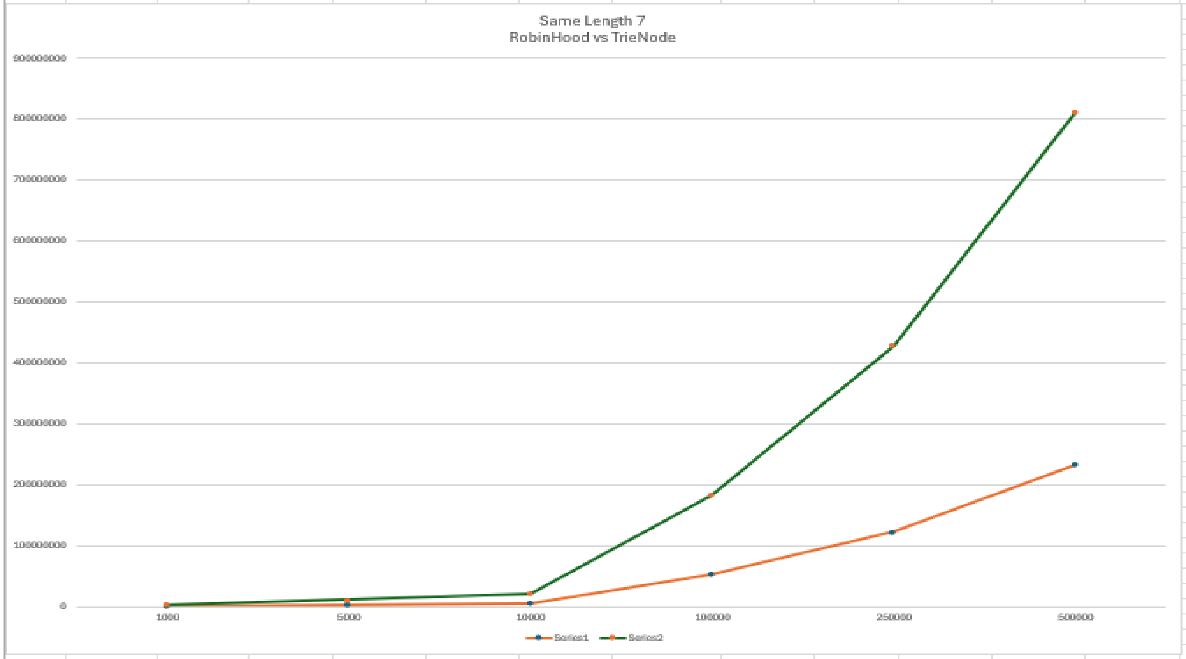
As shown below, while the dictionary length gets bigger, the static table utilizes more useless space as it always generates 26 space tables and the memory utilized is being increased exponentially.

The RobinHood Hashing method is more efficient memory-wise due to its rehashing techniques.



Same Len 7

	RobinHood						TrieNode					
	1	2	3	4	5	m.o.	1	2	3	4	5	m.o.
1000	707120	705968	703920	708784	706864	706531.2	2475200	2471168	2464000	2481024	2474304	2473139.2
5000	3205040	3203632	3205296	3202352	3206790	3204622	11217920	11212992	11218816	11208512	11207840	11213216
10000	6171696	6172720	6176560	6176688	6174256	6174384	21601216	21604800	21618240	21618688	21610176	21610624
100000	52170800	52174512	52160944	52149552	52170800	52165322	182598080	182611072	1.83E+08	182523712	182598080	182578906
250000	122652976	122672048	1.23E+08	122637616	122626096	122643862	429285696	429352448	4.29E+08	429231936	429191616	429253798
500000	231827504	231890992	2.32E+08	231807408	231828144	231834826	811396544	811618752	8.11E+08	811326208	811398784	811422170



Same Length 10:

In this section we generated dictionaries of size n = 1k, 5k, 10k, 100k, 250k and 500k.

Word size = 10.

We run each dictionary using the TrieNode with static array and the TrieNode with the RobinHood Hashing table.

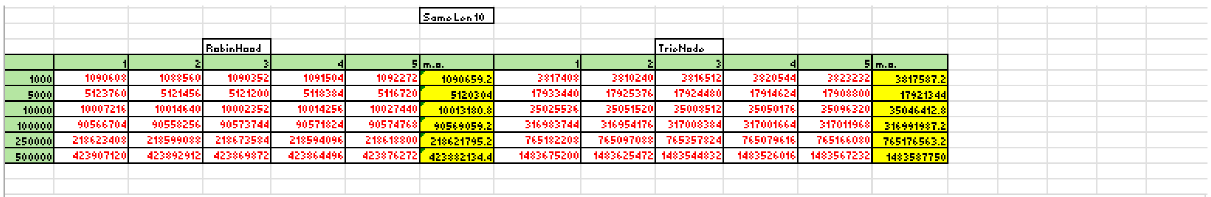
We generated 5 dictionaries for each n and calculated the average of each n for each TrieNode.

The Graph below shows two lines, the blue being the static and the green being the RobinHood.

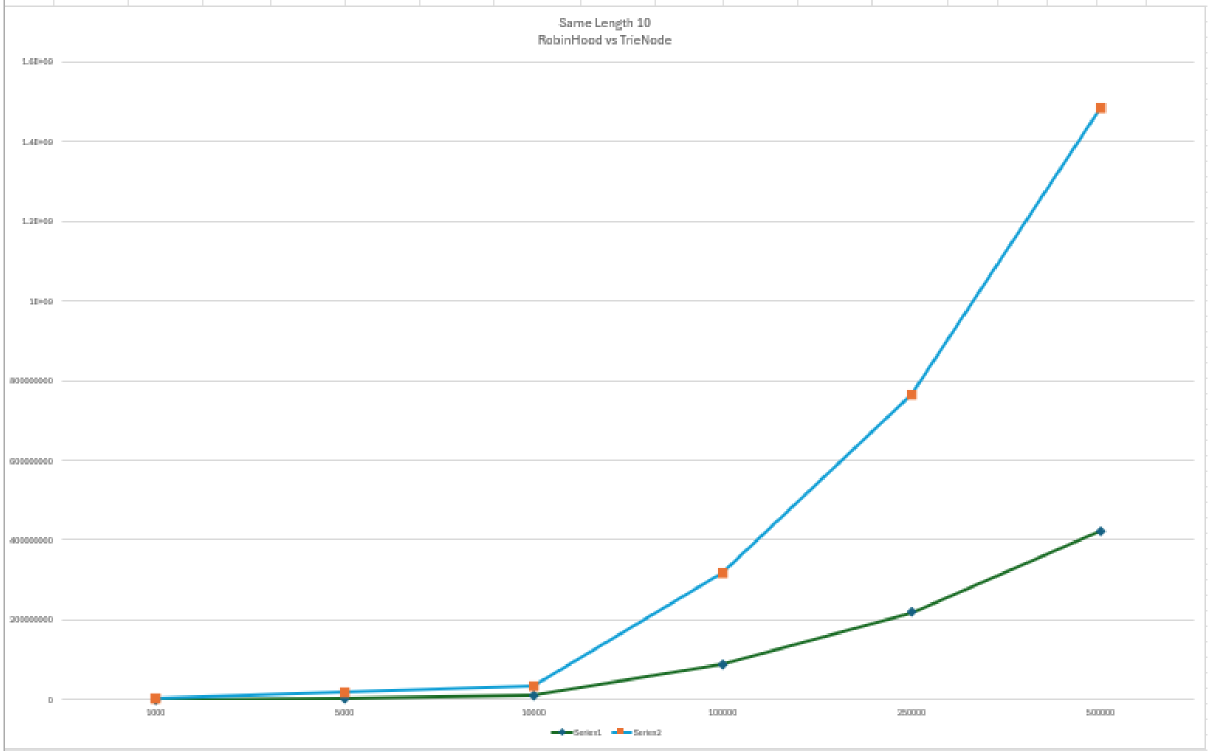
The Static table creates a 26 static array of children while the RobinHood hashing creates a table of 5, then if 90% of the array is used it rehashes to 11 then to 19 and then to 29.

As shown below, while the dictionary length gets bigger, the static table utilizes more useless space as it always generates 26 space tables and the memory utilized is being increased exponentially.

The RobinHood Hashing method is more efficient memory-wise due to its rehashing techniques.



						Same Len 10						
			RobinHood						TrieNode			
	1	2	3	4	5	m.a.	1	2	3	4	5	m.a.
1000	1090608	1088560	1090352	1091504	1092272	1090659.2	3817408	3810240	3816512	3820544	3823232	3817587.2
5000	5123760	5121456	5121200	5118384	5116720	5120304	17933440	17925376	17924480	17914624	17908800	17921344
10000	10007216	10014640	10002352	10014256	10027440	10013180.8	35025536	35051520	35008512	35050176	35096320	35046412.8
100000	90566704	90558256	90573744	90571824	90574768	90569059.2	316983744	316954176	317008384	317001664	317011968	316991987.2
250000	218623408	218599088	218673584	218594096	218618800	218621795.2	765182208	765097088	765357824	765079616	765166080	765176563.2
500000	423907120	423892912	423869872	423864496	423876272	423882134.4	1483675200	1483625472	1483544832	1483526016	1483567232	1483587750



Varying Length 3 - 12 :

In this section we generated dictionaries of size n = 1k, 5k, 10k, 100k, 250k and 500k.

Word size = 3-12 with distribution (3-5 more likely).

We run each dictionary using the TrieNode with static array and the TrieNode with the RobinHood Hashing table.

We generated 5 dictionaries for each n and calculated the average of each n for each TrieNode.

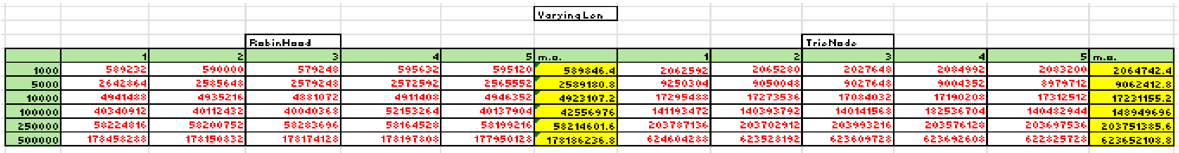
The Graph below shows two lines, the green being the static and the orange being the RobinHood.

The Static table creates a 26 static array of children while the RobinHood hashing creates a table of 5, then if 90% of the array is used it rehashes to 11 then to 19 and then to 29.

As shown below, while the dictionary length gets bigger, the static table utilizes more useless space as it always generates 26 space tables and the memory utilized is being increased exponentially.

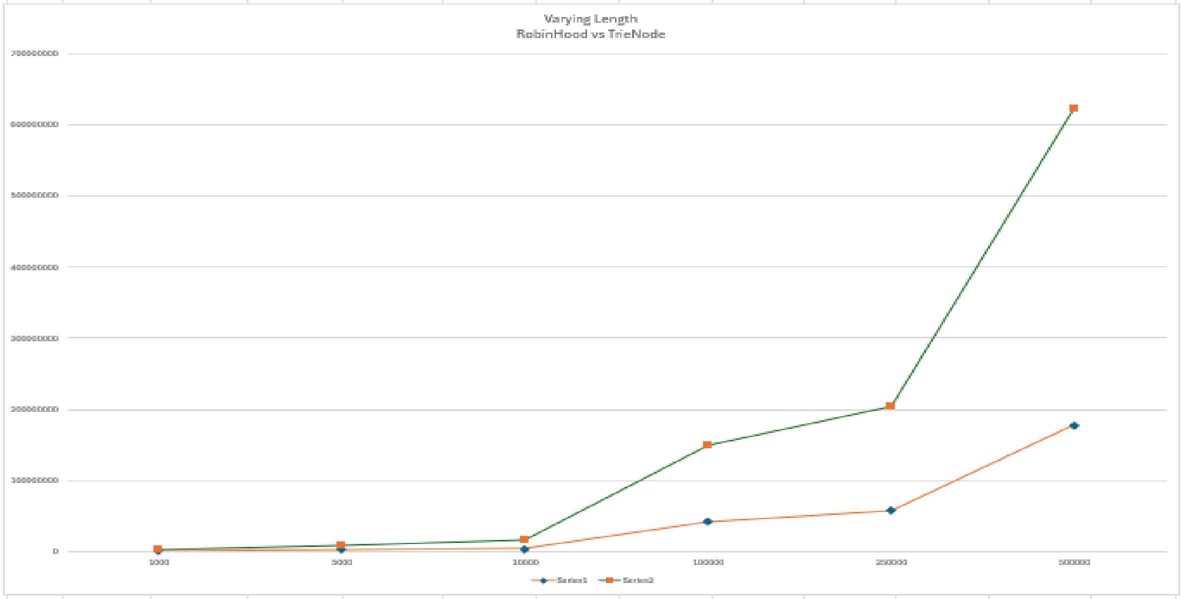
Due to the varying length there is a possibility that bigger words are inserted in the dictionary thus more memory is being utilized for both options. However due to the static array being 26 length, it uses a lot more space thatn the RobinHood Hashing technique.

The RobinHood Hashing method is more efficient memory-wise due to its rehashing techniques.



VaryingLen

	RobinHood						TrieNode					
	1	2	3	4	5	m.o.	1	2	3	4	5	m.o.
1000	589232	590000	579248	595632	595120	589846.4	2062592	2065280	2027648	2084992	2083200	2064742.4
5000	2642864	2585648	2574248	2572592	2565552	2589180.8	9250304	9050048	9027648	9004352	8979712	9062412.8
10000	4941488	4935216	4881072	4911408	4946352	4923107.2	17295488	17273536	17084032	17190208	17312512	17231155.2
100000	40340912	40112432	40040368	52153264	40137904	42556976	141193472	140393792	140141568	182536704	140482944	148949696
250000	58224816	58200752	58283696	58164528	58199216	58214601.6	203787136	203702912	203993216	203576128	203697536	203751385.6
500000	178458288	178150832	178174128	178197808	177950128	178186236.8	624604288	623528192	623609728	623692608	622825728	623652108.8


Varying Length
RobinHood vs TrieNode
Series1
Series2