CO 322-Lab 04
E/15/208

Part 1 - Using Trie
Part2 - Modified Data Structure -> Using RadixTree

Results of the tests

Table 1 : No of nodes

	Wordlist1000.txt	Wordlist10000.txt	Wordlist70000.txt
Trie	3026	24178	224752
Modified Data Structure	1314	11925	90890

Table 2 : Memory taken

	Wordlist1000.txt	Wordlist10000.txt	Wordlist70000.txt
Trie	326808 Bytes	2611224 Bytes	24273216 Bytes
Modified Data Structure	147168 Bytes	1335600 Bytes	10179680 Bytes

Table 3 : Insertion time

	Wordlist1000.txt	Wordlist10000.txt	Wordlist70000.txt
Trie	0.000000 s	0.005000 s	0.058000 s
Modified Data Structure	0.000000 s	0.004000 s	0.037000 s

Table 4 : Auto completion time

	Prefix	Trie	Modified Data Structure
Wordlist1000.txt	st	0.003000 s	0.006000 s
	wai	0.001000 s	0.001000 s
	b	0.004000 s	0.012000 s
	RE	0.003000 s	0.011000 s
Wordlist10000.txt	st	0.008000 s	0.050000 s
	wai	0.001000 s	0.003000 s
	b	0.026000 s	0.148000 s
	RE	0.017000 s	0.122000 s
Wordlist70000.txt	st	0.003000 s	0.360000 s
	wai	0.046000 s	0.006000 s
	b	0.180000 s	1.423000 s
	RE	0.082000 s	0.571000 s

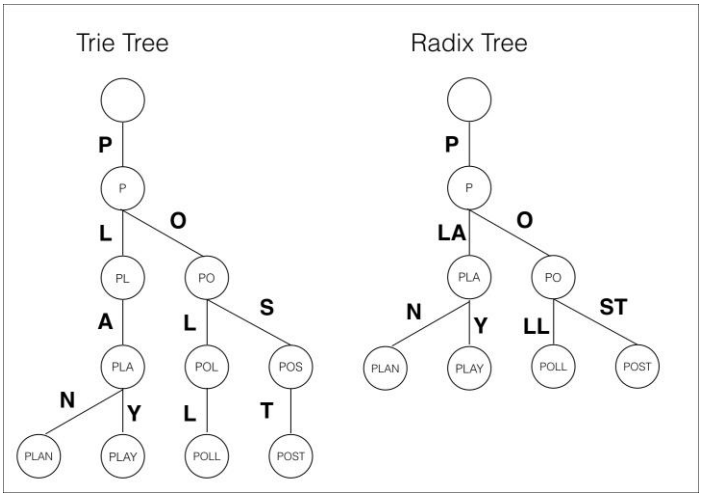
A **radix tree** (also **radix trie** or compact prefix **tree**) is a data structure that represents a space-optimized **trie** (prefix **tree**) in which each node that is the only child is merged with its parent.

There are some differences between a trie and a radix trie. Otherwise the use case of both is pretty much the same.

Below table shows some differences.

Table 5 : Trie vs Radix Trie

Trie	Radix Tree
1. Edge contains a single letter or single part of a key	1. Edges can contain more than a single letter, even an entire word (if we are using them for words/letters)
2. Takes more *memory*	**2. Takes *less memory***
3. Easy to implement	3. Harder to implement



1.Memory Space Usage

As seen in the 2nd point of table 5 , Tries take up a lot of memory space. This can be reduced by using a radix tree instead of using trie.

Table 2 shows how much memory space is taken by each wordlist in order to store the words in both trie and radix trie. There we can see that the memory taken by the trie is larger than the memory taken by radix trie when comparing the same wordlist.

For example, to store Wordlist1000.txt words, trie has taken 326808 Bytes memory space while radix trie has taken only 147168 Bytes memory space. It is nearly 2 times larger than the memory acquired by the radix trie.

2. Time taken to store the dictionary

As shown in the table 3, Radix trie has taken less time in inserting the letters in to the nodes while trie has taken more time than radix tree when comparing the same wordlists.

For example, to insert Wordlist70000.txt words, trie has taken 0.058000 s while radix trie has taken only 0.037000 s.

3. Time taken to print a list of suggestions for chosen word prefixes

As shown in the table 4, Radix trie has taken more time in auto suggesting the words while trie has taken less time than radix tree when comparing the same wordlists according to a same prefix.

For example, to auto suggest the words relating to prefix ‘wai’ of Wordlist10000.txt words, trie has taken 0.001000 s while radix trie has taken only 0.003000 s.