CO 322 Data Structures and Algorithms
Lab 04 - Tree ADT

E/16/267
Parackrama G.T.W.

Comparison Report for the lab

Trie structure is used for **part1** and radix tree structure is used as modified data structure for **part2**.

Here the codes provided memory space usage and time taken to store and search are compared below. **(Here all the words are converted to lowercase)**

1. **Memory Space usage**

Case1 – wordlist1000.txt,

	Trie (Bytes)	Radix tree (Bytes)
Node size	216	224
Total number of nodes	3027	687
Used Memory space	653832	153888

Case2 – wordlist10000.txt,

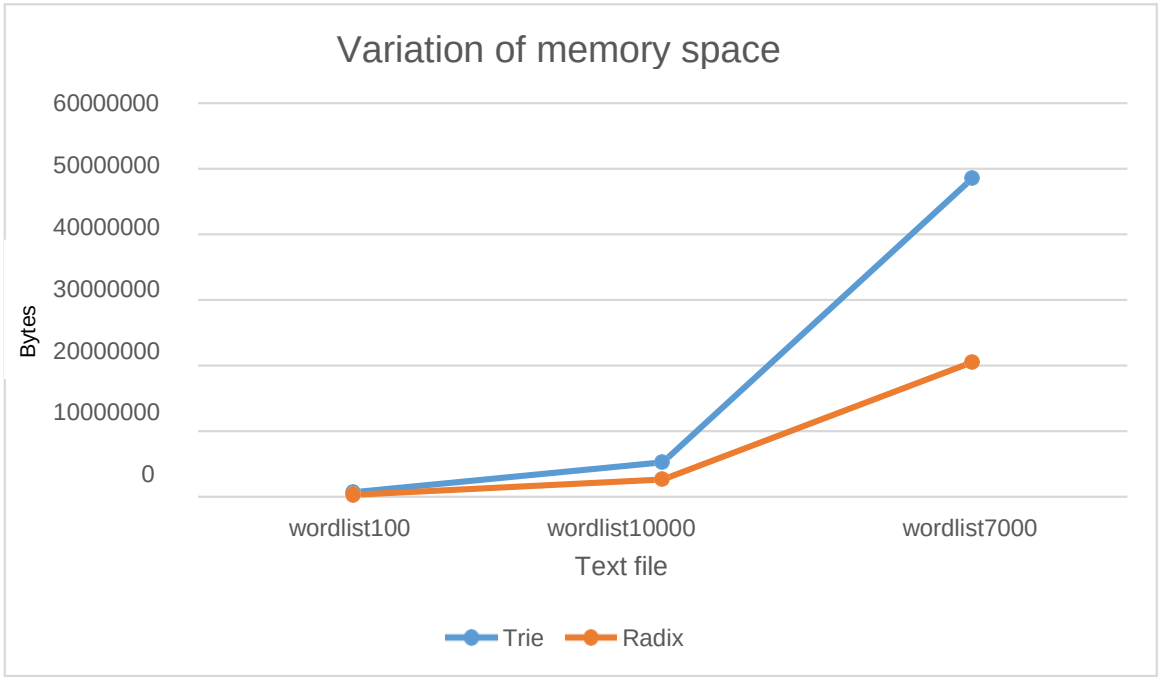
	Trie (Bytes)	Radix tree (Bytes)
Node size	216	224
Total number of nodes	24179	8297
Used Memory space	5222664	1858528

Case3 – wordlist70000.txt,

	Trie (Bytes)	Radix tree (Bytes)
Node size	216	224
Total number of nodes	224753	48207
Used Memory space	48546648	10798368

In a Trie,it has an empty root node, with links to other nodes one for each possible alphabetic value. The the structure and shape of a trie is always a set of linked nodes, connecting back to an empty root node. The number of child nodes in a trie depends completely on the total number of values possible. For example, if we are representing the English alphabet, then the total number of child nodes is directly connected to the total number of letters possible. Since there are 26 letters in In the English alphabet, the total number of child nodes will be 26. But tries need a lot of memory for storing the strings. For each node we have too many node pointers (equal to number of characters of the alphabet).

Radix 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. In this structure number of nodes used for storing words is very low compared to trie structure



According to the graph and the tables it shows that memory space which needs for Radix tree is low than tries. As an example memory space taken to store set of words in wordlist1000.txt in trie is 653832 bytes and radix tree only takes 294336 bytes

2.Time taken to store the dictionary

Case1 – wordlist1000.txt,

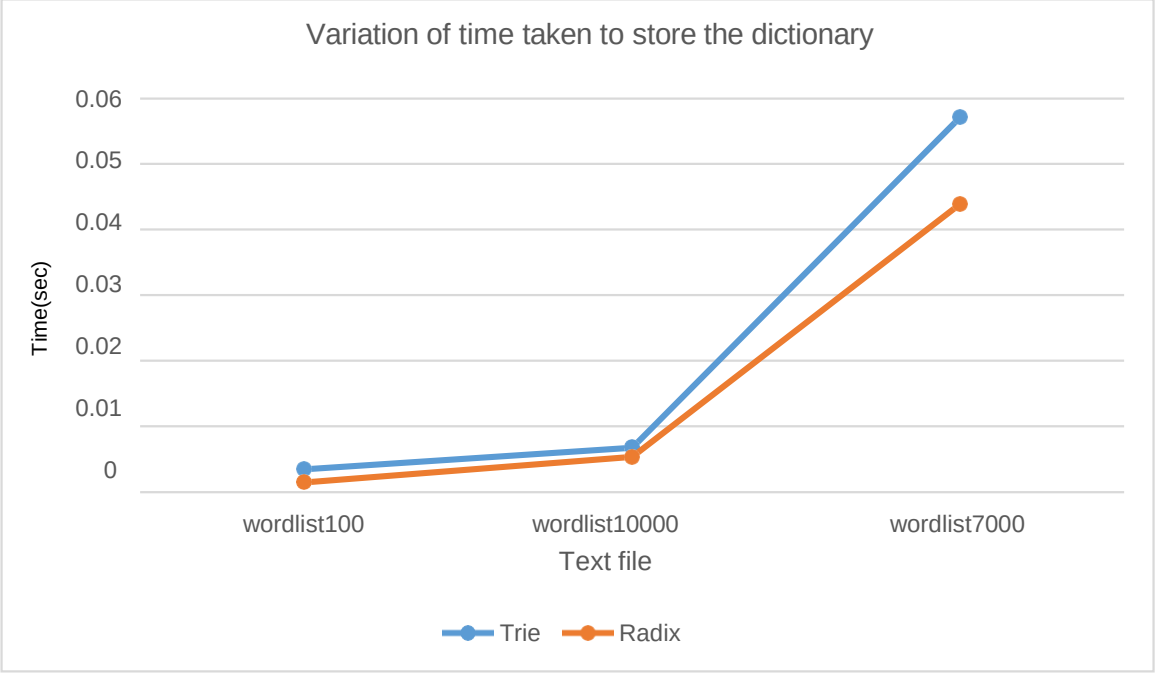
	Trie (s)	Radix tree (s)
Time taken to store	0.003462	0.001448

Case2 – wordlist10000.txt,

	Trie (s)	Radix tree (s)
Time taken to store	0.030642	0.015625

Case3 – wordlist70000.txt,

	Trie (s)	Radix tree (s)
Time taken to store	0.078125	0.062500



If string is of length n, then using trie worst case time complexity for searching the string is O(n). Insertion also takes O(n) time in worst case.

According to the above data and the graph it clearly shows that, time taken to store the same set of words is low in radix tree structure when compared to trie node structure,As an example for storing wordlist10000.txt trie structure takes 0.030642 seconds and radix tree takes only 0.015625 seconds.

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

Case1 – wordlist1000.txt,

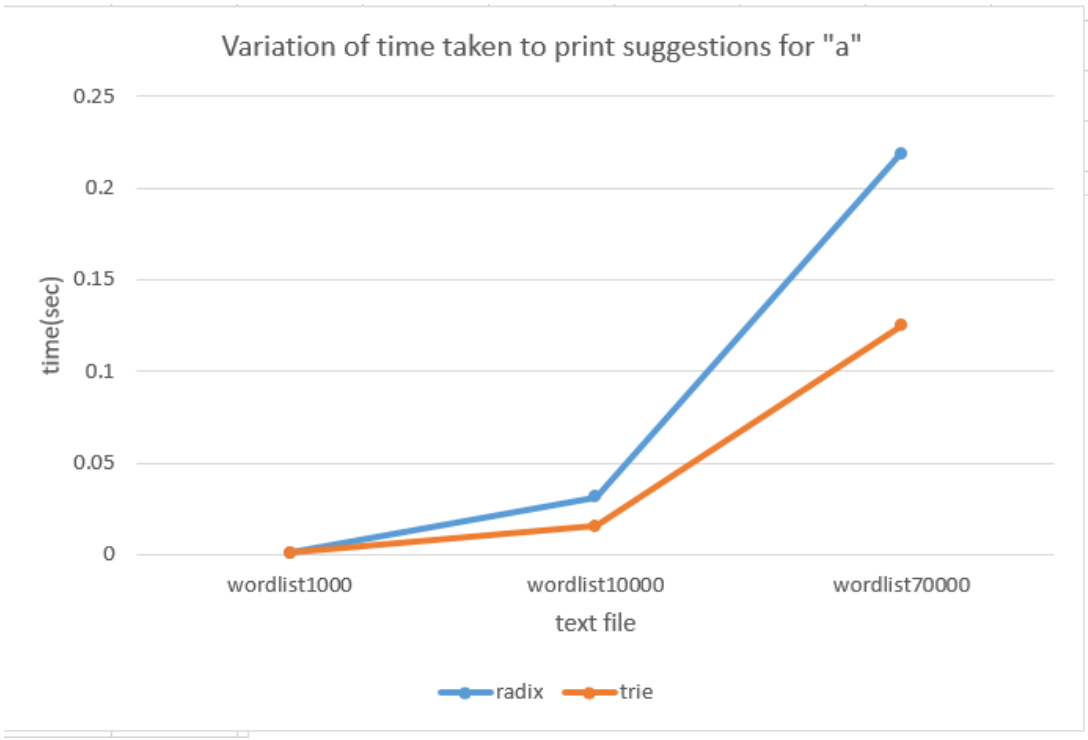
Prefix	Radix tree (seconds)	Trie (seconds)
you	0.000060	0.000053
a	0.001046	0.001014
we	0.000203	0.000150
ra	0.000095	0.000081

Case2 – wordlist10000.txt,

Prefix	Radix tree (seconds)	Trie (seconds)
you	0.000109	0.000075
a	0.031250	0.015625
we	0.000068	0.000055
ra	0.000195	0.000139

Case3 – wordlist70000.txt,

Prefix	Radix tree (seconds)	Trie (seconds)
you	0.000233	0.000193
a	0.218750	0.125000,
we	0.000684	0.000395
ra	0.001856	0.001229



As we can see in the above tables time taken to print a suggestion list for a given prefix is littlle bit high in radix structure and low in trie tree structure. As an example for prefix “a” to print the suggestion list in wordlist1000.txt trie tree structure takes only 0.001014 and radix takes 0.001046 seconds.
So considering these 3 senarios, Time taken to print a list of suggestions for chosen word prefixes is bit high in the radix trie. When wordlist(prefix) is small difference is non considerable when comparing the tables above. So we can say time taken to print a list of suggestions for chosen word prefixes is almost same in two methods except suggestions are really long.
So Finally we can say that it is clear the Radix tree structure uses less memory space and less time to search than trie tree structure, but for insertion trie uses less time compared to radix. And Radix trie is faster than the regular trie when store happen.

So since Tries take up a lot of memory space. Radix is away to reduce the space usage of Trie structure by removing unnecessary nodes without losing any information.