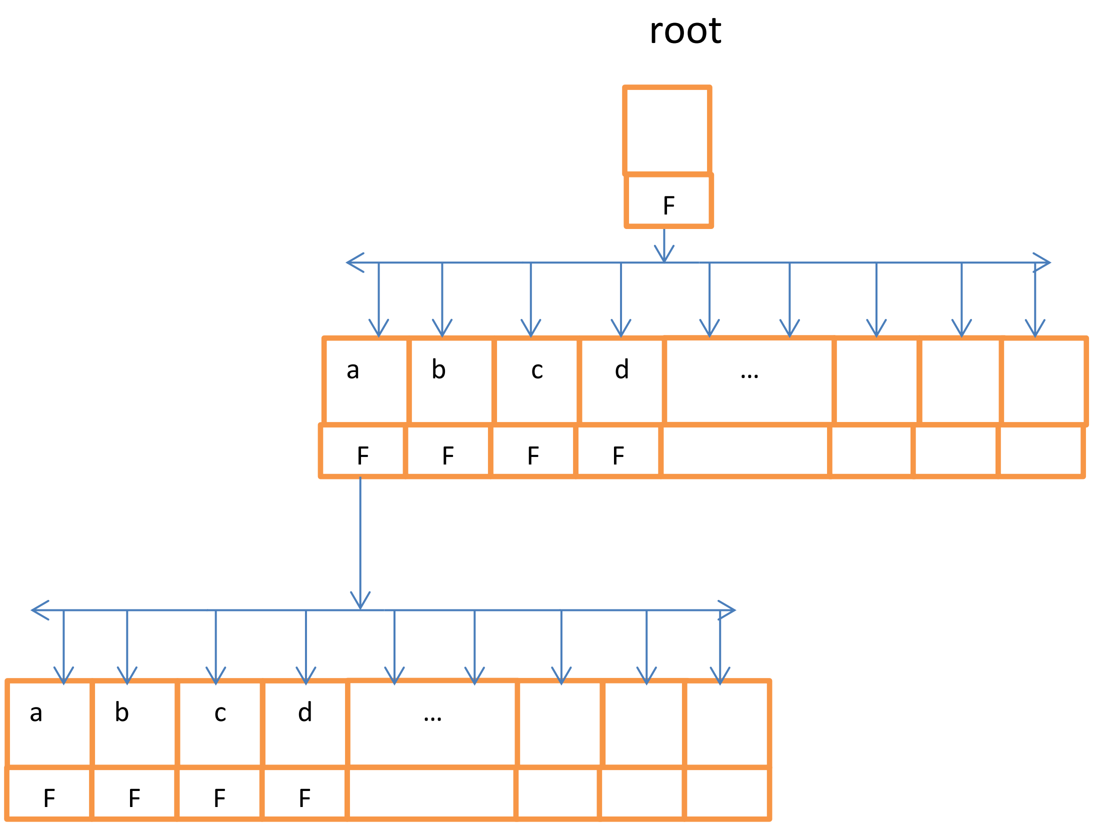
<u>Lab04</u>

<u>Report</u>

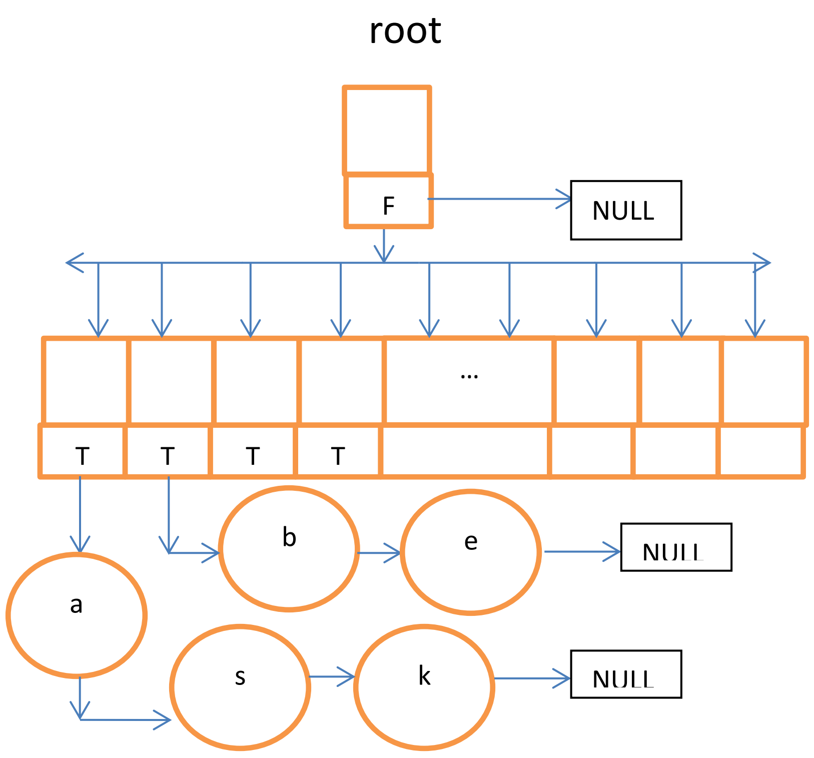
Thilakarathna K.A.H.P.

E/15/362

Following is the trie data structure that I have implemented in the first part.



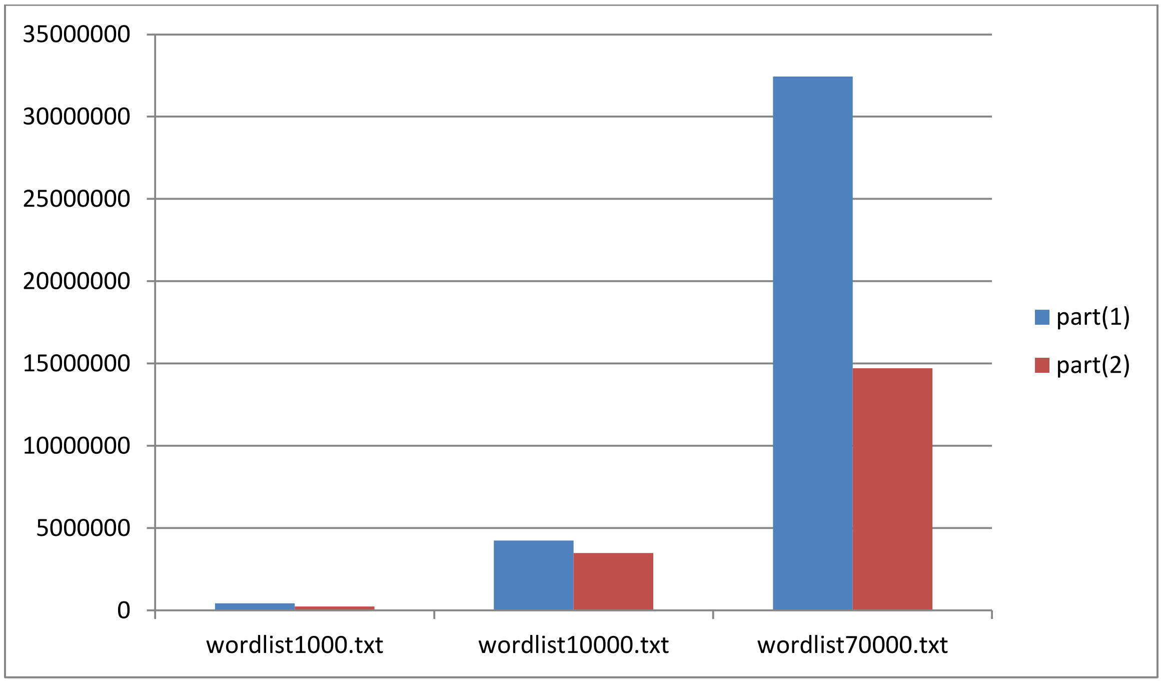
Following is the trie data structure I have implemented in the second part with space reduction.



Here characters are linked with using a kind of a linked structure to store required words. Only the needed characters are stored in the trie unlike in the previous implementation. Hence, the number of nodes used is reduced preserving the memory space.

1. <u>memory space usage</u>

	Memory usage-part(1)/ Bytes	Memory usage part(2)/ Bytes
wordlist1000.txt	424000	239760
wordlist10000.txt	4240424	3488832
wordlist70000.txt	32436384	14718672



When the second implementation is used, memory usage is reduced in a considerable amount as you can see in the above table and graph. This is because in the second implementation linked structure for storing the words is used so that only the needed characters are stored.

2. **<u>time taken to store the dictionary</u>**

   If we run the two codes separately with any prefix of our choice, we get the results in the ranges given below. It doesn't give an exact time but slightly different values in different times.

	Time taken part(1)/ seconds	Time taken part(2)/ seconds
wordlist1000.txt	0.004-0.02	0.003-0.006
wordlist10000.txt	0.05 – 0.08	0.04-0.06
wordlist70000.txt	0.7-0.8	0.5-0.6

For all three text files, the time taken to store the words are less in the second implementation while the first implementation takes comparatively more time for storing words. Hence, we can conclude that the second implementation is better when taking the time taken to store the dictionary as a factor.

3. **<u>time taken to print a list of suggestions for chosen word prefixes</u>**

   Let's say we input prefix 'ab' ,

	Time taken part(1)/ seconds	Time taken part(2)/ seconds
wordlist1000.txt	0.000022-0.0008	0.000017-0.000023
wordlist10000.txt	0.000085-0.002	0.000071-0.002
wordlist70000.txt	0.003-0.004	0.003-0.01

As you can see from the above table, trie implementation in the second part takes less time to print the suggestions for the prefix 'ab' for first two test files and it takes a little bit more time to print suggestions, when the wordlist70000.txt is used.

Let’s give the input as prefix ‘as’

	Time taken part(1)/ seconds	Time taken part(2)/ seconds
wordlist1000.txt	0.000015-0.000022	0.000013-0.000021
wordlist10000.txt	0.00018-0.00088	0.00016-0.002
wordlist70000.txt	0.003-0.01	0.002-0.007

Here for all three text files, less time is taken when the second implementation is used.

Thus, we can conclude that the time taken for printing a suggestion list for a given prefix is less when the second implementation is used, when considering both the above two cases averagely. But it could change slightly for different inputs.