Department of Computer Engineering University of Peradeniya

CO 322 Data Structures and Algorithms

Lab 04 – Tree ADT

Name : Jayathilaka H.A.D.T.T.

E.Number : E/16/156

Semester : 5

Date : 20/08/2020

A Trie is an ordered data structure, a type of search tree used to store associative data structures. It is also referred to as a Radix tree or Prefix tree. The Trie is used to store strings that can be visualized like a graph. It consists of nodes and edges. Each node consists of at max 26 children and edges connect each parent node to its children. These 26 pointers are nothing but pointers for each of the 26 letters of the English alphabet A separate edge is maintained for every edge.

The main disadvantage of tries is that they 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).

The final conclusion is regarding *tries data structure* is that they are faster but require huge memory for storing the strings.

So, we can use a radix tree instead of a trie.

A radix tree is like a trie, but it saves space by combining nodes together if they only have one child. So, this does not need a lot of memory for storing strings.

Let us consider lab4part1.c (A program for storing strings using trie) and lab4part2.c (A program for storing strings using radix tree).

The behavior of two programs, when using the same non complete word (**add**), is given below.

1. Using wordlist1000.txt

1.1 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 108000 Bytes
Time taken to store : 0.00100000 s
Time taken to search :

1.2 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 76944 Bytes
Time taken to store : 0.00000000 s
Time taken to search : 0.002000 s

2. Using wordlist10000.txt

2.1 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 1080108 Bytes
Time taken to store : 0.00800000 s
Time taken to search : 0.020000 s

2.2 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 904512 Bytes
Time taken to store : 0.00600000 s
Time taken to search : 0.013000 s

3. Using wordlist70000.txt

3.1 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 7549632 Bytes
Time taken to store : 0.06200000 s
Time taken to search : 0.052000 s

3.2 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 5399184 Bytes
Time taken to store : 0.05900000 s
Time taken to search : 0.044000 s

The behavior of two programs, when using the same non complete word (**second**), is given below.

1. Using wordlist1000.txt

1.3 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 108000 Bytes
Time taken to store : 0.00100000 s
Time taken to search : 0.005000 s

1.4 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 76944 Bytes
Time taken to store : 0.00000000 s
Time taken to search : 0.003000 s

2. Using wordlist10000.txt

2.1 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 1080108 Bytes
Time taken to store : 0.00800000 s
Time taken to search : 0.008000 s

2.2 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 904512 Bytes
Time taken to store : 0.00600000 s
Time taken to search : 0.004000 s

3. Using wordlist70000.txt

3.3 For lab4part1.c (A program for storing strings using trie)

Memory space usage : 7549632 Bytes
Time taken to store : 0.06200000 s
Time taken to search : 0.022000 s

3.4 For lab4part2.c (A program for storing strings using radix tree)

Memory space usage : 5399184 Bytes
Time taken to store : 0.05900000 s
Time taken to search : 0.015000 s

If consider the above results for two parts, a considerable reason can be seen. In the first part which was built using trie, has taken more time and more memory space, while the second part which was built using radix tree has taken less time and less memory space.

Also when printing the suggestions, radix tree has taken less time compared to trie. So, it is clear that the Radix tree structure uses less time to search than trie tree structure.

If we consider two non complete words also, the result does not change. Trie tree structure has taken more seaching time. So, it prove that the Radix tree structure uses less time to search than trie tree structure and the searching time doesnot depend on the keyword.

So, when compared to trie tree structure, the radix tree structure is more efficient.