Department of Computer Engineering
Faculty of Engineering, University of Peradeniya

CO322 Data Structures and Algorithms - 2019

Lab 4 - Tree ADT

Radix Tree (Trie) Structure for Text Auto-complete

Radix Trees or Tries are data structures that provide fast data re**trie**val, at the cost of high space consumption. Typically, data values are associated with edges rather than with nodes in Tries, although this may depend on implementation. In this lab you will be implementing a Trie data structure in C to store a dictionary of English words, and use it to quickly retrieve words for an *text auto-complete* application.

Part 1: 50%

Design and implement a Trie node structure and *associated operations and algorithms* to store and retrieve a collection of English words. Then *design and implement* an algorithm to auto-complete words (provide a short list of suggestions) as a user types in a prefix string. For example, if a user types *'att'*, your program might suggest *'attack*, *'attention'*, *etc.*

You may use the following skeletons to start with, and add additional functions as needed. Complete this part and show your work to an instructor.

```
typedef struct trienode{
       ...
}TrieNode;

TrieNode* createNode(...);

TrieNode* insertWord(...);

int printSuggestions(...);
```

Part 2: 50%

As mentioned earlier, Tries take up a lot of memory space. Come up with a way to reduce the space usage of your Trie structure by removing unnecessary nodes *without losing any information*. Modify your data structure and operations accordingly. Using the same word collection, compare the modified data structure with the previous one for:

1. memory space usage
2. time taken to store the dictionary
3. time taken to print a list of suggestions for chosen word prefixes.

Do multiple tests and report your results with a discussion. Show your work to an instructor and submit (codes for parts 1 & 2 along with the PDF report).