Department of Computer Engineering
Faculty of Engineering, University of Peradeniya

CO322 Data Structures and Algorithms - 2018

Lab Exercises - Trees

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:

Implement a Trie node structure and associated operations to store a dictionary, and auto-complete words (provide a list of suggestions) when a user types in a prefix string (if a user types *'app'*, your program should suggest *'apple'*, *'application'*, *etc.*). You may use the following skeletons, and add additional functions as needed. Show your work to an instructor.

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

TrieNode* createNode(...);

TrieNode* insertWord(...);

int printSuggestions(...);
```

Part 2:

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, and modify your structure and operations accordingly. Using the same dictionary file, 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 retrieve words

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