

Department of Computer Engineering  
University of Peradeniya

CO 322 Data Structures and Algorithms

Lab 04 - Tree ADT

20<sup>th</sup> of August 2020

1. Radix Tree (Trie) Structure for Text Auto-complete

Radix Trees or Tries are data structures that provide fast data retrieval, 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.

2. Part I

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.

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

TrieNode* createNode(...);
TrieNode* insertWord(...);
int printSuggestions(...);
```

3. Part II

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:

- (a) memory space usage
- (b) time taken to store the dictionary
- (c) time taken to print a list of suggestions for chosen word prefixes

4. Submission

Do **multiple tests** and report your results with a discussion. Submit a zip file with codes for parts 1,2 and the PDF report. Deadline for submission **03rd September 2020**.