

Trie and Ternary Search Tree Implementation Report

Tousif

November 13, 2023

1 Trie Implementation

1.1 Overview

The Trie data structure is used to compile an alphabetical index of words and the lines on which they appear. The implementation includes the following functions:

- `Trie()`
- `void insert(const string& word, int line)`
- `void printIndex()`
- `void printIndexHelper(TrieNode* node, string wordSoFar)`
- `~TrieNode()`
- `~Trie()`

1.2 Function Details

1. `Trie()`

Description: Constructor for the Trie class. Initializes the root node of the Trie.

Time Complexity: $O(1)$

2. `void insert(const string& word, int line)`

Description: Inserts a word and the line number into the Trie.

Time Complexity: $O(L)$, where L is the length of the word.

3. `void printIndex()`

Description: Prints the alphabetical index using the Trie.

Time Complexity: $O(N)$, where N is the total number of nodes in the Trie.

4. `void printIndexHelper(TrieNode* node, string wordSoFar)`

Description: Recursive helper function for printing the alphabetical index.

Time Complexity: $O(N)$, where N is the total number of nodes in the Trie.

5. `~TrieNode()`

Description: Destructor for the TrieNode class. Frees the memory allocated for the TrieNode and its children.

Time Complexity: $O(1)$

6. `~Trie()`

Description: Destructor for the Trie class. Calls the destructor for the root TrieNode, freeing all memory.

Time Complexity: $O(N)$, where N is the total number of nodes in the Trie.

2 Ternary Search Tree (TST) Implementation

2.1 Overview

The Ternary Search Tree (TST) data structure is used for the same purpose as the Trie. The implementation includes the following functions:

- TernarySearchTree()
- void insert(const string& word, int line)
- void printIndex()
- void printIndexHelper(TSTNode* node, string wordSoFar)
- ~TSTNode()
- ~TernarySearchTree()

2.2 Function Details

1. TernarySearchTree()

Description: Constructor for the TernarySearchTree class. Initializes the root node of the TST.

Time Complexity: $O(1)$

2. void insert(const string& word, int line)

Description: Inserts a word and the line number into the TST.

Time Complexity: $O(L)$, where L is the length of the word.

3. void printIndex()

Description: Prints the alphabetical index using the TST.

Time Complexity: $O(N)$, where N is the total number of nodes in the TST.

4. void printIndexHelper(TSTNode* node, string wordSoFar)

Description: Recursive helper function for printing the alphabetical index.

Time Complexity: $O(N)$, where N is the total number of nodes in the TST.

5. ~TSTNode()

Description: Destructor for the TSTNode class. Frees the memory allocated for the TSTNode and its children.

Time Complexity: $O(1)$

6. ~TernarySearchTree()

Description: Destructor for the TernarySearchTree class. Calls the destructor for the root TSTNode, freeing all memory.

Time Complexity: $O(N)$, where N is the total number of nodes in the TST.

3 Memory Management

3.1 Trie

3.2 TST

```

tihaz@ecc-linux2:~/CSCI311/Projects/Project 3$ valgrind --leak-check=full --show-leak-kinds=all ./Project3Part1
==593605== Memcheck, a memory error detector
==593605== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==593605== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==593605== Command: ./Project3Part1
==593605==
fdfsd
ygfh
fdfsd: 1
ygfh: 2
==593605==
==593605== HEAP SUMMARY:
==593605==    in use at exit: 0 bytes in 0 blocks
==593605==   total heap usage: 19 allocs, 19 frees, 77,696 bytes allocated
==593605==
==593605== All heap blocks were freed -- no leaks are possible
==593605==
==593605== For lists of detected and suppressed errors, rerun with: -s
==593605== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Figure 1: Valgrind output for Trie

```

tihaz@ecc-linux2:~/CSCI311/Projects/Project 3$ g++ -std=c++11 -g Project3Part2.cpp -o Project3Part2
tihaz@ecc-linux2:~/CSCI311/Projects/Project 3$ valgrind --leak-check=full --show-leak-kinds=all ./Project3Part2
==594780== Memcheck, a memory error detector
==594780== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==594780== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==594780== Command: ./Project3Part2
==594780==
red
flag
free
flag: 2
free: 3
red: 1
==594780==
==594780== HEAP SUMMARY:
==594780==    in use at exit: 0 bytes in 0 blocks
==594780==   total heap usage: 22 allocs, 22 frees, 76,128 bytes allocated
==594780==
==594780== All heap blocks were freed -- no leaks are possible
==594780==
==594780== For lists of detected and suppressed errors, rerun with: -s
==594780== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Figure 2: Valgrind output for TST