VIETNAM NATIONAL UNIVERSITY, HCMC
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

# DATA STRUCTURES AND ALGORITHMS

PROJECT REPORT - 19CLC3

## CHALLENGE 2

# TRIES

**GROUP 03879698**

- Đỗ Tiến Trung 19127603
- Vũ Tuấn Hải 19127387
- Trương Quang Minh Nhật 19127496
- Lê Yến Nhi 19127498

**LECTURERS**

- PhD. Nguyễn Hải Minh
- M.Sc. Bùi Huy Thông
- M.Sc. Trần Thị Thảo Nhi

HO CHI MINH CITY, DECEMBER 2020

# 1

# WORKING PROGRESS

## Group Information

| Student ID | Full Name | Email | Roles |
|---|---|---|---|
| 19127603 | Đỗ Tiến Trung | 19127603@student.hcmus.edu.vn | Coder |
| 19127387 | Vũ Tuấn Hải | 19127387@student.hcmus.edu.vn | Developer |
| 19127496 | Trương Quang Minh Nhật | 19127496@student.hcmus.edu.vn | Tester, Leader |
| 19127498 | Lê Yến Nhi | 19127498@student.hcmus.edu.vn | Secretary |

## Project Information

| Project name | TRIES | |
|---|---|---|
| Tools and functions | Messenger | Communicate each other |
| | CodeBlocks | Coding |
| | Google Documents, Office 365 | Write report and functions |
| | ZOOM Cloud Meetings | Meeting |

# Meeting

## 1. Overview

**Group name:** Group 03879698

**Members:** Đỗ Tiến Trung, Vũ Tuấn Hải, Trương Quang Minh Nhật, Lê Yến Nhi

**Purpose of the meeting:**

1. Discuss about the project.
2. Get the division of works.

**Place:** ZOOM Cloud Meetings, Messenger.

**Date:** Monday 30/11/2020

**Time:** 9:30' – 11am

**Status:** 100% DONE

## 2. Table of works

| ID | NAME | DESCRIPTION | STATUS |
|---|---|---|---|
| 19127603 | Đỗ Tiến Trung | - Coder<br>- Tester | DONE |
| 19127387 | Vũ Tuấn Hải | - Project Manager<br>- Developer | DONE |
| 19127496 | Trương Quang Minh Nhật | - Coder<br>- Tester | DONE |
| 19127498 | Lê Yến Nhi | - Write report<br>- Summarize the project | DONE |

# 2

# RESEARCH ANSWERS

## 2.1 The time complexity

The time complexity of the following operations (of Trie)

• Adding a word : O(n) where n is the length of the word being added.
• Removing a word : O(n) where n is the length of the word being removed.
• Searching a word : O(n) where n is the length of the word being searched.
• Searching words which has the same prefix with length i: O(m * k) where m is the number of words found and k is the difference of "max of length word" and i.

## 2.2 The advantages of Trie

- **The main advantages of trie over binary search trees:**

- The search time of Trie is less than the search time of the binary search trees.
- The search for a key of length m requires **O(m)** character comparison. A search binary tree uses **O(logn)** string comparison (n is the number of keys). In the worst case, the search binary tree requires **O(m logn)** character comparison.
- Trie uses less memory because the common prefix only needs to be stored once.
- Trie allows searching for the longest polymerization prefix.
- The number of nodes from root to leaf is exactly equal to the length of the key.

- **The main advantages of trie over hash tables:**

- Trie allows to list keys in dictionary order.
- Trie allows to search for the longest polymerization prefix.
- Because trie does not have to calculate a hash function, trie is usually faster than a hash table in case of small keys such as integers or pointers.

# 3

# INFORMATION OF CODE FRAGMENTS

## 3.1 Data structures

**Trie** (**digital tree** or **prefix tree**) is a kind of search tree - an ordered tree data structure used to store a dynamic set or associative array where the keys are usually string.

## 3.2 Algorithms

```
// Main void main()
```

Import Dictionary

```
TrieNode* root = createNode();
sendWordToTrie(root, "Dic.txt");
```

Read from file "Input.txt"

```
- if required action cannot open file
                print "Can't open file Input.txt"
- else
      // INPUT
  while (if character is not end of file)
  {
      - read a whole line (input)
      - if input is a valid string, break.
  }
```

```
        // OUTPUT
   vector <char> characters = Store strings that removed spaces
   vector <string> results  = Store answer strings
        - print the number of words of output
        - for (int i = 0 → Number of words of output)
              print all words as result

        // DELETE TRIES
  Remove Trie
```

```
// Check input is a valid string
bool isValidInput(string input)

    int charMin = the ASCII value of 'a';
    for (int i = 0 → Number of characters of input)
    {
        if character of "i" in ASCII code is letter (i % 2 == 0)
        {
            if input[i] is lower case letter and sorted in
alphabetical order
            {
                charMin = the ASCII value of (input[i]);
            }
            else return false;
        }
        else
        {
            if (input[i] is not a character ' ')
            {
                return false;
            }
        }
    }
```

```
// Check for validity of a found string
bool isValidString(string found, vector <char> characters)
```

**countChar** is used for counting word in characters(countChar = 1)

Because we take the characters in vector<char> to compare with character in found(key). Characters that we are examining in vector <char> character always exist.

**countKey** is used for counting word in string found(key).

When we compare character[i] with character[i+1], there will be error in vector<char> so i push back '/' to solve that problem and after checking string found, i pop back it.

## 3.3 Functions

- Return the ASCII value of a character

```
int charToIndex(char c)
```

- Return character of a ASCII code

```
char indexToChar(int i)
```

- Add one key into Trie

```
void Insert(TrieNode*& root, string key)
```

- Read from file "Dic.txt"

```
void sendWordToTrie(TrieNode*& root, string file_to_read)
```

- Check input is a valid string

```
bool isValidInput(string input)
```

- Remove spaces from input string

```
vector <char> splitToChar(string input)
```

- Check for validity of a found string

```
bool isValidString(string found, vector <char> characters)
```

- Check for existence of a string

```
bool isExistBefore(vector <string> results, string found)
```

- Search for a word based on key "found"

```
void searchWord(TrieNode* root, string found,
                vector <string>&results, vector <char> characters)
```

- Find all valid words and return result

```
vector <string> findAllWords(TrieNode* root,
                                        vector <char> characters)
```

- Remove Trie

```
void deleteTrie(TrieNode*& root)
```

# 4

# REFERENCES

https://www.geeksforgeeks.org/trie-insert-and-search/?fbclid=IwAR0h7j-f5qx_LgascJyaQejVVhT1_uwtbUQN0kE0sS9coyFd8LakN1qF5nE

https://www.geeksforgeeks.org/print-valid-words-possible-using-characters-array/?fbclid=IwAR3YUXej3h4gOcsNySGxZYIVNGSGd9XSDyDPmLZPolC90pA5yZWHFFaU3FM

https://en.wikipedia.org/wiki/Trie