國立臺南大學資訊工程學系

資工三「演算法」課程

第三次作業

**題目: Dictionary**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 曾俊杰 |
| 學號 | ： | S11059003 |

老師：陳宗禧

中華民國 112年12月10日

**目錄**

[(一) 簡介及問題描述 3](#_Toc153303203)

[**1. 簡介 3**](#_Toc153303204)

[**2. 問題 3**](#_Toc153303205)

[(二) 理論分析 4](#_Toc153303206)

[**1. Binary Search Tree理論 4**](#_Toc153303207)

[**2. Splay Tree理論 4**](#_Toc153303208)

[**3. Treap理論 4**](#_Toc153303209)

[(三) 演算法則 5](#_Toc153303210)

[**1. 第一個演算法(BST) 5**](#_Toc153303211)

[**i. 演算法時間複雜度(time complexity): O(*log(n)*) 7**](#_Toc153303212)

[**ii. 演算法空間複雜度(space complexity): O(*n*) 7**](#_Toc153303213)

[**2. 第二個演算法(SPT) 7**](#_Toc153303214)

[**i. 演算法時間複雜度(time complexity): O(*log(n)*) 11**](#_Toc153303215)

[**ii. 演算法空間複雜度(space complexity): O(*n*) 11**](#_Toc153303216)

[**3. 第三個演算法(Algorithm) 12**](#_Toc153303217)

[**i. 演算法時間複雜度(time complexity): O(*log(n)*) 14**](#_Toc153303218)

[**ii. 演算法空間複雜度(space complexity): O(*n*) 14**](#_Toc153303219)

[(四) 程式設計環境架構 15](#_Toc153303220)

[**1. 程式語言 15**](#_Toc153303221)

[**2. 程式開發工具 15**](#_Toc153303222)

[**3. 電腦硬體 15**](#_Toc153303223)

[**4. 作業系統 15**](#_Toc153303224)

[(五) 程式 16](#_Toc153303225)

[**1. 主程式 16**](#_Toc153303226)

[**(1) BST 16**](#_Toc153303227)

[**(2) SPT 25**](#_Toc153303228)

[**(3) Treap 36**](#_Toc153303229)

[**2. Input Code Format 45**](#_Toc153303230)

[**(1) Test article 45**](#_Toc153303231)

[**(2) Test(Search) with files 46**](#_Toc153303232)

[**3. Output Code Format 48**](#_Toc153303233)

[**(1) Constructure result(same for each search) 48**](#_Toc153303234)

[**(2) TestFile11.txt 49**](#_Toc153303235)

[**(3) TestFile12.txt 49**](#_Toc153303236)

[**(4) TestFile13.txt 50**](#_Toc153303237)

[(六) 執行結果、討論與心得 51](#_Toc153303238)

[**1. 執行結果 51**](#_Toc153303239)

[**(1) BST 52**](#_Toc153303240)

[**(2) SPT 52**](#_Toc153303241)

[**(3) Treap 52**](#_Toc153303242)

[**2. 討論 52**](#_Toc153303243)

[**3. 心得 53**](#_Toc153303244)

[參考文獻 55](#_Toc153303245)

**(一) 簡介及問題描述**

1. 簡介

利用 Binary Search Tree (BST), Splay Tree (SPT), and Treap 分別建構相對應的Dictionary，並且利用該 Dictionary 做 Searching 以及其時間分析，包括分析查詢的時間、成功率與失敗率等；此外，亦要討論 Dictionary size and Searching file size 與執行時間與效率的關係!

2. 問題

對於三種資料結構分別做Insert a word，Delete a word，Seach a word，Seach a file，其中Search a file是將多行以tab做區分的字串作為輸入，進行Search。其中，InputFile1.txt、InputFile2.txt是測試用的文章建立Trees，然後用test.txt對程式進行測試。

Token定義:

* Input file(article): 以空格作為分隔的字串(無論標點符號或tab，只看空格)

Ex. This is a good day. -> This, is, a, good, day.，總共5個字串；This, This (This後加一個空格)是不同的字串

* Input test file with a word: 以單行為準，一整行的輸入當作是一個word

Ex. This is a good day. -> 一個word

* Input test file with a file: 以單行為準，一整行輸入當作是一個word，下一行是另一個word

Ex. This -> 五個word

Is

A

Good

Day.

1. Insert(root, word)
2. Delete(root, word)
3. Search(root, word)
4. Search(root, vector<string>)

**(二) 理論分析**

1. Binary Search Tree理論

用比較字串來比對每一個元素的順序，建立BST。做三個操作之前都要先檢查是不是空BST，如果不是才能進行操作。首先是Insert，從根往下比對，找到對應的leaf後插入；接著是Delete，從根往下比對找到要刪除的元素，判斷該元素有幾個子樹，沒有子數就直接刪除，只有一個就直接替換，有兩個的話要改用successor(刪除元素右子樹中最小值)做替換；Seach，從根一路比對下去，直到leaf前找到該元素。

2. Splay Tree理論

在Binary Search Tree的基礎上，加上pointer指向某節點的parent。如果節點x沒有祖先節點，那麼就看x是parent的哪一邊子樹，就做相反邊的rotation；如果節點x有祖先，又分為全部同側跟parent節點異側，但是只要觀察需要做rotation的節點，根據左子樹或右子樹，做出相反側的rotation就可以了。Splay tree與Binary Search Tree沒有什麼不同，只是在insert, delete, search之後要對最終的節點x做splay。

3. Treap理論

在Binary Search Tree的基礎上，加上一個整數priority當成節點的heap key。一樣使用左右rotation，整個樹要保持min heap for heap key，binary search tree for tree key的結構，插入的節點會是隨機指派一個priority。首先insert先用tree key找到應該插入的位置，產生一個新node，插入進去後，若節點有左或右子樹且priority大於節點，就需要做右或左rotation；delete，找到欲刪除的節點，分為三種狀況：若是leaf則直接刪除；若是有一個子樹，直接取代節點；若是有兩個子樹，把priority高的子樹做相反側的rotation到節點，此時要刪除的節點會在另一個子樹內，進去做刪除該節點。

**(三) 演算法則**

包含Input file的讀取，僅寫出BST、SPT、Treap的Construct、Insert跟Search功能，但實際程式還有Delete、Inorder traversal功能。

1. 第一個演算法(BST)

一張含有 文字, 螢幕擷取畫面, 字型, 圖形 的圖片

自動產生的描述

struct SPT:

element <- ""

leftChild <- NULL

rightChild <- NULL

parent <- ""

end struct

func getArticle(fileName, choose): element is a string vector

if !file.open then

print("Fail file")

end if

element = []

if choose = 1 then

element <- word by word separated by blank

else

element <- word by word separated by tab key

return element

func insertNode(root, element):

if node existed then

print("The node existed")

return

end if

node\* current <- root, \*previous <- NULL

while current do

previous <- current

if element > current.element then

current <- current.rightChild

else if element < current.element then

current <- current.leftChild

end if

end while

if not previous then

insert the node with element to previous

else

if element > previous.element then

insert the node with element to previous.right

else

insert the node with element to previous.left

end if

end if

func searchNode(root, element):

while root:

if element = root.element then

print("Found " + element)

return root

else if element < root.element then

root <- root.leftChild

else

root <- root.rightChild

end if

end while

print("Not found " + element)

1. 演算法時間複雜度(time complexity): O(*log(n)*)

Because the search time complexity is dependent on the height of the BST, and the worst case of it is log(n) for a bad case.

1. 演算法空間複雜度(space complexity): O(*n*)

Since BST using linked list data structure to store the input file, so the space complexity is dependent on the number of inputs which represented as n.

2. 第二個演算法(SPT)

一張含有 文字, 螢幕擷取畫面, 字型, 圓形 的圖片

自動產生的描述

struct SPT:

element <- ""

leftChild <- NULL

rightChild <- NULL

parent <- ""

end struct

func getArticle(fileName, choose): element is a string vector

if !file.open then

print("Fail file")

end if

element = []

if choose = 1 then

element <- word by word separated by blank

else

element <- word by word separated by tab key

return element

func insertNode(root, element):

if node existed then

print("The node existed")

return

end if

node\* current <- root, \*previous <- NULL

while current do

previous <- current

if element > current.element then

current <- current.rightChild

else if element < current.element then

current <- current.leftChild

end if

end while

if not previous then

insert the node with element to previous

else

if element > previous.element then

insert the node with element to previous.right

else

insert the node with element to previous.left

end if

splay the insert node x

end if

func searchNode(root, element):

while root:

if element = root.element then

print("Found " + element)

splay(current)

return root

else if element < root.element then

root <- root.leftChild

else

root <- root.rightChild

end if

end while

print("Not found " + element)

func left\_rotation(x):

node\* y <- x.rightChild

x.rightChild <- y.leftChild

if y.leftChild then

y.leftChild.parent <- x

end if

y.parent = x.parent

if not x.parent then

root <- y

else if x = x.parent.leftChild then

x.parent.leftChild <- y

else

x.parent.leftChild <- y

end if

y.leftChild <- x

x.parent <- y

func right\_rotation(x):

node\* y <- x.leftChild

if y.rightChild then

y.rightChild.parent <- x

end if

y.parent <- x.parent

if not x.parent then

root <- y

else if x = x.parent.rightChild

x.parent.rightChild <- y

else

x.parent.leftChild <- y

end if

y.rightChild <- x

x.parent <- y

func splay(x)

while x.parent do

if not x.parent.parent then

if x = x.parent.leftChild then

right\_rotation(x.parent)

else

left\_rotation(x.parent)

end if

else if x = x.parent.leftChild and x.parent = x.parent.parent.leftChild then

right\_rotation(x.parent.parent)

right\_rotation(x.parent)

else if x = x.parent.rightChild and x.parent = x.parent.parent.rightChild then

left\_rotation(x.parent.parent)

left\_rotation(x.parent)

else if x = x.parent.rightChuld and x.parent = x.parent.parent.leftChild then

left\_rotation(x.parent)

right\_rotation(x.parent)

else

right\_rotation(x.parent)

left\_rotation(x.parent)

end if

end while

1. 演算法時間複雜度(time complexity): O(*log(n)*)

Because the search time complexity is dependent on the height of the SPT, but it would splay after each operation, which makes it is O(log(n)) generally.

1. 演算法空間複雜度(space complexity): O(*n*)

Since SPT using linked list data structure to store the input file, so the space complexity is dependent on the number of inputs which represented as n.

3. 第三個演算法(Algorithm)

一張含有 文字, 螢幕擷取畫面, 字型, 圓形 的圖片

自動產生的描述

struct Treap:

element <- ""

priority <- 0

leftChild <- NULL

rightChild <- NULL

Treap(element, priority): element(element), priority(priority), leftChild(NULL), rightChild(NULL)

end struct

func getArticle(fileName, choose): element is a string vector

if !file.open then

print("Fail file")

end if

element = []

if choose = 1 then

element <- word by word separated by blank

else

element <- word by word separated by tab key

return element

func insertNode(root, element):

insert(root, element)

func insert(root, element):

if not root then

root = new Treap(element, random integer)

end if

else

if element = root.element then

print("The element existed)

return

else if element > root.element then

insert(root.rightChild, element)

else

insert(root.leftChild, element)

end if

if root.leftChild and root.leftChild.priority > root.priority then

right\_rotation(root)

else if root.rightChild and root.rightChild.priority < root.priority then

left\_rotation(root)

end if

end if

func searchNode(root, element):

while root:

if element = root.element then

print("Found " + element)

splay(current)

return root

else if element < root.element then

root <- root.leftChild

else

root <- root.rightChild

end if

end while

print("Not found " + element)

func left\_rotation(x):

node\* y <- x.rightChild

x.rightChild <- y.leftChild

y.leftChild <- x

x <- y

func right\_rotation(x):

node\* y <- x.leftChild

x.leftChild <- y.rightChild

y.rightChild <- x

x <- y

1. 演算法時間複雜度(time complexity): O(*log(n)*)

Because the search time complexity is dependent on the height of the Treap, generally, it is O(log(n)).

1. 演算法空間複雜度(space complexity): O(*n*)

Since Treap using linked list data structure to store the input file, so the space complexity is dependent on the number of inputs which represented as n.

**(四) 程式設計環境架構**

## **1. 程式語言**

C++ in Microsoft Windows10 64-bits operating system

## **2. 程式開發工具**

Microsoft Visual Studio 2019

## **3. 電腦硬體**

CPU: 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz

Main Memory: 16.0 GB

顯示卡1: Intel® UHD Graphics

顯示卡2: NVIDIA GeForce RTX 3050 Ti Laptop GPU

## **4. 作業系統**

Windows 10 家用版

版本: 22H2

**(五) 程式**

1. 主程式

1. **BST**

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

struct node {

string element;

struct node\* leftChild;

struct node\* rightChild;

};

vector<string> getArticle();

string get\_test\_word();

vector<string> get\_test\_file();

node\* build\_root(string);

node\* check\_exist\_node(node\*, string);

node\* insertNode(node\*, string);

node\* deleteNode(node\*, string);

node\* searchNode(node\*, string);

void inorder(node\*);

int main(void) {

node\* root = NULL;

vector<string> article = getArticle();

vector<string> test\_words;

string test\_word;

double start, end;

int i, choose = 4;

start = clock();

root = insertNode(root, article[0]);

for (i = 1; i < article.size(); i++) {

insertNode(root, article[i]);

}

cout << "\n";

cout << "Enter a number to choose what to do(with a file) [1] Insert a word, [2] Delete a word, [3] Search a word, [4] Search part of the input file: ";

cin >> choose;

cout << "\n";

switch (choose) {

case 1:

test\_word = get\_test\_word();

insertNode(root, test\_word);

break;

case 2:

test\_word = get\_test\_word();

deleteNode(root, test\_word);

break;

case 3:

test\_word = get\_test\_word();

searchNode(root, test\_word);

break;

case 4:

test\_words = get\_test\_file();

for (i = 0; i < test\_words.size(); i++) {

searchNode(root, test\_words[i]);

}

break;

default:

break;

}

/\*cout << "The final tree is shown with inorder traversal:";

inorder(root);

cout << ".\n";\*/

end = clock();

cout << "The execution time is: " << end - start << "ms.\n";

return 0;

}

vector<string> getArticle() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the article file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

while (true) {

element.push\_back(line.substr(0, line.find(" ")));

line = line.substr(line.find(" ") + 1, line.length());

if (line.find(" ") == -1) {

element.push\_back(line);

break;

}

}

}

return element;

}

string get\_test\_word() {

ifstream in\_file;

string file\_name, element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

getline(in\_file, element);

return element;

}

vector<string> get\_test\_file() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

element.push\_back(line);

}

return element;

}

node\* build\_root(string element) {

node\* root = new node;

root->element = element;

root->leftChild = NULL;

root->rightChild = NULL;

return root;

}

node\* check\_exist\_node(node\* root, string element) {

while (root) {

if (root->element == element) {

return root;

}

else if (root->element > element) {

root = root->leftChild;

}

else {

root = root->rightChild;

}

}

return NULL;

}

node\* insertNode(node\* root, string element) {

if (check\_exist\_node(root, element)) {

cout << "\"" << element << "\"" << " already exists\n";

return root;

}

node\* current = root;

node\* previous = NULL;

while (current) {

previous = current;

if (element > current->element) {

current = current->rightChild;

}

else if (element < current->element) {

current = current->leftChild;

}

}

if (!previous) {

previous = build\_root(element);

}

else {

if (element > previous->element) {

previous->rightChild = build\_root(element);

}

else {

previous->leftChild = build\_root(element);

}

}

cout << "Inserted " << element << " to BST.\n";

return previous;

}

node\* deleteNode(node\* root, string element) {

if (!root) {

cout << "The BST is blank such that don't able to delete\n";

return root;

}

node\* current = root;

node\* previous = NULL;

while (current && current->element != element) {

previous = current;

if (element > current->element) {

current = current->rightChild;

}

else if (element < current->element) {

current = current->leftChild;

}

}

if (!current) {

cout << element << " is not exist in the article\n";

return root;

}

if (!current->leftChild || !current->rightChild) {

node\* replace;

if (!current->leftChild) {

replace = current->rightChild;

}

else {

replace = current->leftChild;

}

if (!previous) {

return replace;

}

if (current == previous->leftChild) {

previous->leftChild = replace;

}

else {

previous->rightChild = replace;

}

cout << "Deleted " << element << " from BST\n";

free(current);

}

else {

node\* checker = NULL;

node\* successor = current->rightChild;

while (successor->leftChild) {

checker = successor;

successor = successor->leftChild;

}

if (checker) {

checker->leftChild = successor->rightChild;

}

else {

checker->rightChild = successor->rightChild;

}

cout << "Deleted " << element << " from BST\n";

current->element = successor->element;

free(successor);

}

return root;

}

node\* searchNode(node\* root, string element) {

while (root) {

if (element == root->element) {

cout << "Found " << element << " \n";

return root;

}

else if (element < root->element) {

root = root->leftChild;

}

else {

root = root->rightChild;

}

}

cout << "Not found " << element << " \n";

return NULL;

}

void inorder(node\* root) {

if (root) {

inorder(root->leftChild);

cout << " " << root->element << " ";

inorder(root->rightChild);

}

return;

}

1. **SPT**

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <ctime>

using namespace std;

struct node {

string element;

struct node\* leftChild;

struct node\* rightChild;

struct node\* parent;

};

class splay\_tree {

private:

node\* root;

public:

splay\_tree() {}

void insertNode(string);

void deleteNode(string);

void searchNode(string);

void inorder();

void inorder\_helper(node\*);

void left\_rotation(node\*);

void right\_rotation(node\*);

void splay(node\*);

};

vector<string> getArticle();

string get\_test\_word();

vector<string> get\_test\_file();

int main(void) {

splay\_tree splayT;

node\* root = NULL;

vector<string> article = getArticle();

vector<string> test\_words;

string test\_word;

double start, end;

int i, choose = 4;

start = clock();

for (i = 0; i < article.size(); i++) {

splayT.insertNode(article[i]);

}

cout << "\n";

cout << "Enter a number to choose what to do(with a file) [1] Insert a word, [2] Delete a word, [3] Search a word, [4] Search part of the input file: ";

cin >> choose;

cout << "\n";

switch (choose) {

case 1:

test\_word = get\_test\_word();

splayT.insertNode(test\_word);

break;

case 2:

test\_word = get\_test\_word();

splayT.deleteNode(test\_word);

break;

case 3:

test\_word = get\_test\_word();

splayT.searchNode(test\_word);

break;

case 4:

test\_words = get\_test\_file();

for (i = 0; i < test\_words.size(); i++) {

splayT.searchNode(test\_words[i]);

}

break;

default:

break;

}

/\*cout << "The final tree is shown with inorder traversal:";

splayT.inorder();

cout << "\n";\*/

end = clock();

cout << "The execution time is: " << end - start << "ms.\n";

return 0;

}

vector<string> getArticle() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the article file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

while (true) {

element.push\_back(line.substr(0, line.find(" ")));

line = line.substr(line.find(" ") + 1, line.length());

if (line.find(" ") == -1) {

element.push\_back(line);

break;

}

}

}

return element;

}

string get\_test\_word() {

ifstream in\_file;

string file\_name, element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

getline(in\_file, element);

return element;

}

vector<string> get\_test\_file() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

element.push\_back(line);

}

return element;

}

void splay\_tree::insertNode(string element) {

node\* insert = new node;

insert->parent = NULL;

insert->leftChild = NULL;

insert->rightChild = NULL;

insert->element = element;

node\* current = root;

node\* previous = NULL;

while (current) {

previous = current;

if (insert->element == current->element) {

cout << element << " already exists\n";

return;

}

else if (insert->element < current->element) {

current = current->leftChild;

}

else {

current = current->rightChild;

}

}

insert->parent = previous;

if (!previous) {

root = insert;

}

else if (insert->element < previous->element) {

previous->leftChild = insert;

}

else {

previous->rightChild = insert;

}

cout << element << " is inserted\n";

splay(insert);

}

void splay\_tree::deleteNode(string element) {

node\* first = root;

node\* second = NULL;

node\* third = NULL;

while (root) {

if (root->element <= element) {

first = first->rightChild;

}

else {

first = first->leftChild;

}

}

if (!first) {

cout << element << "not found to delete\n";

}

splay(first); //split the tree

if (first->rightChild) {

third = first->rightChild;

third->parent = NULL;

}

else {

third->parent = NULL;

}

second = first;

second->rightChild = NULL;

first = NULL;

if (third->leftChild) { //remove element

cout << element << " is deleted\n";

third->leftChild->parent = NULL;

}

if (!third->leftChild) { //join the tree

root = second;

}

else if (!second) {

root = third->leftChild;

}

node\* predecessor = third->leftChild;

while (!predecessor->rightChild) {

predecessor = predecessor->rightChild;

}

splay(predecessor);

first->rightChild = third;

third->parent = first;

root = predecessor;

delete(second);

second = NULL;

}

void splay\_tree::searchNode(string element) {

node\* current = root;

while (current) {

if (element == current->element) {

cout << "Found " << element << " \n";

splay(current);

return;

}

else if (element < current->element) {

current = current->leftChild;

}

else {

current = current->rightChild;

}

}

cout << "Not found " << element << " \n";

}

void splay\_tree::inorder() {

inorder\_helper(root);

}

void splay\_tree::inorder\_helper(node\* root) {

if (root) {

inorder\_helper(root->leftChild);

cout << " " << root->element << " ";

inorder\_helper(root->rightChild);

}

return;

}

void splay\_tree::left\_rotation(node\* x) {

node\* y = x->rightChild;

x->rightChild = y->leftChild;

if (y->leftChild) {

y->leftChild->parent = x;

}

y->parent = x->parent;

if (!x->parent) {

root = y;

}

else if (x == x->parent->leftChild) {

x->parent->leftChild = y;

}

else {

x->parent->rightChild = y;

}

y->leftChild = x;

x->parent = y;

}

void splay\_tree::right\_rotation(node\* x) {

node\* y = x->leftChild;

x->leftChild = y->rightChild;

if (y->rightChild) {

y->rightChild->parent = x;

}

y->parent = x->parent;

if (!x->parent) {

root = y;

}

else if (x == x->parent->rightChild) {

x->parent->rightChild = y;

}

else {

x->parent->leftChild = y;

}

y->rightChild = x;

x->parent = y;

}

void splay\_tree::splay(node\* x) {

while (x->parent) {

if (!x->parent->parent) { //only two node

if (x == x->parent->leftChild) { //zig

right\_rotation(x->parent);

}

else { //zag

left\_rotation(x->parent);

}

}

else if (x == x->parent->leftChild && x->parent == x->parent->parent->leftChild) {

// zig-zig

right\_rotation(x->parent->parent);

right\_rotation(x->parent);

}

else if (x == x->parent->rightChild && x->parent == x->parent->parent->rightChild) {

// zag-zag

left\_rotation(x->parent->parent);

left\_rotation(x->parent);

}

else if (x == x->parent->rightChild && x->parent == x->parent->parent->leftChild) {

// zig-zag

left\_rotation(x->parent);

right\_rotation(x->parent);

}

else {

// zag-zig

right\_rotation(x->parent);

left\_rotation(x->parent);

}

}

}

1. **Treap**

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <random>

#include <ctime>

using namespace std;

#define MAX\_RANDOM 999999

struct node {

string element;

int priority;

struct node\* leftChild;

struct node\* rightChild;

node(string element, int rand) : element(element), leftChild(NULL), rightChild(NULL), priority(rand) {}

};

class treap {

private:

node\* root;

void left\_rotation(node\*&);

void right\_rotation(node\*&);

void insert(node\*&, string);

void remove(node\*&, string);

public:

treap() {}

void insertNode(string);

void deleteNode(string);

void searchNode(string);

void inorder();

void inorder\_helper(node\*);

};

vector<string> getArticle();

string get\_test\_word();

vector<string> get\_test\_file();

int main(void) {

treap Treap;

vector<string> article = getArticle();

vector<string> test\_words;

string test\_word;

double start, end;

int i, choose = 4;

start = clock();

for (i = 0; i < article.size(); i++) {

Treap.insertNode(article[i]);

}

cout << "\n";

cout << "Enter a number to choose what to do(with a file) [1] Insert a word, [2] Delete a word, [3] Search a word, [4] Search part of the input file: ";

cin >> choose;

cout << "\n";

switch (choose) {

case 1:

test\_word = get\_test\_word();

Treap.insertNode(test\_word);

break;

case 2:

test\_word = get\_test\_word();

Treap.deleteNode(test\_word);

break;

case 3:

test\_word = get\_test\_word();

Treap.searchNode(test\_word);

break;

case 4:

test\_words = get\_test\_file();

for (i = 0; i < test\_words.size(); i++) {

Treap.searchNode(test\_words[i]);

}

break;

default:

break;

}

/\*cout << "The final tree is shown with inorder traversal:";

Treap.inorder();

cout << "\n";\*/

end = clock();

cout << "The execution time is: " << end - start << "ms.\n";

return 0;

}

vector<string> getArticle() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the article file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

while (true) {

element.push\_back(line.substr(0, line.find(" ")));

line = line.substr(line.find(" ") + 1, line.length());

if (line.find(" ") == -1) {

element.push\_back(line);

break;

}

}

}

return element;

}

string get\_test\_word() {

ifstream in\_file;

string file\_name, element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

getline(in\_file, element);

return element;

}

vector<string> get\_test\_file() {

ifstream in\_file;

string file\_name, line;

vector<string> element;

cout << "Enter the test file name in (.txt) format: ";

cin >> file\_name;

in\_file.open(file\_name);

if (!in\_file.is\_open()) {

cout << "Failed to open the file.\n";

exit(1);

}

while (getline(in\_file, line)) {

element.push\_back(line);

}

return element;

}

void treap::insertNode(string element) {

insert(root, element);

}

void treap::deleteNode(string element) {

remove(root, element);

}

void treap::searchNode(string element) {

node\* current = root;

while (current) {

if (element == current->element) {

cout << "Found " << element << " \n";

return;

}

else if (element < current->element) {

current = current->leftChild;

}

else {

current = current->rightChild;

}

}

cout << "Not found " << element << " \n";

}

void treap::inorder() {

inorder\_helper(root);

}

void treap::inorder\_helper(node\* root) {

if (root) {

inorder\_helper(root->leftChild);

cout << " " << root->element << " ";

inorder\_helper(root->rightChild);

}

return;

}

void treap::left\_rotation(node\*& root) {

node\* x = root->rightChild;

root->rightChild = x->leftChild;

x->leftChild = root;

root = x;

}

void treap::right\_rotation(node\*& root) {

node\* x = root->leftChild;

root->leftChild = x->rightChild;

x->rightChild = root;

root = x;

}

void treap::insert(node\*& root, string element) {

random\_device rd;

uniform\_int\_distribution<int> dist(1, MAX\_RANDOM);

if (!root) {

root = new node(element, dist(rd));

}

else {

if (element == root->element) {

cout << element << " already exists\n";

return;

}

else if (element > root->element) {

insert(root->rightChild, element);

}

else {

insert(root->leftChild, element);

}

if (root->leftChild && root->leftChild->priority > root->priority) {

right\_rotation(root);

}

else if (root->rightChild && root->rightChild->priority < root->priority) {

left\_rotation(root);

}

}

}

void treap::remove(node\*& root, string element) {

if (root->element == element) {

cout << element << " is deleted\n";

if (!root->leftChild) {

root = root->rightChild;

}

else if (!root->rightChild) {

root = root->leftChild;

}

else {

if (root->leftChild->priority > root->rightChild->priority) {

right\_rotation(root);

remove(root->rightChild, element);

}

else if (root->leftChild->priority < root->rightChild->priority) {

left\_rotation(root);

remove(root->leftChild, element);

}

}

}

else {

if (element < root->element) {

remove(root->leftChild, element);

}

else if (element > root->element) {

remove(root->rightChild, element);

}

}

}

2. Input Code Format

* Please don’t destory the file format I delivered, or you might not able to get the same result as mine.

**(1) Test article**

1. TestFile1.txt excerpted from https://www.channelnewsasia.com/news/sport/badminton-taiwan-s-tai-beats-marin-to-win-singaporeopen-8727786 (not able to available now)

SINGAPORE: Tai Tzu Ying cemented her status as the world's top female shuttler as she beat reigning Olympic champion and world number two Carolina Marin in straight games at the Singapore Open on Sunday (Apr 16). World number one Tai took just 38 minutes to seal a clinical 21-15, 21-15 victory at the Singapore Indoor Stadium.

1. TestFile2.txt excerpted from the lyrics of Michael Bublé - It's A Beautiful Day

I don't know why. You think that you could hold me. When you couldn't get by by yourself. And I don't know who. Would ever want to tear the seam of someone's dream. Baby, it's fine, you said that we should just be friends. While I came up with that line and I'm sure. That it's for the best. If you ever change your mind, don't hold your breath. 'Cause you may not believe. That baby, I'm relieved, hmm. When you said goodbye, my whole world shines. Hey hey hey. It's a beautiful day and I can't stop myself from smiling. If we're drinking, then I'm buying. And I know there's no denying. It's a beautiful day, the sun is up, the music's playing. And even if it started raining. You won't hear this boy complaining. 'Cause I'm glad that you're the one who got away. It's a beautiful day. It's my turn to fly, so girls, get in line. 'Cause I'm easy, no playing this guy like a fool. Now I'm alright. Might've had me caged before, but not tonight. And you may not believe, hmm. That baby, I'm relieved. This fire inside, it burns too bright. I don't want to say "So long", I just want to say "Goodby". It's a beautiful day and I can't stop myself from smiling. If we're drinking, then I'm buying. And I know there's no denying. That it's a beautiful day, the sun is up, the music's playing. And even if it started raining. You won't hear this boy complaining. 'Cause I'm glad that you're the one who got away, hmm. 'Cause if you ever think I'll take up. My time with thinking of our break-up. Then, you've got another thing coming your way. 'Cause it's a beautiful day. Beautiful day. Oh, baby, any day that you're gone away. It's a beautiful day.

**(2) Test(Search) with files**

1. For TestFile1.txt
2. TestFile11.txt

It was Tai's fifth consecutive Super Series title and her second victory over Marin in less than two weeks, having

already triumphed at the Malaysian Open final.

1. TestFile12.txt

game game game game game game game game game game game game game game world world world world world

world world world world world world world world world world world world world world world world

1. TestFile13.txt

Saina Nehwal lost in straight games to world no. 1 Tai Tzu Ying in the French Open 2018 women's singles quarterfinals on Friday. Saina Nehwal, who has been in fine form for the last couple of weeks, was looking to avenge her

Denmark Open final loss against Tai Tzu Ying. The Chinese Taipei now has an overwhelming 14-5 head-to-head

record against the Indian. In the men's singles quarter-finals match, Kidambi Srikanth lost to world no.1 Kento

Momota in a closely-contested match. In a match that lasted 52 minutes, Srikanth lost 16-21, 19-21.

1. For TestFile2.txt
2. TestFile21.txt

I don't know why

You think that you could hold me

When you couldn't get by by yourself

And I don't know who

Would ever want to tear the seam of someone's dream

1. TestFile22.txt

don'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon'tdon't

think think think think think think think think think think think think think think think think think think think think

1. TestFile23.txt

I

don't

know

why

You think that you could hold me

When

you

couldn't

get

by

by

yourself

And I don't know who

Would

ever

want

to

tear

the

seam

of

someone's

dream

3. Output Code Format

Search TestFile1.txt with TestFile11.txt, TestFile12.txt, TestFile13.txt in Dictionary\_Prog1\_S11059003.cpp by using selecting the function ‘4’.

1. **Constructure result(same for each search)**

Inserted SINGAPORE: to BST.

Inserted Tai to BST.

Inserted Tzu to BST.

Inserted Ying to BST.

Inserted cemented to BST.

Inserted her to BST.

Inserted status to BST.

Inserted as to BST.

Inserted the to BST.

Inserted world's to BST.

Inserted top to BST.

Inserted female to BST.

Inserted shuttler to BST.

"as" already exists

Inserted she to BST.

Inserted beat to BST.

Inserted reigning to BST.

Inserted Olympic to BST.

Inserted champion to BST.

Inserted and to BST.

Inserted world to BST.

Inserted number to BST.

Inserted two to BST.

Inserted Carolina to BST.

Inserted Marin to BST.

Inserted in to BST.

Inserted straight to BST.

Inserted games to BST.

Inserted at to BST.

"the" already exists

Inserted Singapore to BST.

Inserted Open to BST.

Inserted on to BST.

Inserted Sunday to BST.

Inserted (Apr to BST.

Inserted 16). to BST.

Inserted World to BST.

"number" already exists

Inserted one to BST.

"Tai" already exists

Inserted took to BST.

Inserted just to BST.

Inserted 38 to BST.

Inserted minutes to BST.

Inserted to to BST.

Inserted seal to BST.

Inserted a to BST.

Inserted clinical to BST.

Inserted 21-15, to BST.

Inserted 21-15 to BST.

Inserted victory to BST.

"at" already exists

"the" already exists

"Singapore" already exists

Inserted Indoor to BST.

Inserted Stadium. to BST.

1. **TestFile11.txt**

Not found It was Tai's fifth consecutive Super Series title and her second victory over Marin in less than two weeks, having

Not found already triumphed at the Malaysian Open final.

The execution time is: 23ms.

1. **TestFile12.txt**

Not found game game game game game game game game game game game game game game world world world world world

Not found world world world world world world world world world world world world world world world world

The execution time is: 17ms.

1. **TestFile13.txt**

Not found Saina Nehwal lost in straight games to world no. 1 Tai Tzu Ying in the French Open 2018 women's singles quarterfinals on Friday. Saina Nehwal, who has been in fine form for the last couple of weeks, was looking to avenge her

Not found Denmark Open final loss against Tai Tzu Ying. The Chinese Taipei now has an overwhelming 14-5 head-to-head

Not found record against the Indian. In the men's singles quarter-finals match, Kidambi Srikanth lost to world no.1 Kento

Not found Momota in a closely-contested match. In a match that lasted 52 minutes, Srikanth lost 16-21, 19-21.

The execution time is: 44ms.

**(六) 執行結果、討論與心得**

1. 執行結果

* You may check the result screenshots in my Github. The format example is shown below: 11.png, it means using TestFile11.txt to test TestFile1.txt; or 23.png, it means using TestFile23.txt to test TestFile2.txt. Each of them are stored in a folder, the folder name is the using data structure. Each data structure(folder) would have 6 result.
* Before you try to reappear my screenshot result, you must change some function for avoiding the input time from keying some inputs from keyboard. This operation to the three programs is the same, no need to change case by case.
* Main function

Make sure “choose” variable in main function is equal to 4, and comment like following:

choose = 4;

//cout << "Enter a number to choose what to do(with a file) [1] Insert a word, [2] Delete a word, [3] Search a word, [4] Search part of the input file: ";

//cin >> choose;

* getArticle function

Enter the article file name into in\_file.open()

//cout << "Enter the article file name in (.txt) format: ";

//cin >> file\_name;

in\_file.open("TestFile1.txt");

* get\_test\_file function

Enter the test file name for the article file into in\_file.open()

//cout << "Enter the article file name in (.txt) format: ";

//cin >> file\_name;

in\_file.open("TestFile11.txt");

1. **BST**

[NUTN\_Algoritm/HW3\_Dictionary/BST at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW3_Dictionary/BST)

1. **SPT**

[NUTN\_Algoritm/HW3\_Dictionary/SPT at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW3_Dictionary/SPT)

1. **Treap**

[NUTN\_Algoritm/HW3\_Dictionary/Treap at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW3_Dictionary/Treap)

2. 討論

1. Search operation with files execute time(by execution result)
2. BST

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

1. SPT

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

1. Treap

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

3. 心得

一開始我在課堂上並不瞭解Splay tree的作用，只是覺得每做一個動作就要執行splay相當麻煩，但是經由這次的實作，我了解到在麻煩背後的意義是為了提前準備下次的使用，splay的目的是將所得到的結果(Insert, Delete, Search基本上有異曲同工之妙)，往根部的方向上拉，這樣在下次如果有重複性的操作，就可以得到更少的執行時間(更快的執行效率)，不過從結果上來看，Splay tree可能尚未被我執行到可以真正跟Binary Search Tree做出區別的input size，理論上來說，Splay tree在處理具有重複性的工作時，將會具有優勢；而Treap則是以效率作為考量做設計，老師曾說過在Google的入職試中，很多人不知道Treap怎麼寫，然而，我在課堂上也只是覺得多了一個莫名其妙的隨機priority key，還要維持對子樹是min/max heap的treap有什麼意義。正因為這種特性(限制)，才使得treap具有唯一性，不會因為輸入的順序調換而導致產生的treap長得不一樣，它的效率也因此被提升，符合我所測試出的結果，在log(n)的時間複雜度中，顯然是treap相對於其他兩者更勝一籌，不論input size(當然也有可能是我沒有測大size)，總而言之，這是一個非常令人印象深刻的學習。

**參考文獻**

1. [Ordering - 演算法筆記 (ntnu.edu.tw)](https://web.ntnu.edu.tw/~algo/Ordering2.html)
2. [伸展树（二） - C++实现-CSDN博客](https://blog.csdn.net/u012294613/article/details/125636296)
3. [C++ Program to Implement Splay Tree (tutorialspoint.com)](https://www.tutorialspoint.com/cplusplus-program-to-implement-splay-tree)
4. [Splay Trees : Splaying, Insertion and Deletion (codesdope.com)](https://www.codesdope.com/course/data-structures-splay-trees/)
5. [4.2 Splay Trees Insertion | with Example | Advanced Data Structures (youtube.com)](https://www.youtube.com/watch?v=7O6nW1Mcz-c&ab_channel=UzairJavedAkhtar)
6. [資料結構與演算法：Splay Tree 伸展樹 - Joseph's blog (josephjsf2.github.io)](https://josephjsf2.github.io/data/structure/and/algorithm/2020/05/03/splay-tree.html)
7. [Splay Trees (with implementations in C++, Java, and Python) | Algorithm Tutor](https://algorithmtutor.com/Data-Structures/Tree/Splay-Trees/)
8. [Treap - HackMD](https://hackmd.io/@konchin/ryRbPBxso?utm_source=preview-mode&utm_medium=rec#Merge)
9. [How to convert int to string in C++? - Stack Overflow](https://stackoverflow.com/questions/5590381/how-to-convert-int-to-string-in-c)
10. [C++ String Functions: 10 Definitions + Examples (hubspot.com)](https://blog.hubspot.com/website/c-string-functions)
11. [Treap (A Randomized Binary Search Tree) - GeeksforGeeks](https://www.geeksforgeeks.org/treap-a-randomized-binary-search-tree/)