THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - INDIVIDUAL ASSESSMENT

Unit of Study: INFO1905

Assignment name: Assignment 1

Tutorial time: Friday 8am

Tutor name: Simon Koch

DECLARATION

I declare that I have read and understood the *University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy*, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the the *Academic Dishonesty and Plagiarism in Coursework Policy*, can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Student ID: 460430331

Student name: Nahian-Al Hasan

Signed _____ **Date** 22/09/2017

SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

INFO1905 Assignment

Nahian-Al Hasan, SID: 460430331
Tutor: Simon Koch, Lab: Friday 8:00 AM

September 22, 2017

Overview

The number of Data Structures used are quite a few and are slightly complicated. The Assignment class contains one `HashMap<String, MyTreeMap>` and one `TreeMap<Integer, TreeSet<String>>`. The HashMap contains my MyTreeMap object, which further contains a `TreeMap, TreeSet and HashMap`. This combined structure handles all the `Date` operations using `TreeMap` functions, while the inner HashMap and the TreeSet handle best grade and duplicates.
On the other hand, the `TreeMap<Integer, TreeSet<String>>` structure is used solely for the purpose of `listTopStudents()`.
Upon my analysis, all the functions perform better than or according to the bare minimum requirements of the assignment specifications. I have discussed this more in detail in the following sections.

Data Structures

1. `TreeMap<Integer, TreeSet<String>> bestGrades` - this dynamically updates the best grades of every student after every insertion and removal.

2. `HashMap<String , MyTreeMap> bigMap` - contains my class called `MyTreeMap`, which further contains the following objects and Data Structures.

   (a) `TreeMap<Date, MyEntry> map` - contains `MyEntry`, which is my implementation of the `Submission` interface and is sorted by `Date`. This `TreeMap` is useful for getting `Submissions` based on `Date`.

   (b) `TreeSet<Integer> grades` - contains the sorted grades for the student, and is useful for returning the best grade of every students.

   (c) `HashMap<Integer, Integer> gradeCount` - keeps track of the count of every grade to account for duplicates in the `TreeSet<Integer> grades`.

Complexity Analysis

- `public Integer getBestGrade(String unikey)` calls `HashMap.containsKey(...)` and `HashMap.get(...).getBestGrade()` , which further calls `TreeSet.first()` , which in total gives worst case of $O(1) + O(1) + O(logn) = O(logn)$

- `public Submission getSubmissionFinal(String unikey)` calls `HashMap.containsKey(...)` and `HashMap.get(...).getMostRecentSubmission()` which then calls `TreeMap.size()` and `TreeMap.lastEntry().getValue()` which gives a worst case of $O(1) + O(1) + O(1) + O(logn) = O(logn)$

- `public Submission getSubmissionBefore(String unikey, Date deadline)` calls `HashMap.containsKey(...)` and `HashMap.get(...).getSubmissionPriorToTime(deadline)` which again calls `TreeMap.floorEntry())` and `Map.Entry.getValue()` giving a worst case of $O(1) + O(1) + O(logn) + O(1) = O(logn)$

- `public Submission add(String unikey, Date timestamp, Integer grade)` calls `HashMap.containsKey(...)` once and `HashMap.get(...)` once and `MyTreeMap.getBestGrade()`, and then `MyTreeMap.addSubmission(...)`, which then calls `HashMap.containsKey(...)` and `HashMap.put(..., HashMap.get(...))` and then `TreeSet.add(...)` and `TreeMap.put(...)` and finally `TreeMap.containsKey(...)` once and `TreeMap.put(...).TreeSet.add(...)` once giving a worst case of $O(1) + O(1) + O(logn) + O(1) + O(1) + O(1) + O(logn) + O(logn) + O(logn) + O(logn) + O(logn) = O(logn)$

- `public void remove(Submission submission)` calls `HashMap.size()` once, `HashMap.get(...)` once, `MyTreeMap.getBestGrade()` once and then `MyTreeMap.removeSubmission(...)` once, which then calls `TreeMap.containsKey(...)` once `HashMap.put(...)` and `HashMap.get(...)` once, `MyTreeMap.getBestGrade()` once and `HashMap.get(...)` once, `TreeSet.remove(...)` once, `TreeMap.remove(...)` once and then finally calls `TreeMap.get(...).TreeSet.remove(...)` once to give a worst case of $O(1) + O(1) + O(logn) + O(logn) + O(1) + O(1) + O(logn) + O(1) + O(logn) + O(logn) + O(logn) + O(logn) = O(logn)$

- `public List<String> listTopStudents()` calls `TreeMap.lastEntry()` and `TreeMap.lestEntry().getValue()` which is then converted to an `ArrayList<String>` giving a worst case of $O(logn) + O(logn) + O(logn) = O(logn)$

- `public List <String> listRegressions()` *iterates* through every student in `HashMap <String, MyTreeMap> bigMap`, and finds their best grades using `MyTreeMap.getBestGrade()` and then get their most recent grades using `MyTreeMap.getMostRecentSubmission()`, and then comparing and adding to the ArrayList using `ArrayList<Integer>.add(...)`. This gives a worst case complexity of
$O(n) * (O(logn) + O(logn) + O(1)) = O(nlogn)$