COMP3506/7505 Project

Due: 23rd September 2022

Report Template

	Full Name	Student ID
Details	Erik Flink	S46267445

1. Overview

This document is the ***mandatory*** template which must be used to submit the report section of your project.

This template is automatically synced with Gradescope to identify the location of each section of the report; therefore, it is imperative that the overall format/layout of this document not be modified. Modification of the template **will** result in a penalty being applied.

You are permitted to make changes inside the purple boxes for each question provided however the overall size of the box cannot change. Your report should easily fit within the boxes provided however please be aware the minimum font size allowed is Arial 10pt. If you are exceeding the box size, then this may be a good indication your response is not succinct.

2. Submission

Once you have completed your report this document must be exported as a pdf and uploaded to the Gradescope Report Portal for Part B of this assignment. This document **should not** be uploaded to the autograder.

3. Marking

The report will be hand marked by the teaching team. Information regarding the rubrics used while marking will be made available after grades are released. While this report will indicate the relative weighting of each section of the report, should there be any discrepancy with the official assignment specification, the assignment specification shall take precedent.

4. Plagiarism

The University has strict policies regarding plagiarism. Penalties for engaging in unacceptable behaviour can range from cash fines or loss of grades in a course, through to expulsion from UQ. You are required to read and understand the policies on academic integrity and plagiarism in the course profile (Section 6.1).

If you have any questions regarding acceptable level of collaboration with your peers, please see either the lecturer or your tutor for guidance. Remember that ignorance is not a defence!

5. Task

You are required to complete all sections of this report in line with the programming tasks completed in the project.

Hospital Appointment System (16 Marks)

Hospital 1

1. State the data structure used to store patients and explain why this data structure is the best for the task in hand.

A sorted singly linked list was used to store patients. As the there are no duplicate time slots allowed, this required the data structure to search and then insert the patient into the hospital. By using a sorted linked list, while checking for the new patient's insertion position, if the time slot is unavailable, the new patient isn't added, otherwise they are placed in the new position.

2. Describe the algorithm used to order the patients. Briefly explain how it works, from where it is called (from iterator/__iter__ or from addPatient/add_patient) and why it is the best algorithm in comparison with the other known algorithms.

When addPatient is called, the new patient is added to its sorted position by parsing through the linked list till this position is found. When the iterator function is called, no sorting algorithm is needed as the list is already sorted so the iterator parses through the already sorted list.
This algorithm is the best for this situation as addPatient requires the list to be parsed to see if the timeslot is already occupied, so by also checking for its position in the list at the same time, time is saved sorting the list for the iterator function.

3. Assuming n to be the number of patients, state the best-case (Ω) and worst-case (O) time complexity of iterator/__iter__ and addPatient/add_patient with respect to n.

Iterator:
Iterator function and associated functions (hasnext() and next()) all take O(1) in the best and worst case time complexity as the list is already sorted.

addPatient:
The worst-case time complexity for adding a patient would be O(n) in the situation where the patient timeslot is after all the other patients scheduled appointments. The best-case time complexity would be Ω(1) in the situation where the timeslot is in the before all the other patients scheduled appointments.

Hospital 2

1. State the data structure used to store patients and explain why this data structure is the best for the task in hand.

A sorted linked list was used as the hospital allowed for O(n) time complexity for the adding a patient. By using a sorted linked list and inserting the patient in its sorted position, the iterator function can be called as quickly as possible.

2. Describe the algorithm used to order the patients. Briefly explain how it works, from where it is called (from iterator/__iter__ or from addPatient/add_patient) and why it is the best algorithm in comparison with the other known algorithms.

The addPatient function adds the new patient into its sorted position by parsing through the linked list till its position is found. If the timeslot is already taken, the patient is added given last priority and inserted after all other patients of that given timeslot. This allows the iterator function to only require the head node where it then parses through the already sorted linked list. This algorithm was used as opposed to other algorithms as addPatient or insertion was to be done in O(n) time and by doing so, made sure the iterator function would be completed as quickly as possible.

3. Assuming n to be the number of patients, state the best-case (Ω) and worst-case (O) time complexity of iterator/__iter__ and addPatient/add_patient with respect to n.

Iterator:
Iterator function and associated functions (hasnext() and next()) all take O(1) in the best and worst case time complexity as the list is already sorted.

addPatient:
The worst-case time complexity for adding a patient would be O(n) in the situation where the patient timeslot is after all the other patients scheduled appointments. The best-case time complexity would be Ω(1) in the situation where the timeslot is in the before all the other patients scheduled appointments.

Hospital 3

1. State the data structure used to store patients and explain why this data structure is the best for the task in hand.

An array was used for addPatient to have the fastest insert time complexity as this was a requirement for the hospital. An array would be able to add a value in O(1) while also maintaining the input value order which will be important for the sorting algorithm used to get the lowest complexity.

2. Describe the algorithm used to order the patients. Briefly explain how it works, from where it is called (from iterator/__iter__ or from addPatient/add_patient) and why it is the best algorithm in comparison with the other known algorithms.

Merge sort was used to sort the array for the iterator function. This algorithm recursively divides the unsorted list by 2 until n sub lists containing one element is left, which is then merged and compared to other sub lists until the list is sorted. This sorting algorithm has one of the fastest time complexities and is also stable. As patients are added to the end of the list, when the list is stable sorted, the original order of input is maintained, so the patient first added for the same timeslot is given priority.

3. Assuming n to be the number of patients, state the best-case (Ω) and worst-case (O) time complexity of iterator/__iter__ and addPatient/add_patient with respect to n.

Iterator:
When the iterator function is called, merge sort is called on the list, once called this algorithm takes O(nlogn) for the worst-case time complexity and Ω(nlogn) for the best time case time complexity. The associated functions hasnext() and and next() will take O(1).

addPatient:
As patients will just be added to the end of the list, the best-case time complexity will take Ω (1). The worst case however is when the list needs to be resized into a larger array, this will take O(n) time for the worst-case time complexity.

Login System (14 Marks)

1. What is the advantage of storing a hash code of the password instead of the plain text password?

By hashing a password, you don't need to store the password preventing hackers from getting users passwords as it can't be reversed in a reasonable amount of time. Passwords can still be authenticated if they have the same hash code which is the only purpose of the password as it doesn't need to be uniquely identified.

2. Do we need to store the email in the hash table? If your answer is yes, explain why it is necessary. If your answer is no, explain how you use the email (if you use it at all).

Yes, as you need to make sure the email is unique, so it needs to be saved as a text. When trying to find the key (email) of a user, it is put through a hash function to get its position but still needs to be uniquely identifiable, therefore it is stored as a text.

3. Which of the following is an example of a collision? Explain your answer for both cases.

   (a) Two users have the same email hash

   (b) Two users have the same password hash

(a)
This is an example of a collision as the key is the email address which is also used to get its position in the hash table. When two keys share their hash value, so they are in the same index position in an array, ways to deal with hash collisions need to be taken to allow both key values to be inserted, for example linear probing.

(b)
This is not an example of collision as passwords aren't used to get the index position of the key value pair, but instead are stored according to the key.

4. What is the type of hash code function being used? Explain why it is suitable for use in this hash table.

Polynomial accumulation is being used to reduce the chances of collisions due to the hash value getting increasingly higher with each element in the string. The constant value being multiplied by, 31, is also a prime number which is another factor in reducing the probability of a collision.

Tree of Symptoms (10 Marks)

1. What is the type of the restructured tree?

A binary search tree is being used.

2. For any given binary tree, is the reconstructed tree **balanced**? If so, explain why. If not, give a counter-example.

No as the root node must be of a specific severity value. This becomes a problem if the root node is unbalanced, by tri-node restructuring the root node, a different symptom will replace its position.
For example, a root symptom node with a severity of 5 is unbalanced, so their left child has a height of 3 while the right has a height of 5 making giving a height difference of 2. This requires a tri-node restructuring of making the right child the new root breaking the rule that the root must be a specific symptom of the old tree, one which satisfied severity >= threshold or was the most severe symptom is the initial condition couldn't be met.

3. For any given binary tree, is the reconstructed tree **unique**? If so, explain why. If not, give a counter-example.

No as the order in which the symptoms are inserted into the tree can affect its structure. For example, if the tree has a root of 5 and then had to add the values 3 and 4, the order in which these two values were inserted matter. If 3 was added first, 4 would be the right child of 3 however if 4 was inserted first, 3 would be the left child of 4.

END OF REPORT

ALIGNMENT TEST BOX
DO NOT EDIT