COMP7505 Project - Part B

Weighting: 15%

Due date: 18th October at 4:00 PM AEST

Task

This project will require you to implement algorithms efficiently in order to solve desired tasks. For each question in this report you will be required to:

1. Complete programming task(s) in reference to a given algorithm or desired efficiency.
2. Write a report explaining the efficiency of your implementation and/or some other algorithmic analysis.

The specific details required for each of these two points are outlined in the questions below.

Submission Details

All assignment related documents and files will be submitted via Gradescope. The programming sections of your assignment will be marked by auto-grading software while the report section of your assignment will be marked by a tutor. Therefore, you will need to submit to two separate Gradescope submission portals. The name of these submission portals and the documents required to submit to them are listed below.

- **Autograder:** You must submit the programming part of your assignment to the autograder portal. You should only submit either the Java or Python code according to the language you have used. You *must not* submit your report PDF here. If you do so and fail to submit the project PDF to the Report Portal (see below), penalties will apply. You should submit only the following files: `airport.py, routine.py, security_db.py` for Python or `Airport.java, SecurityRoutine.java, SecurityDB.java` for Java.
- **Report:** You must submit a PDF export of your report to the report portal. You must use the template provided for your report. If you do not use the template or you modify the template in anyway, with the exemption of adding your answers in the box provided, you will incur a penalty.

Late Submissions and Extensions

Please consult the courses ECP for the policies and procedures regarding late submission and extensions. Note that course staff cannot process requests for extension or otherwise. All such applications must be made through the school in accordance with ECP.

Academic Misconduct

This assignment is an individual assignment. Posting questions or copying answers from the internet is considered cheating, as is sharing your answers with classmates. All your work (including code) will be analysed by sophisticated plagiarism detection software.

Students are reminded of the University's policy on student misconduct, including plagiarism. See the course profile and the School web page: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism.

Part B

1 Security Database (40 Marks)

1.1 Context

Once a plane has landed, the first name and last name of all passengers, along with their passport identification number are processed and stored by the airport security team. Although two people may have the same name, all passport numbers are unique. This means a specific passport ID can only belong to one person. If a person tries to enter security using a passport which has already been used to enter by another person with a different name, then the system will raise an alert. Additionally, one person can only pass through airport security up to a maximum of 5 times. If they attempt to pass through security more than 5 times, the system should raise an alert, and `add_passenger`/`addPassenger` should return false.

When an action raises an alert in the system, the string "Suspicious behaviour" should be printed to standard error with no trailing newline.

1.2 Requirements

Java Requirements

- You must use Java version 11 or higher. Additional language features introduced after version 11 may not be supported. It is recommended you use OpenJDK, for example, AdoptOpenJDK 11.
- Your solution will be automatically marked. No marks will be awarded for non-compiling submissions.
- You should NOT use any additional classes from the Java Collections Framework (e.g. `ArrayList`, `LinkedList`, `HashMap`, `HashSet`). If you wish to utilise similar functionality, you should implement the needed data structures yourself. If you are unsure about whether a certain class is allowed to be used, please contact teaching staff immediately.
- You should NOT use any additional third-party libraries. No marks will be awarded to submissions which import non-supported libraries.
- Remove the `main()` method and any `System.out.print()` calls before submitting to the autograder.

Python Requirements

- You must use Python version 3.7.
- Your solution will be automatically marked.
- You should NOT use dictionary `dict` nor {}. You should NOT use the Python built-in hash function `hash()`. If you wish to utilise similar functionality, you should implement the needed methods yourself.
- You should NOT use any additional third-party libraries, such as `numpy`, `scipy`. No marks will be awarded to submissions which import non-supported libraries.
- Remove `__main__` method and `print` functions before submitting to the autograder.

PROJECT CONTINUES ON NEXT PAGE

1.3 Programming (20 Marks)

Assuming that there are at most N planes per day with around K passengers per plane, your task is to implement all the functions in `security_db_base` (Python) or `SecurityDBBase` (Java).

Your solution *must* make use of a hash table to efficiently store the passengers. Your implementation of the hash table must conform to the following requirements:

- **Size:** The hash table should be initialised to a static size M. M should be a prime number, larger than $N \times K$. If the number of passengers at any point exceeds the expected value, resize the hash table to its maximum capacity of 1021. You can assume generally $N \times K < 1021$.
- **Hash Function:** You should implement a hash function that consists of a custom hash code and a compression function.
  - **Hash Code:** For the hash codes use a cumulative component sum. Assume that the key represents a string $s : s_1, s_2, s_3, ..., s_n$; and $A(s_n)$ represents an ASCII value for the letter s_n. Then the hash codes should be calculated as follows:

$$h_1(s_1, s_2, .., s_n) = \Big(1 + A(s_1)\Big) + \Big(1 + A(s_1) + A(s_2)\Big) + ... + \Big(1 + A(s_1) + ... + A(s_n)\Big)$$

  - **Compression Function:** For the compression function, use a division by the hash table size:

$$h_2(y) = y \ mod \ M$$

- **Collision Handling:** Use linear probing to handle all collisions.

1.4 Report (20 Marks)

Using the Gradescope document template provided, you must complete a report which analyses your code and the algorithms you have implemented. The report will be broken up into the following three sections:

1. State the best and worst case time complexity of adding and removing a passenger. Briefly explain your answer.
2. Describe the advantage(s) of the cumulative component sum hashcode function over a regular summation of the ASCII values.
3. The hash table you have implemented for the security database starts at a fixed size, M, then will only increase once to the maximum size of 1021. For a regular hash table implementation using the doubling strategy though, why do we need to expand the size of the hash table when it reaches a load factor of 75% and not when it nears 100%?
4. If separate chaining was used instead of linear probing for collision handling, which data structure could be used instead of linked lists to make the implementation more efficient?

PROJECT CONTINUES ON NEXT PAGE

2 Shuttle Recommendation System (40 Marks)

2.1 Context

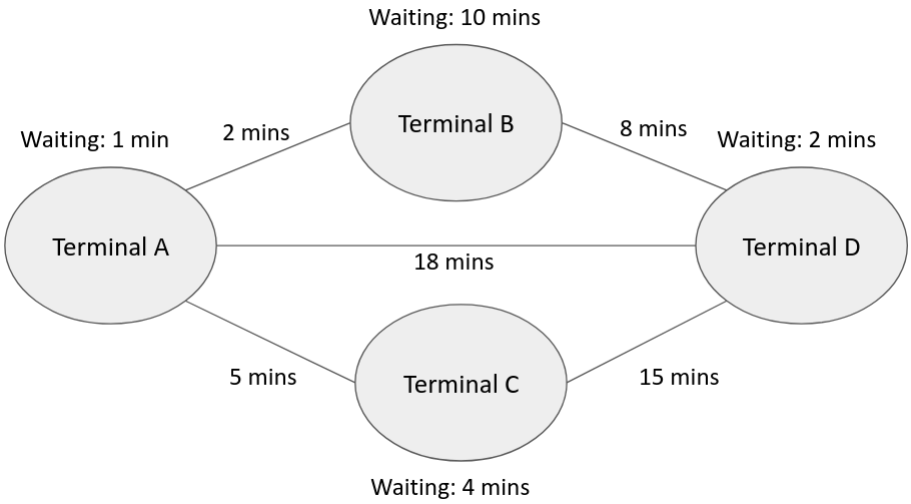

Figure 1: Graph of terminals in the airport

Once in the airport, a transiting passenger might need to change between terminals. Big airports have a complex system of shuttles circulating between them. This system can be described in the form of a graph, where the nodes represent terminals and the edges represent the time it takes for the shuttle to drive between them.

Moreover, some terminals are busier than others, which results in waiting lines for the passengers to enter a shuttle (see Fig.1).

2.2 Programming Requirements

Java Requirements

- You must use Java version 11 or higher. Additional language features introduced after version 11 may not be supported. It is recommended you use OpenJDK, for example, AdoptOpenJDK 11.
- Your solution will be automatically marked. No marks will be awarded for non-compiling submissions.
- You may use additional classes from the Java Collections Framework (e.g. `ArrayList`, `LinkedList`, `HashMap`, `HashSet`).
- You should NOT use any additional third-party libraries. No marks will be awarded to submissions which import non-supported libraries.

Python Requirements

- You must use Python version 3.7.
- Your solution will be automatically marked.
- You may use dictionary `dict` or {}. You should NOT use any additional third-party libraries, such as `numpy`, `scipy`, but you may use `collections`. You may use `heapq`, however we encourage you to use the priority queue class we have provided (`AdaptableHeapPriorityQueue` in `priority_queue.py`). No marks will be awarded to submissions which import non-supported libraries.
- Remove `__main__` method and `print` functions before submitting to the autograder.

2.3 Programming (20 Marks)

Your task is to design a data structure to efficiently store the information about the terminals and shuttles of the airport. Using this data structure, you will then implement all the functions in `airport_base` (Python) or `AirportBase` (Java).

Additionally, you are tasked with writing a recommendation system that will determine an optimal route for a transiting passenger, given the terminal they initially arrived at and the terminal they need to go to. Based on their preferences, passengers can choose between the shortest and the fastest paths.

1. Shortest Path - The path that requires the least number of shuttles.
2. Fastest Path - The path that requires the lowest total time spent travelling and waiting. **Note:** The total time taken to travel a path is defined as the sum of:
   - The sum of the travel times of all shuttles travelled in the path; plus
   - The sum of the waiting times of all terminals included in the path, except the destination terminal.

The number of times a shuttle can circulate between the terminals is defined by its capacity. Each time the shortest/fastest path is calculated, every shuttle included in the calculated path should have its available capacity decreased by 1. Once reaching the capacity, the shuttle becomes deactivated and should be removed from the airport.

2.4 Report (20 Marks)

Using the Gradescope document template provided, you must complete a report which analyses your code and the algorithms you have implemented. The report will be broken up into the following four sections:

1. Describe the data structure you have used to represent a graph and explain its advantages over alternative data structures.
2. Describe the algorithms you used to calculate the shortest path and the fastest path, and state their complexities.
3. An adaptive algorithm takes advantage of some properties of the input. For example, some algorithms do not need to traverse the remaining input once the desired output has been found. Explain which of your shortest/fastest path algorithms (if any) are adaptive and why.
4. (a) Depending on the graph, there may exist two equally fastest paths. For example, in the graph below, there are two fastest paths from terminal A to terminal D: ABD and ACD. Which of these two paths will be chosen by your implementation of the fastest path algorithm, and why?

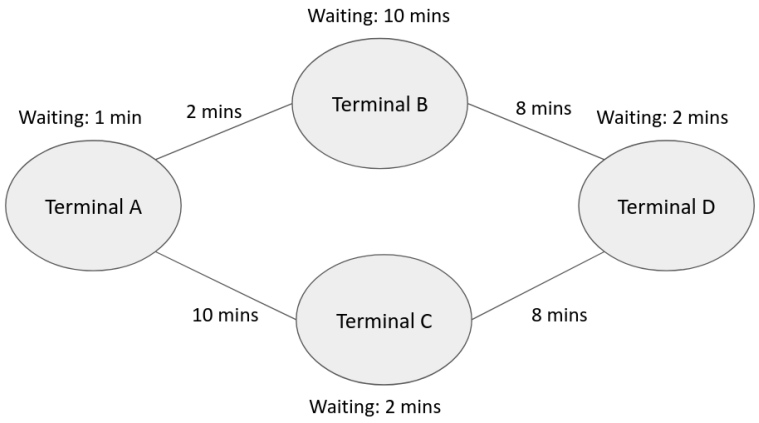

Figure 2: Graph of terminals in the airport

(b) Explain how you would modify your algorithm such that it would choose the other path instead.

3 Security Routine (20 Marks)

3.1 Context

At the beginning and end of each working day, a security check and seal process is performed in the airport. The security guidelines enforce that some areas have to be checked before other areas. Given all these areas and their relative order, propose an algorithm to provide an order for checking all the areas by the security team.

3.2 Programming Requirements

Java Requirements

- You must use Java version 11 or higher. Additional language features introduced after version 11 may not be supported. It is recommended you use OpenJDK, for example, AdoptOpenJDK 11.
- Your solution will be automatically marked. No marks will be awarded for non-compiling submissions.
- You may use additional classes from the Java Collections Framework (e.g. `ArrayList`, `LinkedList`, `HashMap`, `HashSet`).
- You should NOT use any additional third-party libraries. No marks will be awarded to submissions which import non-supported libraries.

Python Requirements

- You must use Python version 3.7.
- Your solution will be automatically marked.
- You may use dictionary `dict` or {}. You should NOT use any additional third-party libraries, such as `numpy`, `scipy`, but you may use `collections`. No marks will be awarded to submissions which import non-supported libraries.
- Remove `__main__` method and `print` functions before submitting to the autograder.

3.3 Programming (20 Marks)

Your task is to design and implement a data structure to **efficiently** store a collection of areas in the airport. Furthermore, you should implement an algorithm to calculate and return the order in which the areas must be checked by the security team.

Note: There is no report task for this question.