UNIVERSITI PUTRA MALAYSIA

BERILMU BERBAKTI

ASSIGNMENT 1

SEMESTER 2 SESSION 2023/2024

CSC4202 DESIGN AND ANALYSIS OF ALGORITHMS

LECTURER' NAME : DR. NUR ARZILAWATI BINTI MD YUNUS

GROUP : 6

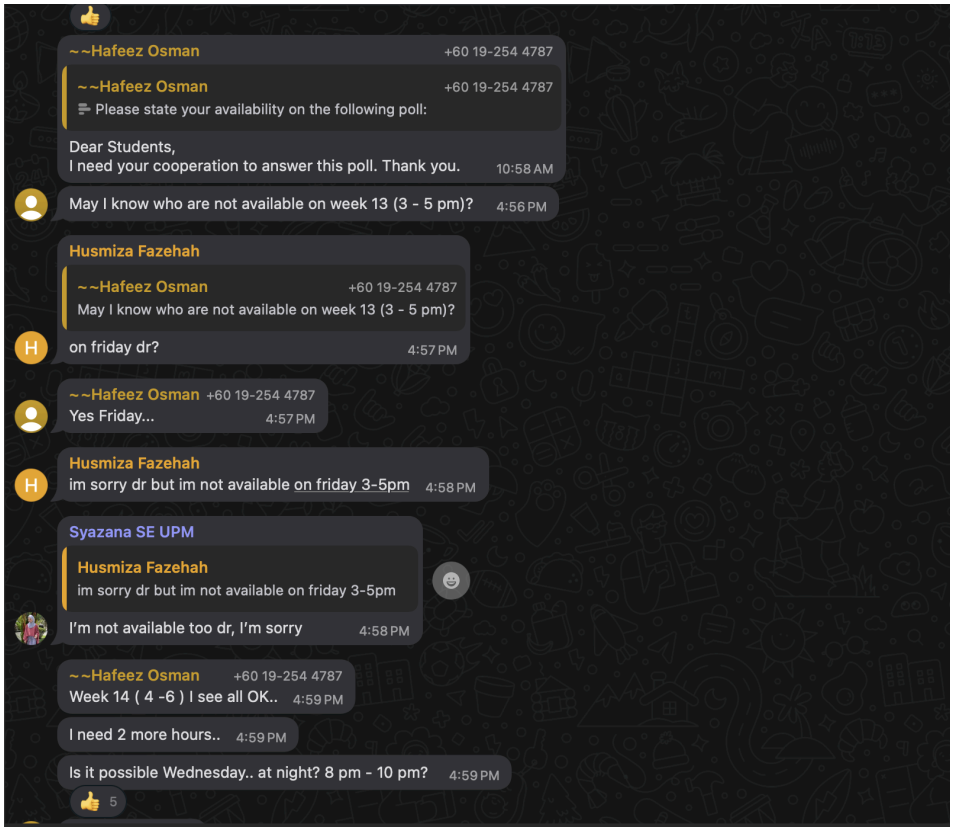
STUDENT'S NAME :

NO.	NAME	MATRIC NO.
1.	MUHAMMAD NAJWAN BIN NIZARIMI	209897
2.	TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI	209861

Table of content

A. Scenario

The illustration below represents students from the Secure Software Development (SWE4356) class at University Putra Malaysia (UPM), where the lecturer struggles to find replacement classes that accommodate all the students.



B. Problem definition

Dr. Hafeez is a dedicated lecturer in the Department of Software Engineering at Universiti Putra Malaysia (UPM), located in Serdang, Malaysia. Throughout the semester, his busy schedule and multiple public holidays occurring on class days have resulted in multiple postponed lectures. This has led to inconsistent student attendance and the risk of important topics being left uncovered before the final exams. To address these challenges, Dr. Hafeez seeks a solution to identify the most optimal time slots for replacement classes by analysing both his and his students' schedules. The proposed algorithm promises efficiency by quickly pinpointing the

best time slots, accuracy by ensuring maximum possible attendance, and convenience by reducing the manual effort in coordination. The goal is to ensure that all missed lectures are rescheduled effectively, maximising student attendance and ensuring complete syllabus coverage. The expected output is a list of optimal time slots for replacement classes without conflicts. This solution aims to make sure all important lectures are completed and students are well-prepared for their final exams.

C. Importances

- **Maintaining Academic Standards**

  Each subject has designated credit hours and content that must be completed to meet educational standards. Missing out on this material can affect the quality of education and students' understanding of the subject.

- **Avoiding Scheduling Conflicts**

  Replacement classes might be scheduled at inconvenient times, especially if done without an optimal solution, leading to conflicts with other classes or personal commitments. This can cause additional stress for both students and instructors.

- **Ensuring Fairness**

  To ensure all students receive the same amount of learning time, which can be unfair if certain students miss out on certain topics due to time constraints.

- **Minimising Disruption**

  Public holidays can already be disruptive enough. Properly scheduled replacement classes can help minimise additional disruptions to the academic calendar and ensure a smooth learning experience.

D. Development of the model

The data used for allocating time slots to find a replacement class consists of the empty or available slots from the lecture and student schedules for the Secure Software Development (SWE4356) class. The replacement class is scheduled to replace for the Eid al-Adha public holiday on Monday, 17th of June 2024, for 2 hours, and it can be conducted either online or physically

Objective:

To cover all the missed hours of lecture during the holiday, the minimum and maximum number of students required is 38, meaning all students need to attend.

Constraint and Requirements;

- Replacement classes should be scheduled only on weekdays during working hours, from 8 a.m. to 6 p.m.
- Replacement classes can be held online or physically.
- Replacement classes must cover the same number of hours as the original subject schedule during the holiday.
- Replacement classes must not overlap with other scheduled classes.
- A person is considered free when they do not have any classes during that time, and no personal commitments or other events are taken into account.

Data type:

- Lecturer’s timetable

TIME/DAY	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20
MONDAY							SWE4356 (GROUP 1)	SWE4356 (GROUP 1)				
TUESDAY				SWE4356 (GROUP 1)								
WEDNESDAY							SWE4356 (GROUP 2)	SWE4356 (GROUP 2)				
THURSDAY			SWE4356 (GROUP 2)									
FRIDAY												
SATURDAY												
SUNDAY												

- Students' timetable

  All of these students are in Group 1 for the subject SWE4356, Secure Software Development. Despite taking the same subjects, there are still differences in the students' timetables due to different courses taken by the students and their different academic years.

  - Student 1: Tuan Nurhusmiza Fazehah binti Tuan Mat Zawai.

Date : 17/24/2024 page 1 / 1

ACADEMIC DIVISION
UNIVERSITI PUTRA MALAYSIA

TIME TABLE
SECOND SEMESTER 2023/2024

NAME : TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI
MATRIC NO. : 209861
FACULTY : FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
PROGRAMME : 48D-2 BACHELOR OF SOFTWARE ENGINEERING WITH HONOURS
MAJOR : NO MAJOR
MINOR : NO MINOR

TIME / DAY	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23
MONDAY	SWE4355 (GROUP 1) BK2(FSKTM) (LECTURE)	SWE4355 (GROUP 1) BK2(FSKTM) (LECTURE)	SSE3400 (GROUP 1) B2-04 (LECTURE)		SWE4357 (GROUP 1) BK2(FSKTM) (LECTURE)		SWE4356 (GROUP 1) A2-06/07 (LECTURE)	SWE4356 (GROUP 1) A2-06/07 (LECTURE)							
TUESDAY	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)	SWE4356 (GROUP 1) A1-01 (LECTURE)			SWE4357 (GROUP 1) A1-01 (LECTURE)	SWE4357 (GROUP 1) A1-01 (LECTURE)							
WEDNESDAY			CEL2105 (GROUP 61) DKB (LECTURE)	CEL2105 (GROUP 61) DKB (LECTURE)	SWE4355 (GROUP 1) BK2(FSKTM) (LECTURE)		CSC4202 (GROUP 6) BK1(FSKTM) (LECTURE)	CSC4202 (GROUP 6) BK1(FSKTM) (LECTURE)							
THURSDAY	CSC4202 (GROUP 6) A1-01 (LECTURE)	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK.KP1 FSKTM (LAB)											
FRIDAY															
SATURDAY															
SUNDAY															

Printed by : TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI
Code : 16003

- Student 2: Harith Dzikri Bin Hashim

Date : 17/24/2024

ACADEMIC DIVISION
UNIVERSITI PUTRA MALAYSIA

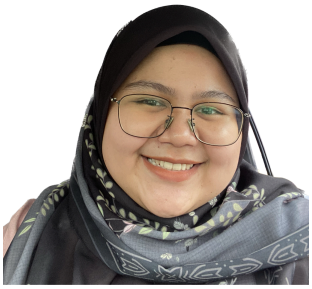
TIME TABLE
SECOND SEMESTER 2023/2024

NAME : HARITH DZIKRI BIN HASHIM
MATRIC NO. : 210183
FACULTY : FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
PROGRAMME : 48D-2 BACHELOR OF SOFTWARE ENGINEERING WITH HONOURS
MAJOR : NO MAJOR
MINOR : NO MINOR

TIME / DAY	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23
MONDAY			SSE3400 (GROUP 2) B2-05 (LECTURE)		SWE4357 (GROUP 1) BK2(FSKTM) (LECTURE)		SWE4356 (GROUP 1) A2-06/07 (LECTURE)	SWE4356 (GROUP 1) A2-06/07 (LECTURE)	CSC4506 (GROUP 1) MAK.PERD ANA FSKTM (LECTURE)						
TUESDAY	SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)	SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)	SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)	SWE4356 (GROUP 1) A1-01 (LECTURE)			SWE4357 (GROUP 1) A1-01 (LECTURE)	SWE4357 (GROUP 1) A1-01 (LECTURE)	CNS3102 (GROUP 2) BK3(FSKTM) (LECTURE)	CNS3102 (GROUP 2) BK3(FSKTM) (LECTURE)					
WEDNESDAY	CSC4506 (GROUP 1) MAK.PERD ANA FSKTM (LECTURE)	CSC4506 (GROUP 1) MAK.PERD ANA FSKTM (LECTURE)	CEL2105 (GROUP 61) DKB (LECTURE)	CEL2105 (GROUP 61) DKB (LECTURE)											
THURSDAY	CNS3102 (GROUP 2) MAK.RANG KAIAN FSKTM2 (LAB)	CNS3102 (GROUP 2) MAK.RANG KAIAN FSKTM2 (LAB)	CNS3102 (GROUP 2) MAK.RANG KAIAN FSKTM2 (LAB)												
FRIDAY		SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)	SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)	SSE3400 (GROUP 2) MAK.KP2 FSKTM (LAB)											
SATURDAY															
SUNDAY															

Printed by : HARITH DZIKRI BIN HASHIM
Code : 16003

- Student 3: Nur Hanan binti Ahmad Sallehuddin



Tarikh : 17/24/2024

m/s : 1 / 1

BAHAGIAN AKADEMIK
UNIVERSITI PUTRA MALAYSIA

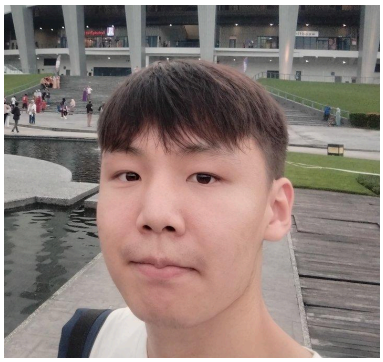
JADUAL WAKTU
SEMESTER KEDUA SESI 2023/2024

NAMA : NUR HANAN BINTI AHMAD SALLEHUDDIN
NO. MATRIK : 211585
FAKULTI : FAKULTI SAINS KOMPUTER DAN TEKNOLOGI MAKLUMAT
PROGRAM : 48D-2 BACELOR KEJURUTERAAN PERISIAN DENGAN KEPUJIAN
MAJOR : TIADA MAJOR
MINOR : TIADA MINOR

MASA / HARI	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23
ISNIN	SWE4355 (KUMP 1) BK2(FSKTM) (KULIAH)	SWE4355 (KUMP 1) BK2(FSKTM) (KULIAH)	SSE3400 (KUMP 2) B2-05 (KULIAH)		SWE4357 (KUMP 1) BK2(FSKTM) (KULIAH)		SWE4356 (KUMP 1) A2-06/07 (KULIAH)	SWE4356 (KUMP 1) A2-06/07 (KULIAH)							
SELASA	SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)	SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)	SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)	SWE4356 (KUMP 1) A1-01 (KULIAH)			SWE4357 (KUMP 1) A1-01 (KULIAH)	SWE4357 (KUMP 1) A1-01 (KULIAH)	CNS3102 (KUMP 2) BK3(FSKTM) (KULIAH)	CNS3102 (KUMP 2) BK3(FSKTM) (KULIAH)					
RABU				SWE3307 (KUMP 2) A2-08 (KULIAH)	SWE4355 (KUMP 1) BK2(FSKTM) (KULIAH)										
KHAMIS	CNS3102 (KUMP 2) MAK.RANG KAIAN FSKTM2 (AMALI)	CNS3102 (KUMP 2) MAK.RANG KAIAN FSKTM2 (AMALI)	CNS3102 (KUMP 2) MAK.RANG KAIAN FSKTM2 (AMALI)	SWE3307 (KUMP 2) B3-08/09 (KULIAH)	SWE3307 (KUMP 2) B3-08/09 (KULIAH)										
JUMAAT		SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)	SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)	SSE3400 (KUMP 2) MAK.KP2 FSKTM (AMALI)											
SABTU															
AHAD															

Dicetak Oleh : NUR HANAN BINTI AHMAD SALLEHUDDIN
Code : 16003

- Student 4: Xue Rui



Date : 26/24/2024

ACADEMIC DIVISION
UNIVERSITI PUTRA MALAYSIA

TIME TABLE
SECOND SEMESTER 2023/2024

NAME : XUE RUI
MATRIC NO. : 209761
FACULTY : FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
PROGRAMME : 48D-2 BACHELOR OF SOFTWARE ENGINEERING WITH HONOURS
MAJOR : NO MAJOR
MINOR : NO MINOR

TIME / DAY	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21	21-22	22-23
MONDAY			SSE3400 (GROUP 1) B2-04 (LECTURE)		SWE4357 (GROUP 2) A2-09 (LECTURE)		SWE4356 (GROUP 1) A2-06/07 (LECTURE)	SWE4356 (GROUP 1) A2-06/07 (LECTURE)							
TUESDAY	SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)	SWE4356 (GROUP 1) A1-01 (LECTURE)			SWE4357 (GROUP 2) A2-06/07 (LECTURE)	SWE4357 (GROUP 2) A2-06/07 (LECTURE)							
WEDNESDAY			CEL2105 (GROUP 60) B3-08/09 (LECTURE)	CEL2105 (GROUP 60) B3-08/09 (LECTURE)			SKM3002 (GROUP 3) M.P.DATA FSKTM (LECTURE)								
THURSDAY		SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)	SSE3400 (GROUP 1) MAK,KP1 FSKTM (LAB)				SKM3002 (GROUP 3) BK3(FSKTM) (LECTURE)	SKM3002 (GROUP 3) BK3(FSKTM) (LECTURE)						
FRIDAY															
SATURDAY															
SUNDAY															

Printed by : XUE RUI
Code : 16003

E. Algorithms' Review

- **Sorting Algorithm**

  Sorting algorithm is used to rearrange the given data in particular order, typically in ascending or descending order.

  Strengths:

  - Sorting algorithms can be a straightforward way to organise schedules by date or availability.
  - Sorting algorithms is faster to access the data by preprocessing steps for more complex algorithms.

  Weaknesses:

  - Sorting alone does not address the complexity of schedule conflicts and optimization for maximum attendance because handling replacement classes involves allocating time slots that accommodate everyone, not just arranging data in order.

- **Divide and Conquer Algorithm (DAC)**

  This algorithm recursively breaks down a problem into two or more subproblems of the same or related type, until these sub-problems become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

  Strengths:

  - DAC is efficient as it can break down a problem into smaller subproblems and solve it independently before combining it back, which potentially can speed up the computation.

  Weaknesses:

- DAC is less effective when the sub-problems always depend on the solution to the other subproblems such as finding the optimal time for one lecture might affect or be affected by the scheduling of other lectures.
- DAC can be complicated to implement for problems that have many dependencies and constraints.

- **Dynamic Programming Algorithms**

  Dynamic programming algorithm is used for problems with complex problems and overlapping subproblems. It will break down the problem into smaller and similar subproblems and store the solution to reuse the previous solution.

  Strengths:

  - Dynamic programming can be used to find optimal time slots for replacement classes by recursively considering sub-problems which is scheduling lectures scheduling lectures to maximise attendance while minimising conflicts with Dr. Hafeez's and students' schedules.
  - Dynamic programming can handle complex constraints and dependencies between different parts of the problem including Dr Hafeez's availability and students' availability.

  Weakness:

  - DP can require significant memory especially with large inputs.
  - The design of a DP solution can be complex as it requires careful definition of states and transitions.

- **Greedy Algorithm**

  Greedy algorithm is an approach that builds up a solution piece by piece by choosing the best option at each step without reconsidering its choices.

  Strengths:

  - Greedy algorithms are easy to implement as it often requires less code due to its straightforward character.

- Greedy algorithms operate on a simple principle where it choose the available option without considering the broader consequences.

Weakness:

- Greedy algorithms do not always produce optimal results especially when future decisions are dependent on earlier decisions.
- Greedy algorithms may not be suitable for complex scheduling problems with multiple dependencies and constraints.

- **Graph Algorithm**

  Graph algorithm is using nodes and edges to represent the problem.

  Strengths:

  - Graph algorithms can represent the relationship between time slots, conflicts and dependencies to identify the time slots for replacement classes.
  - Graph algorithms can be used to find the shortest path to schedule solutions that maximise student attendance and match available time slots to minimise conflicts.

  Weaknesses:

  - Graph algorithms can be used under the assumption of static data and reflects that does not reflect real-word dynamic changes if there are any changes in Dr. Hafeez's and students' timetable.

F. Algorithm Specification

G.

Between all these 5 algorithms, the best to choose for allocating the time slots (2 hours) for replacement classes are Greedy algorithm and Dynamic Programming Algorithm.

Aspect	Greedy Algorithm	Dynamic Programming Algorithm
Approach	Makes locally optimal choices at each step without considering future consequences.	Solves problems by breaking them down into smaller overlapping subproblems.
Decision Making	Makes decisions based on immediate gains or local optimal solutions	Evaluates all possible decisions systematically to find the optimal solution
Backtracking	Does not involve backtracking	Does involve backtracking
Complexity	Lower time and space complexity	Higher time and space complexity
Application	Suitable for quick scheduling with fewer constraint	Suitable if multiple constraints and factors need thorough analysis
Examples	Minimum Spanning Tree, Shortest Path algorithms.	Fibonacci sequence, Longest Common Subsequence.

For the Greedy algorithm, the best option is selected at each step without reconsidering previous choices. This approach can help quickly identify the earliest available time slot with the most students. Greedy algorithms are generally faster and require less memory. This method is a good choice if the goal is to quickly

allocate a time slot for replacement classes, although it might miss out on the best possible schedule due to its locally optimal choices.

In contrast, the Dynamic Programming algorithm breaks down the problem of scheduling replacements into smaller subproblems. This includes allocating time slots over five days while considering various constraints, such as the lecturer's availability, students' availability, time slots, and working hours. However, this algorithm has higher time and space complexity because it needs to store intermediate results.

For this scenario, the Dynamic Programming algorithm is better suited as it provides multiple choices for the students and lecturers, resulting in an optimal allocation of time for replacement classes.

H. Algorithm Design

```
START

DEFINE WORKING_HOURS_START AS 8
DEFINE WORKING_HOURS_END AS 18
DEFINE REQUIRED_FREE_HOURS AS 2

CLASS DPAlgorithm

  FUNCTION findContinuousFreeSlot(lecturerAvailability, studentAvailability)
                INITIALIZE dp AS ARRAY OF BOOLEAN WITH LENGTH (WORKING_HOURS_END - WORKING_HOURS_START + 1)

    FOR hour FROM WORKING_HOURS_START TO WORKING_HOURS_END
                  IF isSlotAvailable(hour, hour + REQUIRED_FREE_HOURS, lecturerAvailability)
                  AND isSlotAvailable(hour, hour + REQUIRED_FREE_HOURS, studentAvailability)
```

```
            SET dp[hour - WORKING_HOURS_START] TO TRUE
        ELSE
            SET dp[hour - WORKING_HOURS_START] TO FALSE
        END IF
    END FOR

    printContinuousFreeSlots(dp)
END FUNCTION

FUNCTION isSlotAvailable(startHour, endHour, availability)
    FOR hour FROM startHour TO endHour - 1
        IF availability[hour] IS FALSE
            RETURN FALSE
        END IF
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION printContinuousFreeSlots(dp)
    PRINT "Free 2-hour slots:"
    FOR hour FROM 0 TO LENGTH(dp) - REQUIRED_FREE_HOURS
        IF dp[hour] IS TRUE AND dp[hour + 1] IS TRUE
                PRINT "Free slot from", WORKING_HOURS_START + hour, "to",
WORKING_HOURS_START + hour + REQUIRED_FREE_HOURS
        END IF
    END FOR
END FUNCTION

END CLASS

FUNCTION main()
    INITIALIZE lecturerAvailability AS ARRAY OF BOOLEAN WITH LENGTH 24
    INITIALIZE studentAvailability AS ARRAY OF BOOLEAN WITH LENGTH 24
```

```
    FOR hour FROM 0 TO 23
        SET lecturerAvailability[hour] TO TRUE
        SET studentAvailability[hour] TO TRUE
    END FOR

    SET lecturerAvailability[10] TO FALSE
    SET studentAvailability[15] TO FALSE

    CREATE instance OF DPAlgorithm

    CALL instance.findContinuousFreeSlot(lecturerAvailability, studentAvailability)

END FUNCTION

END
```

I. Algorithm Analysis and Correctness

To ensure the algorithm correctness, it can be analysed through both its asymptotic complexity and recurrence relations. Asymptotic complexity refers to how the runtime of an algorithm behaves as the size of the input grows towards infinity in terms of time and space. For the pseudocode, the implementation of dynamic programming to allocate 2-hours free time slots for both lecturer and students, the overall time complexity is assessed as O(d x n), where d; number of days and n; number of people.

Time Complexity is the time measured when the number of unique subproblems is multiplied with the time taken per subproblem. For this pseudocode, the best-case scenario occurs when the first checked slot from 8 a.m. to 10 a.m. is free for both the lecturer and all students. In this case, time complexity is *O(n x m),* where *n* is the number of students and m is the length of the working hours. For average-case, the function will need to check multiple slots before finding a free one. The complexity

remains *O(n x m x p),* where n is the number of students, m is the length of the working hours, and p is the average length of the availability lists. Lastly, the worst-case scenario occurs when there are no free slots available, requiring the function to check all possible slots for each student. The worst-case complexity is O(n x m x p).

For the space, the implementation, a boolean size of array ‘dp’ is equal to number of working hours to store intermediate results of availability checks.

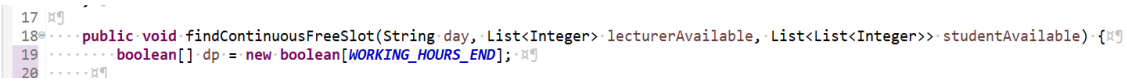
The recurrence relations in this algorithm specifies how solutions to larger subproblems depend on solutions to smaller subproblems. In dynamic programming, a recurrence relation defines how the solution to a larger problem can be constructed from solutions to smaller subproblems. It's a way of recursively defining the value of a function based on its previous values. In this problem context, the recurrence relation is expressed as ‘dp [i] = lecturerFree && studentsFree where dp [i] denotes whether a 2-hour free slot starting at hour ‘i’ is practical based on availability checks for both lecturer and students. This relation builds upon smaller subproblems to construct the solution for the entire working period.

To ensure the algorithm's correctness, it should include the base case and inductive step. The base case is initialising ‘boolean[] dp = new boolean [WORKING_HOURS_END]’. This array serves as the dynamic programming table where each index represents an hour in the working day. Initialization typically sets the starting conditions for the algorithm to start. The inductive step progresses from solving smaller subproblems to larger ones. Within the ‘findContinuousFreeSlot’ method, the recurrence relation is applied through the loop that iterates over each hour. This loop checks if both the lecturer and all students are available for a 2-hour time slot starting at each hour ‘i’. The ‘dp[i]’ entry is set to ‘true’ if the conditions are met, otherwise false.After populating the ‘dp’ array, another loop collects all hours where ‘dp[i]’ is true into the ‘freeSlots’ list, which represents the valid 2-hour continuous time slots.

J. Java language program (implementation)

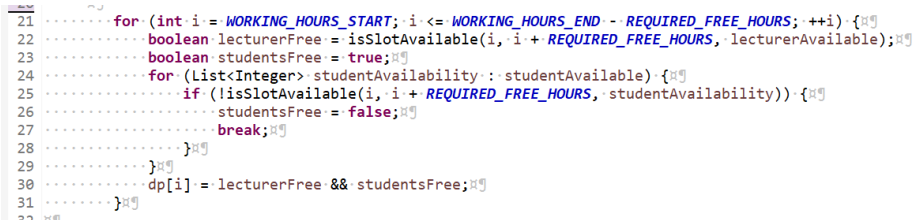
Sneak peak

- Initialisation of ‘dp’ array : To store whether each hour in the working day has a 2-hour free slot available (‘true’) or not (‘false’).



```
17 ¤¶
18    public void findContinuousFreeSlot(String day, List<Integer> lecturerAvailable, List<List<Integer>> studentAvailable) {¤¶
19        boolean[] dp = new boolean[WORKING_HOURS_END];¤¶
20    ¤¶
```

- Filling the ‘dp’ array: The ‘for’ loops iterates through each hour in the range from ‘WORKING_HOURS_START’ to ‘WORKING_HOURS_END - REQUIRED_FREE_HOURS’. For each ‘i’, it checks;
    - If the lecturer is available (lecturerFree).
    - If all students are available (studentsFree).
- DP Table Update: ‘dp[i]’ is set to ‘true’ if both conditions are met, indicating that a 2-hour free slot starts at hour ‘i’.



```
21        for (int i = WORKING_HOURS_START; i <= WORKING_HOURS_END - REQUIRED_FREE_HOURS; ++i) {¤¶
22            boolean lecturerFree = isSlotAvailable(i, i + REQUIRED_FREE_HOURS, lecturerAvailable);¤¶
23            boolean studentsFree = true;¤¶
24            for (List<Integer> studentAvailability : studentAvailable) {¤¶
25                if (!isSlotAvailable(i, i + REQUIRED_FREE_HOURS, studentAvailability)) {¤¶
26                    studentsFree = false;¤¶
27                    break;¤¶
28                }¤¶
29            }¤¶
30            dp[i] = lecturerFree && studentsFree;¤¶
31        }¤¶
```

- Utilisation of results: After populating the ‘dp’ array, this loop iterates through it to identify all hours (‘i’) where ‘dp[i]’ is ‘true’. Each such ‘i’ represents the start of a 2-hour free slot.
- By using the ‘dp’ array, the algorithm efficiently identifies and collects all valid 2-hour free slots without re-evaluating the availability conditions multiple times

```
        List<Integer> freeSlots = new ArrayList<>();
        for (int i = WORKING_HOURS_START; i <= WORKING_HOURS_END - REQUIRED_FREE_HOURS; ++i) {
            if (dp[i]) {
                freeSlots.add(i);
            }
        }

```

Program testing

Full code

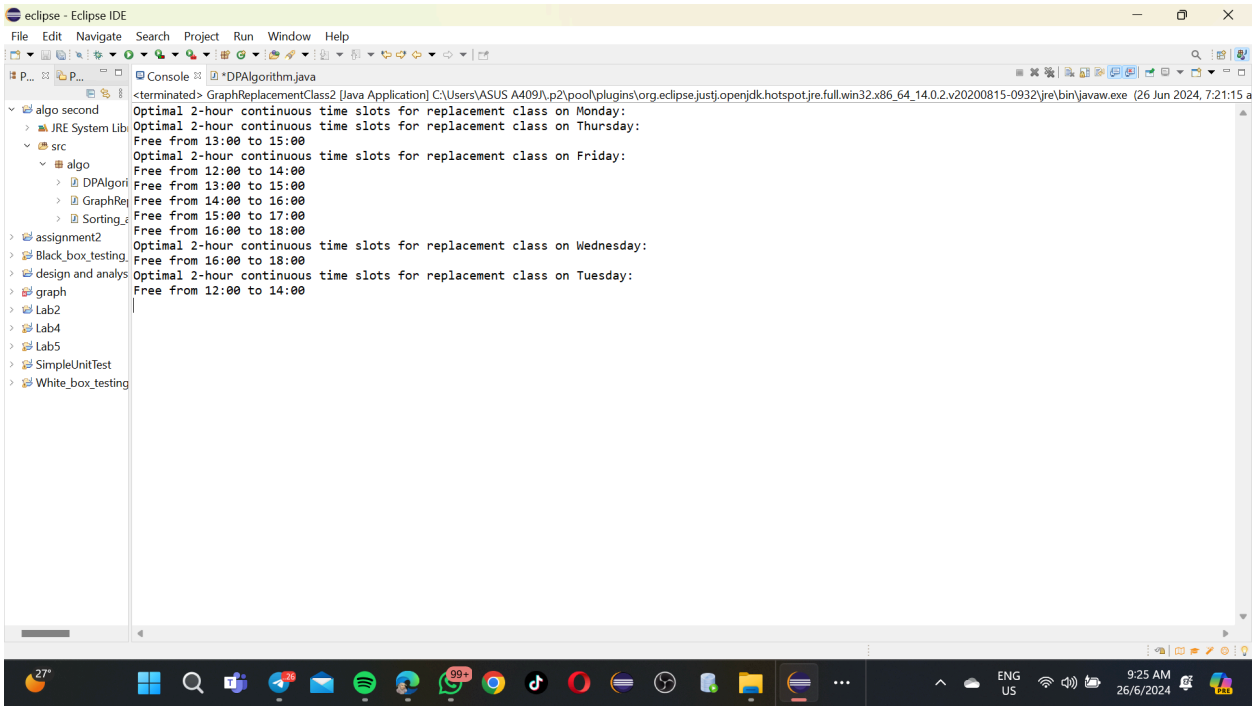
Console *DPAlgorithm.java

```
package algo;

import java.util.ArrayList;

public class DPAlgorithm {

    private static final int WORKING_HOURS_START = 8; // 8am
    private static final int WORKING_HOURS_END = 18; // 6pm
    private static final int REQUIRED_FREE_HOURS = 2; // 2 hours time slots

    public DPAlgorithm() {
    }

    public void findContinuousFreeSlot(String day, List<Integer> lecturerAvailable, List<List<Integer>> studentAvailable) {
        boolean[] dp = new boolean[WORKING_HOURS_END];

        for (int i = WORKING_HOURS_START; i <= WORKING_HOURS_END - REQUIRED_FREE_HOURS; ++i) {
            boolean lecturerFree = isSlotAvailable(i, i + REQUIRED_FREE_HOURS, lecturerAvailable);
            boolean studentsFree = true;
            for (List<Integer> studentAvailability : studentAvailable) {
                if (!isSlotAvailable(i, i + REQUIRED_FREE_HOURS, studentAvailability)) {
                    studentsFree = false;
                    break;
                }
            }
            dp[i] = lecturerFree && studentsFree;
        }


        List<Integer> freeSlots = new ArrayList<>();
        for (int i = WORKING_HOURS_START; i <= WORKING_HOURS_END - REQUIRED_FREE_HOURS; ++i) {
            if (dp[i]) {
                freeSlots.add(i);
            }
        }

```

```
        printContinuousFreeSlots(day, freeSlots);
    }

    private boolean isSlotAvailable(int start, int end, List<Integer> availableTimes) {
        for (int i = start; i < end; ++i) {
            if (!availableTimes.contains(i)) {
                return false;
            }
        }
        return true;
    }

    private void printContinuousFreeSlots(String day, List<Integer> freeSlots) {
        System.out.println("Optimal 2-hour continuous time slots for replacement class on " + day + ":");
        for (int i = 0; i < freeSlots.size(); ++i) {
            System.out.println("Free from " + freeSlots.get(i) + ":00 to " + (freeSlots.get(i) + REQUIRED_FREE_HOURS) + ":00");
        }
    }

    public static void main(String[] args) {
        DPAlgorithm scheduler = new DPAlgorithm();


        Map<String, List<Integer>> lecturerAvailable = new HashMap<>();
        Map<String, List<List<Integer>>> studentAvailable = new HashMap<>();


        // Monday availability
        lecturerAvailable.put("Monday", Arrays.asList(8, 9, 10, 11, 12, 13, 16, 17)); // Lecturer Dr Hafeez
        studentAvailable.put("Monday", Arrays.asList(
            Arrays.asList(11, 13, 16, 17), // Student 1 (Miza)
            Arrays.asList(8, 9, 11, 13, 17), // Student 2 (Jiki)
            Arrays.asList(11, 13, 16, 17), // Student 3 (Hanan)
            Arrays.asList(8, 9, 11, 13, 16, 17)  // Student 4 (Xue Rui)
        ));


        // Tuesday availability
        lecturerAvailable.put("Tuesday", Arrays.asList(8, 9, 10, 12, 13, 14, 15, 16, 17)); // Lecturer Dr Hafeez
        studentAvailable.put("Tuesday", Arrays.asList(
            Arrays.asList(8, 9, 10, 12, 13, 14, 15, 16, 17), // Student 1 (Miza)
            Arrays.asList(12, 13), // Student 2 (Jiki)
            Arrays.asList(12, 13), // Student 3 (Hanan)
            Arrays.asList(12, 13, 16, 17)  // Student 4 (Xue Rui)
        ));

        // Wednesday availability
        lecturerAvailable.put("Wednesday", Arrays.asList(8, 9, 10, 11, 12, 13, 16, 17)); // Lecturer Dr Hafeez
        studentAvailable.put("Wednesday", Arrays.asList(
            Arrays.asList(8, 9, 13, 16, 17), // Student 1 (Miza)
            Arrays.asList(12, 13, 14, 15, 16, 17), // Student 2 (Jiki)
            Arrays.asList(8, 9, 10, 13, 14, 15, 16, 17), // Student 3 (Hanan)
            Arrays.asList(8, 9, 12, 13, 15, 16, 17)  // Student 4 (Xue Rui)
        ));

        // Thursday availability
        lecturerAvailable.put("Thursday", Arrays.asList(8, 9, 11, 12, 13, 14, 15, 16, 17)); // Lecturer Dr Hafeez
        studentAvailable.put("Thursday", Arrays.asList(
            Arrays.asList(12, 13, 14, 15, 16, 17), // Student 1 (Miza)
            Arrays.asList(11, 12, 13, 14, 15, 16, 17), /// Student 2 (Jiki)
            Arrays.asList(13, 14, 15, 16, 17), // Student 3 (Hanan)
            Arrays.asList(8, 12, 13, 14)  // Student 4 (Xue Rui)
        ));

        // Friday availability
        lecturerAvailable.put("Friday", Arrays.asList(8, 9, 10, 11, 12, 13, 14, 15, 16, 17)); // Lecturer Dr Hafeez
        studentAvailable.put("Friday", Arrays.asList(
            Arrays.asList(8, 9, 10, 11, 12, 13, 14, 15, 16, 17), // Student 1 (Miza)
            Arrays.asList(8, 12, 13, 14, 15, 16, 17), // Student 2 (Jiki)
            Arrays.asList(8, 12, 13, 14, 15, 16, 17), // Student 3 (Hanan)
            Arrays.asList(8, 9, 10, 11, 12, 13, 14, 15, 16, 17)  // Student 4 (Xue Rui)
        ));

        ///
        for (String day : lecturerAvailable.keySet()) {
            scheduler.findContinuousFreeSlot(day, lecturerAvailable.get(day), studentAvailable.get(day));
        }
    }
}

```

Writable | Smart Insert | 76 : 12 : 2877

Output



Based on the given output, multiple 2-hour time slots are displayed for each day, reflecting the availability of both lecturers and students. This allows them to select the preferred time slot for replacement classes, optimising the selection process.

K. Conclusion

In conclusion, the scenario of allocating a replacement class to cover missed syllabus and materials during a public holiday underscores the importance of maintaining academic standards and ensuring fairness. By allocating optimal time slots for replacement classes, we can avoid scheduling conflicts among lecturers and students, minimising disruptions and aligning objectives. To cover all missed lecture hours during the holiday, a minimum and maximum of 38 students are required, meaning full attendance is necessary. Six algorithms learned in class, including sorting, dynamic programming, greedy, divide and conquer, and graph algorithms, can be used to approach this problem. Among these, the optimal algorithm in our case and opinion is dynamic programming.

L. Progress

Initial Project Plan (week 10, submission date: 31 May 2024)

<table>
<tr><td>Group Name</td><td>TuNa</td></tr>
<tr><td>Members</td><td>
<table>
<tr><th>Name</th><th>Email</th><th>Phone number</th></tr>
<tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>209861@student.upm.edu.my</td><td>014-3690236</td></tr>
<tr><td>MUHAMMAD NAJWAN BIN NIZARIMI</td><td>209897@student.upm.edu.my</td><td>011-10794327</td></tr>
</table>
</td></tr>
<tr><td>Problem scenario description</td><td>

Dr Hafeez is a UPM lecturer.
He always postponed his classes and lectures due to his busy schedule and many public holidays.
This has led to inconsistent student attendance and risk of important topics being left uncovered before the final exams.
He wants a solution to identify the most optimal time slots for replacement classes considering both his and his students' schedules.

</td></tr>
<tr><td>Why it is important</td><td>

To maximise student attendance
To ensure complete syllabus coverage

</td></tr>
<tr><td>Problem specification</td><td>

To replace

</td></tr>
<tr><td>Potential solutions</td><td>Develop an algorithm that can efficiently pinpoint the best time slots to make a replacement class by using availability time of students and lecturer.</td></tr>
</table>

Sketch (framework, flow, interface)	

Project Proposal Refinement (week 11, submission date: 7 June 2023)

<table>
<tr><td>Group Name</td><td>TuNa</td></tr>
<tr><td>Members</td><td>
<table>
<tr><th>Name</th><th>Role</th></tr>
<tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>209861@student.upm.edu.my</td></tr>
<tr><td>MUHAMMAD NAJWAN BIN NIZARIMI</td><td>209897@student.upm.edu.my</td></tr>
</table>
</td></tr>
<tr><td>Problem statement</td><td>Dr. Hafeez, a lecturer at Universiti Putra Malaysia, faces challenges in rescheduling postponed lectures due to his busy schedule and public holidays, leading to inconsistent student attendance and potential syllabus gaps before final exams. He needs an efficient, accurate, and convenient solution to identify optimal time slots for replacement classes by analysing his and his students' schedules, ensuring all lectures are completed and students are well-prepared for exams.</td></tr>
<tr><td>Objectives</td><td>● Ensure all missed lectures are rescheduled effectively.
● Maximise student attendance.
● Ensure complete syllabus coverage.
● Ensure students are well-prepared for their final exams.</td></tr>
<tr><td>Expected output</td><td>A list of optimal time slots for replacement classes without conflicts</td></tr>
<tr><td>Problem scenario description</td><td>● Dr Hafeez is a UPM lecturer.</td></tr>
</table>

	• He always postponed his classes and lectures due to his busy schedule and many public holidays. • This has led to inconsistent student attendance and risk of important topics being left uncovered before the final exams. • He wants a solution to identify the most optimal time slots for replacement classes considering both his and his students' schedules. • The goal is to reschedule all missed lectures effectively, maximise student attendance, ensure complete syllabus coverage.
Why it is important	• To maximise student attendance • To ensure complete syllabus coverage
Problem specification	
Potential solutions	Develop an algorithm that can efficiently pinpoint the best time slots to make a replacement class by using availability time of students and lecturer.
Sketch (framework, flow, interface)	

Methodology	

Milestone	Time
1. Establish scenario refinement	Week 10 (29/6/2024)
1. Find many example of solutions and 2. Find a suitable algorithm for our project. 3. Determine how the chosen solutions and the problem example are connected to our problem scenario and the project's goal.	Week 11 (6/6/2024)
1. Develop the algorithm 2. Edit the algorithm of the chosen problem 3. Complete the algorithm	Week 12 (12/6/2024)
1. Conduct analysis of time complexity for the implementation 2. Examine the algorithm used in the code carefully to determine its effectiveness.	Week 13 (20/6/2024)
1. Prepare documents and presentation slides	Week 14 (24/6/2024)
1. Prepare for presentation 2. Deliver the presentation	Week 14 (26/6/2024)

Project Progress (Week 10 – Week 14)

<table>
<tr><td>Milestone 1</td><td>Establish scenario (planning)</td></tr>
<tr><td>Date (week)</td><td>Week 10</td></tr>
<tr><td>Description/ sketch</td><td>We begin by identifying the scenario, which involves allocating appropriate time slots for a replacement class. We establish the importance, objectives, and problem statement.</td></tr>
<tr><td>Role</td><td>
<table>
<tr><td>Member 1</td><td>Member 2</td></tr>
<tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Milestone 2</td><td>Find many example of solutions and

Find a suitable algorithm for our project.

Determine how the chosen solutions and the problem example are connected to our problem scenario and the project's goal.</td></tr>
<tr><td>Date (Wk)</td><td>Week 11</td></tr>
<tr><td>Description/ sketch</td><td>We acknowledge the strengths and weaknesses of multiple algorithms to determine the most suitable one for the scenario. From five algorithms, we chose two to compare and concluded that the Dynamic Programming Algorithm is the best solution for solving the problem based on the comparison.</td></tr>
<tr><td>Role</td><td><table><tr><td>Member 1</td><td>Member 2</td></tr><tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr></table></td></tr>
</table>

<table>
<tr><td>Milestone 3</td><td>Develop the algorithm

Edit the algorithm of the chosen problem

Complete the algorithm</td></tr>
<tr><td>Date (Wk)</td><td>Week 12</td></tr>
<tr><td>Description/ sketch</td><td>Based on the chosen algorithm from the previous step, we developed the algorithm and its pseudocode to use during the implementation phase.</td></tr>
<tr><td>Role</td><td><table><tr><td>Member 1</td><td>Member 2</td></tr><tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr></table></td></tr>
</table>

<table>
<tr><td>Milestone 4</td><td>Conduct analysis of time complexity for the implementation

Examine the algorithm used in the code carefully to determine its effectiveness.</td></tr>
<tr><td>Date (Wk)</td><td>Week 13</td></tr>
<tr><td>Description/ sketch</td><td>The algorithm was then analysed based on its time complexity, space complexity, asymptotic behaviour, and recurrence relations to ensure its correctness. The algorithm was then implemented in code to determine its effectiveness in solving the problem.</td></tr>
<tr><td>Role</td><td><table><tr><td>Member 1</td><td>Member 2</td></tr><tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr></table></td></tr>
</table>

<table>
<tr><td>Milestone 5</td><td>Prepare documents and presentation slides</td></tr>
<tr><td>Date (Wk)</td><td>Week 14</td></tr>
<tr><td>Description/ sketch</td><td>All the processes and activities from Milestone 1 to Milestone 5 are documented in the reports, and these reports are included in the presentation slides.</td></tr>
<tr><td>Role</td><td>
<table>
<tr><td>Member 1</td><td>Member 2</td></tr>
<tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Milestone 6</td><td>Prepare for presentation

Deliver the presentation</td></tr>
<tr><td>Date (Wk)</td><td>Week 14</td></tr>
<tr><td>Description/ sketch</td><td>The members are prepared to deliver the presentation in class. All related documents, slides, and code have been uploaded to GitHub for the lecturer to review.</td></tr>
<tr><td>Role</td><td><table>
<tr><td>Member 1</td><td>Member 2</td></tr>
<tr><td>TUAN NURHUSMIZA FAZEHAH BINTI TUAN MAT ZAWAI</td><td>MUHAMMAD NAJWAN BIN NIZARIMI</td></tr>
</table></td></tr>
</table>