COMP1828	**Advanced Algorithms and Data Structures**	**Faculty Header ID:**	**Contribution: 50% of course**
Module Leader: Dr IK SOO LIM	**Designing, developing & testing a journey planner**		**Deadline Date: Mon 11.30pm 18/Nov/2024, UK (GMT)**
This coursework should take an average student who is up-to-date with tutorial work approximately 30 hours Feedback and grades are normally made available within 3 calendar weeks of the coursework deadline (not counting holidays)			
Learning Outcomes: 1 Select and employ data structures and algorithms appropriate to a variety of problems. 2 Formulate models using appropriate algorithms and data structures. 3 Obtain programmatic solutions using appropriate software, including a high level programming language. 4 Describe and discuss the efficiency, complexity, accuracy and limitations of algorithms.			

Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

Coursework Submission Requirements

- An electronic copy of your work for this coursework must be fully uploaded on or before the deadline date using the link on the coursework Moodle page for COMP1828.
- For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). For mathematical notation, you can use MS Word equation tools

https://support.microsoft.com/en-us/office/write-an-equation-or-formula-4f799df7-4ca4-4670-afd3-6135768b01d0 .

- For this coursework you must also upload the source code and any additional supporting work as a single **ZIP** file.
- There are limits on the file size (see the relevant course Moodle page).
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- All coursework must be submitted as above. Under no circumstances can they be accepted in any other form by the academic staff.

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

- **Grading Criteria**

Each member must make a significant individual contribution to the technical development work as well as contributing to the overall team effort. The allocation of marks will reflect the quality of the work produced and **will be based on the breakdown provided by the team**. For example, in a team of three members A, B, and C if the breakdown provided by the team is A=100%, B=100%, and C=50%. Then for an overall project mark of 60% this will mean that A is awarded 60%, B is awarded 60%, and C is awarded 30%. If the team do not provide any breakdown or if any nonsensible breakdown is provided by the team, then the marker will allocate the same mark to each team member.

Criteria for Assessment	80-100	70-79	60-69	50-59	40-49	30-39	0-29
Content, knowledge and understanding	Demonstrates exceptional systematic understanding of problem solving, computer programming and algorithmic performance. There is exceptional evidence of engagement with all key elements.	Demonstrates an excellent systematic understanding of problem solving, computer programming and algorithmic performance. There is also excellent evidence of engagement with all key elements.	There is a very good systematic understanding of problem solving, computer programming and algorithmic performance. There is also some very good evidence of engagement with all key elements.	Has demonstrated a good understanding problem solving, computer programming and algorithmic performance. There is also some good evidence of engagement with most key elements with some omission of detail.	Has demonstrated a satisfactory level of understanding of problem solving, computer programming and algorithmic performance. There are a few notable omissions and there is limited evidence of engagement with all key elements. Overall a satisfactory attempt at this criteria.	A poor understanding of one or more of the following - problem solving, computer programming and algorithmic performance. There is insufficient evidence of engagement with the key elements. Overall an unsatisfactory attempt.	Little or no understanding of one or more of the following - problem solving, computer programming and algorithmic performance. There is very little evidence of engagement with the key elements. Overall a very unsatisfactory attempt.

Cognitive/Intellectual Skills	Demonstrates exceptional use of a critical analysis of information leading to the proposal of a robust and detailed solution. There is exceptional evidence of reflection that identifies the strengths and weakness of the approaches undertaken.	Demonstrates an excellent use of a critical analysis of information leading to the proposal of a robust and detailed solution. There is also excellent evidence of reflection and judgement based on the interpretation of the results obtained.	Demonstrates a very good use of a critical analysis of information leading to the proposal of a detailed solution. There is also some very good evidence of reflection and judgement based on the interpretation of the results obtained.	Demonstrates some good critical analysis of information leading to the proposal of a detailed solution. There are some exposed weaknesses of cognitive skills. There is also some good evidence of reflection and judgement based on the interpretation of the results obtained.	Has shown some satisfactory level of critical analysis of information. There is evidence of reflection and judgement based on the interpretation of the results obtained at a threshold pass level.	Has shown little use of techniques to undertake a critical analysis of information. The reflection and judgement based on the interpretation of results is weak and lacks detail.	Has shown little or no use of techniques to undertake a critical analysis of information. The reflection and judgement based on the interpretation of results is very weak and lacks detail.
Communication, Organisation and Presentation **Graduate Employability and Application of Skills**	Demonstrates exceptional use of argument and language which effectively communicates information to the target audience. The structure and flow of the report is clear and of an exceptional quality. There is exceptional evidence of the qualities of transferrable skills necessary for employment that required personal judgement and successful experimentation.	Demonstrates excellent use of argument and language which effectively communicates information to the target audience. The structure and flow of the report is clear and of an excellent quality. There is excellent evidence of the qualities of transferrable skills necessary for employment that required personal judgement and successful experimentation.	Demonstrates a very good use of argument and language which effectively communicates information to the target audience. The structure and flow of the report is clear and overall is very good. There is also very good evidence of the qualities of transferrable skills necessary for employment that required personal judgement and mostly successful experimentation.	There is good use of argument and language which communicates information to the target audience. The structure and flow of the report is mostly coherent and overall is good. There is also some good evidence of the qualities of transferrable skills necessary for employment.	The use of argument and language which communicates information to the target audience is mostly acceptable with some shortcomings in the grammar. The structure and flow of the report is barely acceptable with some presentation issues. There is also some evidence of the qualities of transferrable skills necessary for employment.	The use of argument and language which communicates information to the target audience is mostly at a substandard level. The structure and flow of the report is unacceptable with some presentation issues. There may also be little evidence of the qualities of transferrable skills necessary for employment.	The use of argument and language which communicates information to the target audience is at a substandard level. The structure and flow of the report is unacceptable with significant presentation issues. There may also be little/no evidence of the qualities of transferrable skills necessary for employment.

Referencing, sourcing, acknowledging and coverage	The exceptional use of appropriate references reflects clear and detailed understanding of the referenced works and its contents from a variety of sources.	The excellent use of appropriate references reflects clear and detailed understanding of the referenced works and its contents referenced works.	The use of references reflects a very good understanding of the cited work and its contents. Some references may not be the most recent.	The use of references reflects a good understanding of the cited work and its contents. Some references may not be the most recent or are taken from a narrow range of sources.	The use of references reflects a satisfactory understanding of the cited work and its contents. Some references may not be appropriate or the most recent or are taken from a narrow range of sources.	The use of references reflects a poor understanding of the cited work and its contents. The references may not be sufficient or appropriate or the most recent or are taken from a narrow range of sources.	Little or no cited work. The references may not be appropriate or the most recent.

Designing, developing and testing solutions for the London Underground system

Group work:

This coursework requires collaborative group efforts, and effective communication among group members to be successful. The demonstration of teamwork accounts for up to 20% of the overall coursework grade. This portion of the grade can be achieved relatively easy, regardless of the actual coursework completion. Securing this 20% could be crucial for students on the threshold of passing.

Documentation of weekly communications between group members is required, including:

1. Weekly email exchanges
2. A record of a weekly Teams meeting
3. Details of each member's cumulative contribution percentage

If a member's cumulative contribution credit reaches 100%, that member's individual grade will match the group's overall grade. Members with contributions under 100% will receive a proportional percentage of the group grade.

Any disagreements concerning contribution percentages or other issues should be resolved within the group. The resolution process may be included in the final report if necessary.

Further instructions and guidelines will be provided in due course.

Data:

The London Underground comprises an extensive network of stations across Greater London. A standard map of this system is available in PDF format from the Transport for London (TfL) website at https://tfl.gov.uk/maps/track/tube . Accompanying this is an Excel spreadsheet titled "London Underground Data.xlsx", which contains the journey times in minutes between adjacent stations, based on previously collected data. Please note:

1. While this data set may contain errors or omissions, you should assume it to be correct for the purposes of this assignment.
2. The provided times do not include waiting periods at stations, or the time taken for passengers to board or alight from trains.

Mandatory Use of Library Code:

• For specific subtasks outlined later in the coursework specification, you must use the Python library code available at https://mitp-content-server.mit.edu/books/content/sectbyfn/books_pres_0/11599/clrsPython.zip .
• For example, if your application code requires a binary search algorithm, you must use or call the binary search function from this library code only.
• This library is structured according to the chapter numbers from one of the required course textbooks, "Introduction to Algorithms" (4th edition). You can easily navigate and identify code for specific algorithms within the library using the book content, whose online copy is available from the library (https://ebookcentral.proquest.com/lib/gre/detail.action?docID=6925615) or its table of contents (https://mitp-content-server.mit.edu/books/content/sectbyfn/books_pres_0/11599/4e_toc.pdf), which lists chapters and sections.
• For these subtasks, using any other library code, including code you've written yourself, is strictly prohibited.
• Please note: Failure to comply with this requirement may result in your maximum score being limited to 50%.
• This mandatory use is a result of the School's directive to limit potential AI misuse and other challenges in coursework design. Without this requirement, the coursework specification would not be approved during the moderation process.
• In professional software development, using pre-existing libraries for data structures and algorithms is common practice, often resulting in faster and more reliable outcomes. Your focus should be on selecting the most appropriate data structures and algorithms for the project at hand.

TASKS

***TASK* 1:**

Journey Planner Based on Minutes

Your group is tasked with developing a Python programme for the London Underground tube system's route planner. It should determine the shortest journey duration in minutes between specified starting and destination stations.

Given a pair of starting (x) and destination (y) stations, your programme should provide:

1. A detailed list of stations indicating the journey from x to y.
2. The total duration of the journey **in minutes**.

(1a)
Manual versus code-based execution of the same algorithm:

[For this specific subtask, use the previously mentioned Python library by calling its functions within your application code.] [It is advisable to copy the library code from each chapter folder into a single directory to facilitate easier searching of dependent files. For example, you may need to use library code for the binary search tree, which might depend on or require code from another chapter folder. Having all the library code files in the same directory, sorted by name, may simplify the process of locating necessary files.]

- Select an appropriate data structure and algorithm for the task. Consider choosing from those covered in the lectures.

- Create a simple (and artificial) data set representing a tube network with only a few stations (e.g., 5 stations named A, B, C, D, and E) and journey durations **in minutes** between adjacent stations.

- Manually apply the chosen algorithm to this simple data set. Choose a pair of stations and determine the shortest path in terms of journey duration **in minutes**, executing the algorithm using pen and paper.

- Utilise the corresponding library code to complete the Python implementation for finding the shortest path and total duration. Run the code with the simple data created above.

- Verify that the manual computation and the code-based execution of the same algorithm produce identical outcomes for the given pair of stations. Explain any discrepancies if they occur.

- Repeat this comparison between the manual and code-based approaches for various pairs of stations in the data set created above.

(1b)
Empirical measurement of time complexity:

- Develop or source Python code to generate an artificial tube network data set with *n* stations, including journey durations **in minutes** between adjacent stations. If you have sourced an existing code, provide a proper reference.

- For a given tube network of *n* stations, employ your Python code from subtask **1a** to determine the path and journey duration **in minutes** between station pairs. Measure the execution time in milliseconds or any other appropriate time unit.

- Calculate the average execution time by repeating the measurement for various station pairs within the *n*-station network.

- Perform this average execution time calculation for networks of different sizes: *n* = 100, 200, ..., 900, and 1000.

- Plot a graph with the average execution time on the vertical axis and network size *n* on the horizontal axis. Compare this empirical graph with the algorithm's theoretical time complexity (in Big-O notation) to assess their alignment. Explain any discrepancies observed.

***TASK* 2:**

Journey Planner Based on Number of Stops

Your group is tasked with repeating Task **1**, with one key difference: instead of measuring journey duration in minutes, you will now calculate it based **on the number of stops** (or stations) between the specified starting and destination points.

(2a)
Manual versus Code-Based Execution of the Algorithm:

- Replicate subtask **1a**, but use the number of stops as the measure of journey duration. Utilise the same network data as in subtask **1a**, adjusting only the journey duration between adjacent stations to reflect the number of stops. This should involve minimal changes to the data whilst using the same algorithm.

- Select a pair of stations and compare:
    - The shortest path based on journey time **in minutes** (from subtask **1a**).
    - The shortest path based **on the number of stops** (from this task).
    - Analyse whether these paths are identical or different.

(2b)
Empirical measurement of time complexity:

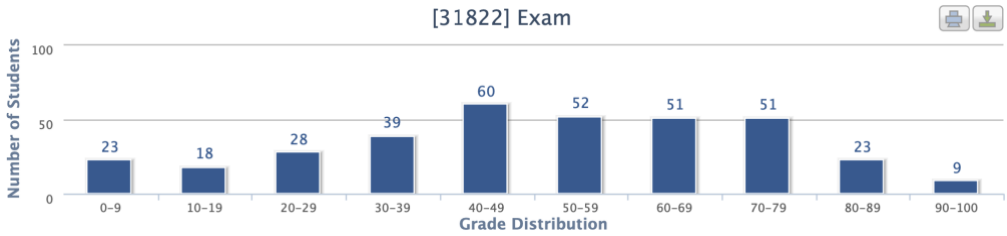
- Repeat subtask **1b**, but instead of using journey duration in minutes, **use the number of stops**. Perform this for n = 1100, 1200, ..., 1900, and 2000.

***TASK* 3:**

Histograms of Journey Duration

Based on the work completed for TASKs **1** and **2**, your group now has the tools and capabilities to determine journey durations (either in minutes or number of stops) between any pair of starting and destination stations in the London Underground network.

One application of these tools is to generate a histogram of journey durations for the entire Underground network. A histogram summarises data by displaying its distribution across a range, providing more insights than a simple average value. The horizontal axis represents ranges of data values, while the vertical axis shows the frequency of each range. For example, a histogram of exam marks for a module in a particular year would show the distribution of those marks.



(3a)
Journey Duration Histograms and Longest Path (in Minutes)

- Import the London Underground network data from the provided spreadsheet ('London Underground Data.xlsx') into your Python code. You may use any method for this, including manual input or automated import using an application or code from external sources. Please specify your chosen method of data importation.

- Utilising the code you developed with the mandatory library code for Task **1**, calculate the journey duration **in minutes** for every possible pair of stations. If the London Underground network comprises *n* stations, you must compute at least *n*(n-1)/2* journey duration values, as there are *n*(n-1)/2* pairs of starting and destination stations.

    - For example, in a network of 100 stations, each station has 99 potential destinations, necessitating 99 journey duration calculations per station. This results in a total of 9,900 (=100*99) journey durations. You may choose to exclude duplicate journeys, as the duration from station 'A' to 'B' is identical to that from 'B' to 'A'. Thus, 4,950 (=100*99/2) unique journey durations would suffice for the histogram. For simplicity, you may retain the duplicates and use all 9,900 durations. Please indicate whether you have removed or retained duplicates.

    - You may use any method to plot the histogram, either through your own coding or by using an application or code from external sources. Please specify your chosen method for histogram plotting.

- Identify the longest journey duration **in minutes** and detail its path (i.e., starting station, destination station, and intermediate stops).

(3b)
Journey Duration Histograms and Longest Path (by Number of Stops)

- Repeat the process from subtask **3a**, but measure journey durations **by the number of stops**.
    - Create a histogram of journey durations based **on the number of stop**s.
    - Identify the longest journey in terms of the number of stops and detail its path.

***TASK* 4:**

(4a)

[For this specific subtask, use the previously mentioned Python library by calling its functions within your application code.] [It is advisable to copy the library code from each chapter folder into a single directory to facilitate easier searching of dependent files. For example, you may need to use library code for the binary search tree, which might depend on or require code from another chapter folder. Having all the library code files in the same directory, sorted by name, may simplify the process of locating necessary files.]

- Imagine the government is considering closing as many line sections as possible between adjacent stations in the London Underground system. However, the closure must adhere to the following requirements:
    - Travel between any two stations must remain possible. For example, even if the line section between adjacent stations Piccadilly Circus and Green Park is closed, journeys between any pair of stations should still be feasible (potentially using a different route).
    - Note that we are not closing any stations. We are only considering closing line sections between adjacent stations.

- Select an appropriate algorithm for this task and use the corresponding library code. List the affected routes by naming the adjacent stations of each closed line section; for example, if the line section between adjacent stations Piccadilly Circus and Green Park is to be closed (but journeys between any two stations would still be possible), specify it as "Piccadilly Circus -- Green Park".

(4b)

- Repeat the process from task **3a** using the reduced network from task **4a**, where certain line sections have been closed.
    - Create a histogram of journey durations **in minutes** for the reduced London Underground system.
    - Determine the longest path **in minutes** for the reduced London Underground system.

- Compare the histogram created before the line section closures (as in subtask **3a**) with the histogram after the closures (as in subtask **4b**). By contrasting these histograms, draw specific insights, such as key differences between them.

- Compare the longest path and its journey duration before the line section closures (as in subtask **3a**) with the longest path and its journey duration after the closures (as in subtask **4b**). Analyse the differences and draw insights from this comparison.

DELIVERABLES

Deliverable 1: PDF Report

- Use the provided template: GroupX_ID1_ID2_ID3_ID4_ID5.doc.

- Replace 'X' in the filename with your group number and 'ID1', 'ID2', etc. with your group members' student ID numbers.

- Save or export the report as a PDF before submission. This PDF will be your main coursework document.

The report should adhere to the structure of the supplied template. Refer to the template for specific content requirements. The marking scheme is as follows:

- Task 1: 20 marks
- Task 2: 20 marks
- Task 3: 20 marks
- Task 4: 20 marks
- Weekly progress journal: 20 marks

Total: 100 marks

The weekly progress journal should include dated entries, evidence of regular group communication, and other relevant details; see the report template

Properly cite any content in your report that is not your own original work. A reference list alone is insufficient; you must clearly indicate within the text where and how each reference is used. **Failure to properly cite sources may result in an academic misconduct investigation.**

Additional coursework guidance may be provided separately.

Deliverable 2: Well-Commented Python Source Code

- Please upload the fully functional source code as a ZIP file.
- Ensure the code is well-commented and self-contained, meaning it should run without any additional downloads.
- After unzipping, executing the main code files should produce the outputs mentioned in the report.
- **The ZIP file for Deliverable 2 should be uploaded separately from the report.**
- Note: **Failing to submit the source code might limit your maximum score to 50%**.

Your solution should be crafted using the Python language. Where the usage of the library code is not mandatory, you are free to use supplementary resources. However, always credit any resource not created by your team. **Failing to cite external materials might lead to an academic misconduct investigation.**

It's highly recommended to start this coursework promptly once it is accessible. Should any instructions be unclear, reach out to the Module Leader. You can use email, visit during office or lab hours, or schedule an appointment at your earliest convenience.