# 55-508226 Algorithms and Data Structures

# Project 4 (25%)

# (Individual Work)

The objective of this project is to enable you to design and test a new algorithm for the problem. In this context, you are required to carry out the following tasks:

1. Comprehend and familiarise yourself with the example program provided for the FIFO elevator algorithm.
2. Design a new algorithm for the problem.
3. Run empirical experiments to test the efficiency of your algorithm in comparison to the FIFO elevator algorithm.
4. Analyse results and report your findings in the portfolio.

**Note**:

* The deadline for completion is the same as the submission date of your portfolio. However, you are advised to complete this project by 3 Dec 2021 (i.e., Week 10) to avoid increasing your workload as other projects will be given out throughout the semester.
* Please read and digest this document line by line.
* This project requires self-research about the subject and discussions with tutors. Should you have references for the work, **cite** them in the report.

## Background

Elevator scheduling is used in managing elevators operation in buildings. The main objective of the scheduling is to make sure the elevator visits each level request at the least waiting time. There are numbers of algorithms used in elevator operations. The most common one is First In First Out (FIFO) method.

Shape

Description automatically generated with medium confidence

This is an example when there are three level requests.

Shape, logo, arrow

Description automatically generated

Each level is visited in the FIFO order. The elevator will come back to level 3 when there is no level request.

## Design Your Own Algorithm

In this project you are required to come out with your own (**ONE**) algorithm for the problem. There is no specific algorithm you must use for the work as long as your proposed algorithm outperforms FIFO. The measurement metric to be used is **time in seconds** that a single elevator can complete all level requests. For example,

N = 5. You will have 5 level requests.

Sample = 3. You will have 3 sets of N (different level requests each)

N1 = {3, 4, 6, 8, 9} 🡺 56 seconds

N2 = {9, 5, 3, 6, 3} 🡺 50 seconds

N3 = {2, 1, 3, 6, 3} 🡺 48 seconds

Get the time in seconds to complete these tasks. Record them and get the **average** and **min** values.

Test your algorithm and run empirical experiments for both methods to see the algorithms performance. Your work should be able to satisfy questions/tasks in the Question section.

## Questions

1. Demonstrate in writing and/or diagram (or video if necessary) about your elevator algorithm.
2. Present your algorithm in ADL.
3. Write a workable program for your designed algorithm and test the program.
4. Run empirical experiments for both algorithms using sample datasets which are to be designed and produced by yourself. The experiments should have the following set/strategy.

*N* = number of level requests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *N* | FIFO | | Algo\_Zairul | |
| Mean | Min | Mean | Min |
| 5 |  |  |  |  |
| 10 |  |  |  |  |
| 20 |  |  |  |  |
| 25 |  |  |  |  |
| 30 |  |  |  |  |
| 40 |  |  |  |  |
| 50 |  |  |  |  |

Mean and Min (minimum) values are in seconds.

Each *N* experiment should have 10 samples of the experiments to get consistency in results.

1. Write the final results in your portfolio report. Based on your empirical experiment observation, discuss and summarise your work about the algorithm. A video recording may be necessary to present the work.

**(All questions 1-5 are to be evident in your portfolio)**

# **Appendix**

## FIFO Code

<https://github.com/zairulmazwan/Elevator.git>

This Github repository consists of three classes of java files. The Elevator.java is the example of the FIFO algorithm. You may run the program and input a few level requests while the elevator is running as it uses threading. The elevator keeps visiting the level so long as there is a request in the list. You can input -1 to quit from the program. This program is not nicely presented on the console and has limited functions. But you can improve it i.e,

* the elevator returns to the default level
* create a function that receive a txt file that consists of the level requests (*N*)
* capture the time stamp to get the duration (in seconds) for all requests complete
* write results into a txt file for analysis

Whilst another 2 files are helper files for your new algorithm **ONLY** if your designed algorithm needs it. It is **NOT** compulsory for this work to make use of this helper files.

Sample outputs of the FIFO Algorithm.

Text

Description automatically generated

Grading Criteria

This work will be graded using the criteria given below.

| **Criterion** | **Marks** | **Artefacts (for each project unless otherwise stated in Notes)** | **What you need to do to get a mark in the first-class range** | **What you need to do to get a pass mark** |
| --- | --- | --- | --- | --- |
| An assessment of time-Space Trade-off choices | 10 | A set of data structures and corresponding data (where applicable) | You show excellent awareness of the trade-off between time and space when you make your arguments  You show excellent awareness of the notion of a data structure and its implication on achieving the required trade-off between time and space.  You are able to make intellectually acceptable arguments as to which data structure is suitable based on the time-space trade-off. | You show some awareness of the trade-off between time and space when you make your arguments  You show satisfactory awareness of the notion of a data structure. However, your choice of data structures could be better to achieve the required trade-off between time and space. |
| An assessment of ability to work out complexity of an algorithm | 10 | Control flow graph  Computational complexity expressed in terms of Big O  Control flow complexity expressed in terms of McCabe’s cyclomatic complexity measure | You show excellent awareness of the notion of frequency counts of a given algorithm  You can accurately translate frequency counts to complexity of an algorithm and represent it using the Big O notation.  You show excellent understanding of converting an algorithm to its corresponding control flow graph.  You show excellent understanding of applying the McCabe’s cyclomatic complexity measure and interpreting it in the context of software maintenance. | You show excellent awareness of the notion of frequency counts of a given algorithm. However, their translation to complexity of an algorithm and representation using the Big O notation is not consistently accurate.  You show satisfactory understanding of converting an algorithm to its corresponding control flow graph. There may be some minor errors, but the application of McCabe’s cyclomatic measure and your interpretation of its outcome is accurate. |
| An assessment of understanding of algorithm design | 20 | A set of algorithms expressed in ADL or any format | You are able explain your problem-solving strategy accurately and clearly.  Your problem-solving strategy is effective.  There is a very close mapping between your strategy and your algorithm.  Your algorithms are correct with respect to the specification, i.e., it accomplishes its task unambiguously and accurately.  Your algorithms are accurate in the way they manipulate data stored in your chosen data structures. | Your explanation of your problem-solving strategy lacks clarity.  Your problem-solving strategy is satisfactory but could be more effective.  Your algorithms are correct with respect to the specification, i.e., it accomplishes its task unambiguously and accurately.  Your algorithms are accurate in the way they manipulate data stored in your chosen data structures. |
| Results Analysis and Discussion | 30 | Tabulate experiment results  Analyse statistical values of the results (if applicable)  Elaborate the results and conclude the finding  Summarise the results for the problem | Present experiment results in tabular format so that empirical analysis can be carried out to justify your algorithms/experiments.  Evidence of good results are produced and comparable to the metric measurement used for the problem.  Critically summarise the result section for findings.  Use graphs or charts to visualise results to support findings.  Improvement to the current experiment is discussed in the report. | Results analysis section is required for projects that entail experiments for problem solving.  Result for each experiment is presented in the report.  Summarise the results for the problem. |
| An assessment of data structure design | 20 | A set of data structures expressed in terms of diagrams and sample data sets (where applicable) | Your choice of data structures is accurate.  You are able to produce a number of alternative data structures by either using data provided to you or by analysing given problem domains.  You are able to provide example data sets to illustrate how data may be stored using your chosen data structures (where appropriate). | Your choice of data structures is mostly accurate.  Alternative data structures are either not considered or not considered consistently.  You are able to provide example data sets to illustrate how data may be stored using your chosen data structures (where appropriate). |
| An assessment of overall software, including testing | 5 | Evidence of how a software system was developed  Evidence of software testing and evaluation of results  A software system and its codebase  A video recording of software developed  Evidence of testing (unit test) | Your software is fit for purpose.  There is a clear evidence that you followed a systematic approach to translate your algorithms to corresponding C#/Java programs.  There is clear and light-touch test specification in place for acceptance tests.  There is evidence of successful execution of the test specification and recording of its results.  Video recordings satisfactorily   * demonstrate the functionality of software. * outline any assumptions you made while developing the software | Your software is mostly fit for purpose. However, there is either no evidence of following a systematic approach to translate your algorithms to corresponding C#/Java programs, or evidence is not consistent.  There is a clear and light-touch test specification in place acceptance tests.  There is a video recording for each project, each of which satisfies the expectations laid out fairly adequately, if not completely. |
| Incorporation of formative feedback | 5 | Written evidence of how formative feedback was evaluated and acted on | Formative feedback (verbal and written) provided by the teaching team during the development of each project is clearly outlined.  There is clear evidence of the evaluation of feedback received (e.g., in the form of a “to-do-list” with priorities) and recording the action plan using a software tool, such as Trello. | Formative feedback (verbal and written) provided by the teaching team during the development of each project is documented.  There is an action plan in place to address feedback. |