# Assessment Brief

|  |  |  |
| --- | --- | --- |
| Module Leader: Mehmet Ozcan | | Level: 5 |
| Module Name: Algorithms and Data Structures | | Module Code: 55-508226 |
| Assignment Title: Portfolio | | |
| Individual / Group | Weighting: 50% | Magnitude: 4000 words |
| Submission date/time:  16th December 2021 by 3 pm | Blackboard submission Y/N Turnitin submission Y/N | Format: e.g. Word, Excel, PowerPoint, source code, digital media. |
| Planned feedback date:  21 Jan 2022 | Mode of feedback:  Written | In-module retrieval available: Yes/No |
|  | | |
| **Module Learning Outcomes**  LO1: Design algorithms in an appropriate abstract notation and characterise them and describe their attributes using appropriate metrics.  LO2: Compare and contrast similar algorithms and describe their relative merits and demerits.  LO3: Design novel algorithms.  LO4: Optimise implementations of algorithms.  LO5: Construct and manipulate the complex data structures required of many algorithms using appropriate programming languages.   * LO1 | | |

**Assessment Brief**

This portfolio represents **100% of the module**. It involves individual work. It addresses all the learning outcome of the module.

A portfolio consists of a set of projects. A project is essentially a series of tasks based on specific topics taught throughout the semester. Each project will contribute differently to overall module mark, deliver specific artefacts, test knowledge and understanding of topics as well as their application to solve computer science problems. A portfolio is an individual work. The number, nature, difficulty level and marking weight of projects will vary from year to year. A portfolio includes the following artefacts (note that not all these artefacts may be needed for each project):

1. a set of algorithms written in Algorithmic Definition Language (or ADL for short).
2. a set of data structures represented as diagrams to store data.
3. a piece of software written as a console application in either C# or Java.
4. a video presentation that showcases the capabilities of software in (2).
5. a descriptive report on any aspect of design and development, including artefacts produced for each project.
6. a short report outlining response to formative feedback.

These will be elaborated next.

## Algorithms

You are expected to express your solution to a given problem using ADL taught in the lecture programme. Your algorithm should not bias towards a particular programming language, such as C# or Java, nor should it be expressed in terms of it. Where possible, your algorithm should be described in terms of suitable abstractions, i.e., either procedural or functional. You are encouraged to consider the notion of complexity while devising an algorithm and make judicious choices if there are alternative solutions to a given problem.

## Data Structures

You are expected to define, describe, and justify data structures needed to manipulate data used in an application. You are also expected to illustrate the data structures with example data. You are encouraged to consider alternative data structures for a given problem and make judicious choices using the notion of space-speed trade off.

## Software and its Video Presentation, including testing

You are expected to submit each project, including all its components (e.g., codebase), compressed in a separate zip file (or 7z), which should be named as Project 1, Project 2, and Project 3, etc. The zip files will be uploaded to Blackboard as directed in the relevant submission point on Blackboard. We may at our discretion request a demonstration if we have difficulty in running your software.

Each project needs to be showcased in a separate video recording of up to 10 minutes. We will stop watching after the 10th minute. You are expected to upload your video files to YouTube as a non-public unlisted video and include their links in your portfolio. Your video files should be named as Project 1 Video, Project 2 Video and Project 3 Video, etc. The following format is advised:

* Introduce yourself.
* Introduce the problem.
* Demonstrate the functionality of your software using different data sets.
* Explain how you tested your software or tested your results/findings (if necessary).

Where applicable, each project should contain a light-touch test specification outlining how acceptance tests will be carried out as well as providing evidence of acceptance testing and its results.

## Descriptive Report, including artefacts

In this section you are required to elaborate on how you devised your algorithm and developed the corresponding implementation in either C# or Java. In software development, several other artefacts in addition to algorithms and data structures are also produced. **Each software project has its own requirements and deliverables. The following may be required (specific requirements will be outlined in project specifications)**:

* Description of how you transformed your algorithm into a corresponding implementation in C# or Java.
* Time-Space Trade-off of algorithms used.
* Computational complexity of algorithms used.
* Control-flow complexity of algorithms used.
* Problem-solving strategy.
* Judgement on results and personal opinions
* Analysis of results
* Tabulation of results
* Drawing a line graph for results

## Incorporation of formative feedback (typically 250 words)

For each project, you are expected to provide written evidence of how you evaluated and acted on the formative feedback you received from your tutors. Written evidence can include in what form you received feedback (e.g., written via email or verbal in a tutorial or support session) and when (e.g., a specific date).

# Assessment Criteria

|  | **FAIL** (insufficient) | | | | **THIRD** (sufficient) | | | **LOWER SECOND** (good) | | | **UPPER SECOND** (very good) | | | **FIRST**  (excellent) | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Zero | Low  Fail | Mid Fail | Marginal Fail | Low  3rd | Mid 3rd | High 3rd | Low  2.2 | Mid  2.2 | High 2.2 | Low  2.1 | Mid 2.1 | High 2.1 | Low  1st | Mid  1st | High 1st | Exceptional 1st | Perfect 1st |
| Criteria and weighting[[1]](#footnote-2) | **<19** | | **20-39** | | **40-49** | | | **50-59** | | | **60–69** | | | **70-84** | | **85+** | | |
| **C1: An assessment of understanding and use of algorithmic notation**  **Artefacts:** A set of algorithms expressed in Algorithmic Definition Language (or ADL for short). | Some attempt has been made to tackle the assessment.  The work has little or no merit. | | Algorithm is not fit for purpose even though they it may meet some of the requirements with some additional effort. | | There is evidence of credible attempt to produce an algorithm.  You show satisfactory understanding of the notion of abstraction. However, the choice of using an abstraction may not always be judicious, e.g., use of a procedural abstraction to calculate the factorial of a given natural number instead of a functional abstraction. | | | Competent work, demonstrating some understanding of the notion of abstraction.  The work as presented has some minor errors but is limited in scope. | | | Very high-quality work, demonstrating good understanding of the notion of abstraction.  The work as presented has some minor errors, and there are some obvious areas where it could be extended/improved. | | | Very high-quality work, demonstrating excellent understanding of the notion of abstraction.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent understanding of the notion of abstraction, and syntax and semantics of ADL.  It is difficult to suggest ways that the work could be improved/extended.  You show excellent understanding of the notion of procedural and functional abstractions.  You make judicious choices between a procedural abstraction and functional abstraction. | | |
| **C2:** **An assessment of time-space trade-off choices**  **Artefacts**: A set of data structures and corresponding data (where applicable) | Some attempt has been made to tackle the task.  The work has little or no merit. | | Work is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts, but this does not meet the level that was taught in class. | | 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. | | | Competent work, demonstrating reasonable knowledge and understanding.  Some analysis and appropriate skills are shown.  The work as presented has some minor errors but is limited in scope. | | | High quality work, demonstrating good knowledge, understanding and analysis.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent knowledge, understanding and analysis.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent knowledge, understanding and analysis.  There is extensive evidence of independent investigation, learning and thought. | | |
| **C3: An assessment of ability to work out complexity of an algorithm**  **Artefacts**: Control flow graph.  Computational complexity expressed in terms of Big-O.  Control flow complexity expressed in terms of McCabe’s cyclomatic complexity measure. | Some attempt has been made to tackle the task.  The work has little or no merit. | | Work produced is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts and techniques, but this does not meet the level that was taught in class. | | Work of limited quality, demonstrating some relevant knowledge and understanding.  You show some awareness of the notion of frequency counts. However, their translation to complexity of an algorithm and representation using the Big-O notation is not consistently accurate.  You show some 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. | | | Competent work, demonstrating reasonable knowledge and understanding, analysis and accuracy.  The work as presented has some minor errors but is limited in scope. | | | High-quality work, demonstrating good knowledge and understanding, analysis and accuracy.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent knowledge and understanding, analysis and accuracy.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent knowledge and understanding, analysis and accuracy.  It is difficult to suggest ways that the work could be improved/extended.  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. | | |
| **C4: An assessment of understanding of algorithm design**  **Artefacts:** A set of algorithms expressed in ADL | Some attempt has been made to tackle the task.  The work has little or no merit. | | Work produced is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts and techniques, but this does not meet the level that was taught in class. | | Work demonstrates some relevant knowledge and understanding.  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. | | | Competent work, demonstrating reasonable knowledge and understanding.  The work as presented has some minor errors but is limited in scope. | | | High-quality work, demonstrating good knowledge and understanding.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent knowledge and understanding.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent knowledge and understanding.  It is difficult to suggest ways that the work could be improved/extended.  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. | | |
| **C5: Results analysis and discussion**  **Artefacts:** Tabulated experiment results.  Analysis of statistical values of the results (if applicable).  Elaboration of the results and conclusion of the finding.  Summary the results for the problem | Some attempt has been made to tackle the task.  The work has little or no merit. | | Work produced is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts and techniques, but this does not meet the level that was taught in class. | | Work demonstrates some analysis and discussion.  Results analysis section, which entails experiments for problem solving, is present.  Result for each experiment is presented in the report.  Summary of the results for the problem is presented. | | | Competent work, demonstrating reasonable analysis and understanding.  The work as presented has some minor errors but is limited in scope. | | | High-quality work, demonstrating good analysis and understanding.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent analysis and discussion.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent analysis and discussion.  It is difficult to suggest ways that the work could be improved/extended.  Excellent presentation of experiment results in tabular format which enables empirical analysis to be carried out to justify algorithms/experiments.  Evidence of good results are produced and comparable to the metric measurement used for the problem.  Excellent critical summary of the result section for findings.  Excellent use of graphs or charts to visualise results to support findings.  Excellent discussion on how to improve current experiment. | | |
| **C6: An assessment of data structure design**  **Artefacts:** A set of data structures expressed in terms of diagrams and sample data sets (where applicable) | Some attempt has been made to tackle the task.  The work has little or no merit. | | Work produced is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts and techniques, but this does not meet the level that was taught in class. | | Choice of data structures is mostly accurate.  Alternative data structures are either not considered or not considered consistently.  Example data sets to illustrate how data may be stored using your chosen data structures (where appropriate) are presented. | | | Competent work, demonstrating reasonable analysis and understanding.  The work as presented has some minor errors but is limited in scope. | | | High-quality work, demonstrating good analysis and understanding.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent analysis and discussion, and presentation.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent knowledge and understanding, and presentation.  It is difficult to suggest ways that the work could be improved/extended.  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). | | |
| **C7: An assessment of overall software, including testing** | Some attempt has been made to tackle the task.  The work has little or no merit | | Work produced is not fit for purpose.  There may be evidence of some basic understanding of relevant concepts and techniques, but this does not meet the level that was taught in class. | | 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# or Java programs, or evidence is not consistent.  There is a clear and light-touch test specification in place for acceptance tests.  There is a video recording which satisfies the expectations laid out adequately, if not completely. | | | Competent work, demonstrating reasonable analysis and understanding.  The work as presented has some minor errors but is limited in scope. | | | High-quality work, demonstrating good analysis and understanding.  The work as presented has some very minor errors, and there are some obvious areas where it could be improved/extended. | | | Very high-quality work, demonstrating excellent analysis and discussion, accuracy, and presentation.  The work as presented is difficult to fault, but there are some obvious areas where it could be extended/improved. | | Exceptional work of the highest quality, demonstrating excellent knowledge and understanding, accuracy, and presentation.  Your software is fit for purpose.  There is clear evidence that you followed a systematic approach to translate your algorithms to corresponding C# or 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. | | |
| **C8: Incorporation of formative feedback: 5%** | Some attempt has been made to tackle the task.  The work has little or no merit. | | Some attempt was made to receive feedback, but no or little attempt was made to respond to it. | | Some attempt was made to receive feedback.  Feedback received was documented.  There is evidence of an action plan to address feedback, even though not all feedback was addressed. | | | Adequate attempt was made to receive feedback.  Feedback received was documented well.  There is evidence of an action plan to address feedback.  Some of the feedback was addressed. | | | Satisfactory attempt was made to receive feedback.  Feedback received was documented well.  There is evidence of an action plan to address feedback.  Most of the feedback was addressed. | | | Excellent attempt was made to receive feedback.  There is clear evidence of evaluation of feedback received (e.g., in the form of a “to-do-list” with priorities).  There is also convincing evidence of successful execution of the action plan. | | Excellent attempt was made to receive feedback.  There is clear evidence of 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.  There is also convincing evidence of successful execution of the action plan. | | |

# Project 1 (10%)

## Problem domain: Production of Pascal’s Triangle

The objective of this project is to enable you to apply the concept of abstraction to problem solving. You are required to do the following tasks:

1. Devise a set of algorithms for the given problem domain and represent them using the Algorithmic Definition Language (ADL).
2. Implement the algorithms you devised in (1) systematically in either C# or Java as a console application and test your implementation.
3. A video presentation that showcases the capabilities of software in (2).
4. A descriptive report on any aspect of design and development, including artefacts produced for the project.
5. Produce a short report outlining response to formative feedback.
6. Add your solutions to your portfolio using the portfolio template provided.

This project represents 10% of the total marks available in this module.

**Note that the deadline for completion of this project is the same as the submission date of your portfolio. However, you are advised to complete it by 18th October 2021 (i.e., Week 13) to avoid increasing your workload as other projects will be given out throughout the semester.**

## Binomial coeffients

"In mathematics, *binomial coefficients* are a family of positive integers that occur as coefficients in the binomial theorem."[[2]](#footnote-3)

The quantities n! / k! (n-k)! are the famous *binomial coefficients*, and they are denoted by[[3]](#footnote-4)



Some of their special values are

Description: C:\Users\cmsmo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\56C3N23Q\CodeCogsEqn.gif Description: C:\Users\cmsmo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\56C3N23Q\CodeCogsEqn (1).gif

Description: C:\Users\cmsmo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\24ONGHAA\CodeCogsEqn.gif Description: C:\Users\cmsmo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\56C3N23Q\CodeCogsEqn (2).gif

It is convenient to define Description: C:\Users\cmsmo\AppData\Local\Temp\CodeCogsEqn-1.gifto be zero if k < 0 or if k > n.

## Pascal's Triangle

In the following table, we show the values of some of the binomial coefficients Description: C:\Users\cmsmo\AppData\Local\Temp\CodeCogsEqn-1.gif. The rows of the table are thought of as labelled 'n = 0', n = 1', etc, and the entries within each row refer, successively, to k = 0, 1, 2, …. n. The table is called 'Pascal's triangle'.

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

Here are some facts about the binomial coefficients:

1. Each row of Pascal's triangle is symmetric about the middle. That is,

Description: C:\Users\cmsmo\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\919FMAQR\CodeCogsEqn.gif

1. The sum of the entries in the nth row of Pascal's triangle is 2n.
2. Each entry is equal to the sum of the two entries that are immediately above it in the triangle.

## Grading Criteria

This project will be graded using the criteria given below.

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| An assessment of understanding and use of algorithmic notation | 25 |
| An assessment of understanding of algorithm design | 40 |
| An assessment of overall software, including testing | 30 |
| Incorporation of formative feedback | 5 |

# Project 2 (15%)

## Problem domain: Algorithmic Complexity and Space-Speed Trade Off

The objective of this project is to enable you to apply the concept of algorithmic complexity to given problems and work out a compromise in a situation where a space-speed trade-off exists. You are required to do the following tasks:

1. Work out the computational complexity of an algorithm.
2. Assess time-space trade-off choices in a problem.
3. Produce a short report outlining response to formative feedback.
4. Add your solutions to your portfolio using the portfolio template provided.

This project represents 15% of the total marks available in this module.

**Note that the deadline for completion of this project is the same as the submission date of your portfolio. However, you are advised to complete it by 25th October 2021 (i.e., Week 14) to avoid increasing your workload as other projects will be given out throughout the semester.**

### Task 1

Consider the following algorithm fragment written in ADL and calculate its computational complexity with justification and represent it in Big-O notation.

**for** i **←** 1 **to** n **by** 1 **do**

**for** j **←** 1 **to** i **by** 1 **do**

**for** k **←** 1 **to** j **by** 1 **do**

x = x + 1

**end**

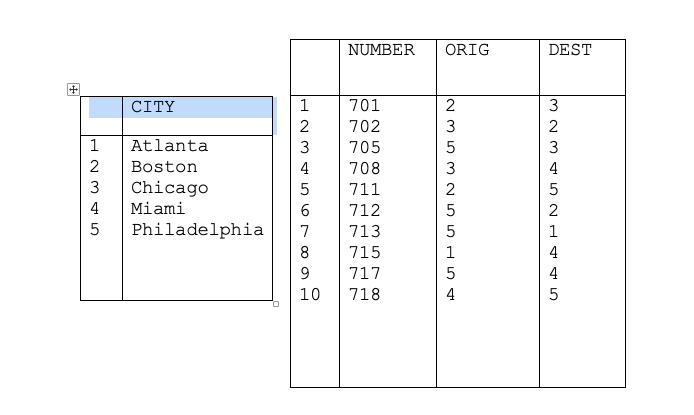
**end**

**end**

### Task 2

Consider the data shown below, which gives the different flights of an airline. Discuss different ways of storing the data to decrease the time in executing the following:

* Find the origin and destination of a flight, given the flight number.
* Given city A and city B, find whether there is a flight from A to B, and if there is, find its flight number.



## Grading Criteria

This work will be graded using the criteria given below.

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| An assessment of time-space trade-off choices | 60 |
| Ability to work out the complexity of an algorithm | 35 |
| Incorporation of formative feedback | 5 |

# Project 3 (20%)

## Problem domain: Priority Queues

The objective of this project is to enable you to apply the concepts of data structure design and implementation. You are required to do the following tasks:

1. Design data structure(s) for the given problem domain.
2. Devise a set of algorithms using the Algorithmic Definition Language (ADL) to manipulate the data structures designed in (i).
3. Implement the algorithms you devised in (2) systematically in either C# or Java as a console application and test your implementation.
4. A video presentation that showcases the capabilities of software in (2).
5. A descriptive report on any aspect of design and development, including artefacts produced for the project.
6. Produce a short report outlining response to formative feedback.
7. Add your solutions to your portfolio using the portfolio template provided.

This project represents 20% of the total marks available in this module.

**Note that the deadline for completion of this project is the same as the submission date of your portfolio. However, you are advised to complete it by 8th November 2021 (i.e., Week 16) to avoid increasing your workload as other projects will be given out throughout the semester.**

### Priority Queue

A *priority queue* is a collection of elements such that each element has been assigned a priority in which elements are deleted and processed comes from the following rules:

1. An element of higher priority is processed before any element of lower priority
2. Two elements with the same priority are processed according to the order in which they were added to the queue.

An example of the use of a priority queue is in a time-sharing operating system: programs of high priority are processed first, and programs with the same priority form a standard queue. There are various ways of maintaining a priority queue in memory. In this project, you are expected to focus on one that uses a linked list. Specifically, in this project, design and implement a priority queue for “job elements” as shown below. Note that the jobs are not in a particular order in this example. In your data structure design, you are required show the beginning and end of the priority queue.

|  |  |
| --- | --- |
| **Job Description** | **Priority** |
| EEE | 4 |
| GGG | 5 |
| CCC | 2 |
| DDD | 4 |
| BBB | 2 |
| FFF | 4 |
| AAA | 1 |

As outlined in the objectives, devise algorithms in ADL as suitable abstractions and implement them that add a job element (i.e., its description and priority), remove a job element and retrieve the priority of a job element given its description. Finally, implement yourt algorithms in either C# or Java as a console application.

This project will be graded using the criteria given below.

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| An assessment of understanding and use of algorithmic notation | 15 |
| An assessment of understanding of algorithm design | 30 |
| An assessment of data structure design | 30 |
| An assessment of overall software, including testing | 20 |
| Incorporation of formative feedback | 5 |

# Project 4 (25%)

## Problem domain: Designing an Algorithm for an Elevator in Comparison to FIFO Algorithm

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:

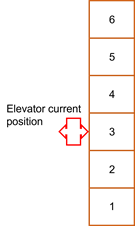
1. Comprehend and familiarise yourself with the example program provided for the FIFO elevator algorithm.
2. Design a new algorithm for the problem.
3. Run empirical experiments to test the efficiency of your algorithm in comparison to the FIFO elevator algorithm.
4. Analyse results and report your findings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

This project represents 25% of the total marks available in this module.

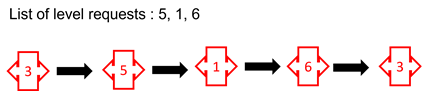
**Note that the deadline for completion of this project is the same as the submission date of your portfolio. However, you are advised to complete it by 3rd December 2021 (i.e., Week 10) to avoid increasing your workload as other projects will be given out throughout the semester.**

### 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.



This is an example when there are three level requests.



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} è 56 seconds

N2 = {9, 5, 3, 6, 3} è 50 seconds

N3 = {2, 1, 3, 6, 3} è 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.

## Grading Criteria

This project will be graded using the criteria given below.

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| An assessment of time-space trade-off choices | 10 |
| Ability to work out the complexity of an algorithm | 10 |
| An assessment of understanding of algorithm design | 20 |
| Results Analysis and Discussion | 30 |
| An assessment of data structure design | 20 |
| An assessment of overall software, including testing | 5 |
| Incorporation of formative feedback | 5 |

# Project 5 (30%)

## Problem domain: Implement Genetic Algorithm to the Scale Problem

The objective of this project is to enable you to implement Genetic Algorithm to the paradigm problem. In this context, you are required to carry out the following tasks:

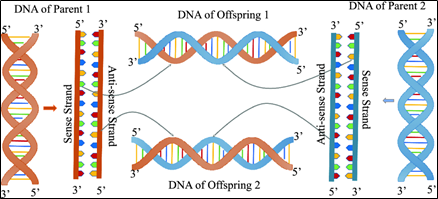
1. Design your Genetic Algorithm (GA) based on the skeleton program provided.
2. Implement your GA for the paradigm problem and run some experiments to answer the research questions outlined in the research question section.
3. Analyse results and report your findings in the portfolio.

This project represents 30% of the total marks available in this module.

**Note that the deadline for completion of this project is the same as the submission date of your portfolio.**

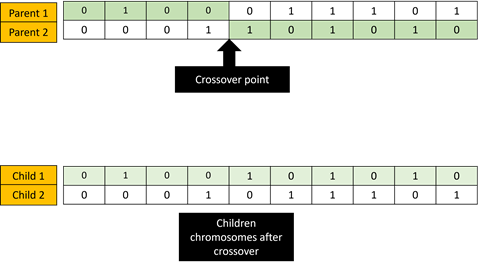
Background

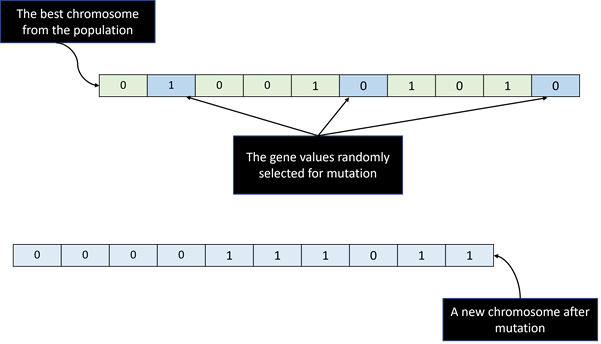
A **genetic algorithm** is a search heuristic method that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.



Five phases are considered in a genetic algorithm.

* Initial population
* Fitness function
* Selection
* Crossover
* Mutation





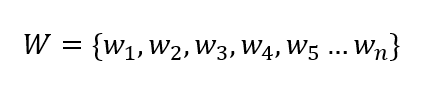
## The Scales Problem

The scales problem is one of the paradigm problems in heuristics and optimisation used to test algorithms.



Suppose you have *n* objects of various weights, and you are required to split those objects into two equally heavy piles (or as equal as possible). Refer [*Problem Solving - Balance a Scale Problem*](https://youtu.be/-n-A5qcO398)for a video explanation.

In this project, you are given with a list of random numbers as the weights.



Using Genetic Algorithm, write a program that optimises,



Where,

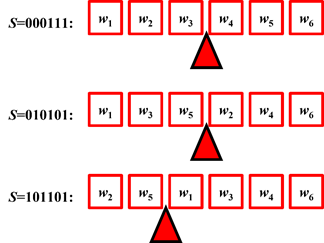
*L* is the sum of left-hand side weights

*R* is the sum of right-hand side weights

## Genetic Algorithm Representation

We represent the solution as an *n* length binary string/integer where:

* 1. A zero (0) in position *i* means that weight *i* is on the left side of the scales
  2. A one (1) in position *i* means that weight *i* is on the right side of the scales



Each solution has its fitness value that computed from the equation. Those solutions with the least fitness values will survive for the next generation. Example:

If your defined population is 10 candidates. The next generation pool will be 10 + reproduction via cross over (2 candidates) + a mutant. Thus, the next generation will have 10 best candidates from those 13.

This process iterates until the number of generation complete.

Instructions

## 

With the skeleton codes and sample dataset provided (**Appendix**), write a program to solve the problem using Genetic Algorithm. Answer the questions in the Research Question section.

Research Questions

1. Design and explain your GA algorithm. Run your algorithm for experimental analysis and demonstrate the results in a table that has maximum, minimum, and mean values of fitness. What are the optima fitness values for the weights?

1. What are the solutions (chromosomes) for question 1? Tabulate the chromosomes and their fitness values in a table. Verify the fitness using a manual calculation (use excel sheet). Provide evident in your report.
2. What are the values used for the following parameters in your experiments,

* Number of generations
* Crossover rate
* Mutation rate

What if you increase and decrease the values? Discuss.

1. Demonstrate in writing and/or pseudocode your crossover strategy for the GA.
2. Demonstrate in writing and/or pseudocode your mutation strategy for the GA. Discuss why mutation is important in GAs?

1. Run empirical experiments to find associate rules of crossover and mutation rates for another dataset of 50 (to be produced from the code provided) weights with the following fitness function,

Where,



*L* is the **product** of left-hand side weights

*R* is the **sum** of right-hand side weights

Analyse and summarise your findings in the report. Plot into graph/s to prove your findings. Discuss your findings.

A video recording to demonstrate the work is part of the marks.

## Grading Criteria

This project will be graded using the criteria given below.

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| An assessment of understanding of algorithm design | 30 |
| Results Analysis and Discussion | 40 |
| An assessment of data structure design | 20 |
| An assessment of overall software, including testing | 5 |
| Incorporation of formative feedback | 5 |

1. Not all these criteria will apply to each project. Furthermore, the weightings of criteria that are applicable to each project may vary. You need to refer to specific project specifications for distribution of marks to each criterion. [↑](#footnote-ref-2)
2. [Wikipedia](http://en.wikipedia.org/wiki/Binomial_coefficient) (<http://en.wikipedia.org/wiki/Binomial_coefficient>) [↑](#footnote-ref-3)
3. "Algorithms and Complexity", Herbert S. Wilf, Prentice-Hall, 1986, ISBN: 0-13-022054-X. [↑](#footnote-ref-4)