ECM 1414: Lift (Elevator) Control System

(Using Queues and Scheduling Algorithms)

Deadline: 3 March 2025

This assessment contributes 30% of the total module mark and is related to the following ILOs (in this order or relevance). See https://www.exeter.ac.uk/study/studyinformation/modules/info/?moduleCode=ECM1414&ay=2024/5&sys=1 for a list of all ILOs

- Demonstrate a familiarity with a range of fundamental data structures used in computing (ILO 1)
- Give rigorous specifications of algorithms and construct solutions to algorithmic tasks using pseudocode (ILO 4)
- Describe the properties of algorithms, explaining clearly their relationship to programs and to heuristics (ILO 3)
- Demonstrate enhanced practical skills in programming and problem analysis as a result of the deeper theoretical understanding gained from this module (ILO 9)

This coursework also touches on the following ILOs

- Explain the meaning of computational complexity, distinguishing space and time complexity, and evaluating the complexity of a range of different algorithms (ILO 7)
- Present and explain abstract ideas using a systematic approach (ILO 11)
- Show an awareness of the issues of termination and correctness, demonstrating these for some simple examples (ILO 5)

This is a group assessment, and you are reminded of the University's Regulations on Collaboration and Plagiarism. You must avoid plagiarism, collusion and any academic misconduct behaviours. Further details about Academic Honesty and Plagiarism can be found at https://ele.exeter.ac.uk/course/view.php?id=1957

Instructions

For this coursework, you must submit the following:

- A PDF with the specification of your implementation/coursework.
- A link to a presentation on YouTube should be included in the PDF. You can make the presentation as unlisted if you do not want people to find it. All members of the group need to participate in the presentation (present parts of it). The presentation should be 12-15 minutes long.
- The PDF should also contain a statement explaining who did what in the project
- All the source files of your implementation
- You should include a Generative AI statement (in PDF).
- You should submit one ZIP file containing the PDF and the sources. The sources should be part of a directory/folder called "sources" inside your ZIP

Further instructions

- No extension will be granted for technical difficulties
- Sending files via email/teams is not considered a submission
- It is your responsibility to make sure the submission contains everything that is required

This assessment falls under the category of **AI supported**.

- The University of Exeter is committed to the ethical and responsible use of Generative AI (GenAI) tools in teaching and learning, in line with our academic integrity policies where the direct copying of AI-generated content is included under plagiarism, misrepresentation and contract cheating under definitions and offences in TQA Manual Chapter 12.3
- Whenever you use Generative AI to assist you in the tasks. Comments in the code as well as in the specification PDF must be clearly outlined.
- The prompts used should be included as part of the specification, indicating clearly the parts (code or specification) where they were used

Introduction

Efficient lift (elevator) systems are vital for enhancing user experience and energy conservation in multi-story buildings; you may not realise it, but almost every lift you use has a control system responsible for answering and scheduling the calls. This coursework focuses on developing an intelligent, highly adaptable lift control system that responds effectively to passenger requests while optimising movement. The system is designed to work with a configurable number of floors ($N > 1$) and accommodate varying numbers of people on each floor requesting specific destinations. By leveraging data structures such as queues and priority queues and implementing scheduling algorithms like SCAN and LOOK, the project aims to minimise people's wait times regardless of the building's size or the volume of requests. This flexible approach ensures that the system can handle a variable number of floors and dynamic request patterns seamlessly. The coursework not only applies core concepts from the "Data Structures and Algorithms" module but also demonstrates practical programming skills, algorithmic thinking, and a comprehensive understanding of computational complexity. Through this project, we provide a real-world application where adaptable data structures and algorithms are key to efficiently managing diverse and changing scenarios.

Group Formation

For this group assessment, students will be automatically assigned to groups of three to five members at the beginning of the course. This random assignment reflects real-world scenarios where you often collaborate with colleagues you haven't chosen. To accommodate personal preferences and promote effective teamwork, a two-week adjustment period is provided. During this time, students can negotiate directly with peers to switch groups or join another group, as long as the group size remains within the minimum of three (3) and maximum of five (5) members. Lecturers will not be involved in this process; it is entirely up to the students to manage group changes. Any adjustments must be finalised and communicated to the module leader (Prof Ronaldo Menezes) by the end of Week 2. This process encourages the development of negotiation and collaboration skills, preparing you for professional environments where team composition may change, and adaptability is essential.

Objectives

This coursework has several objectives

- Efficiently handle requests: Your implementation needs to manage incoming requests from people waiting to go up or down
- Search for optimal algorithms: Use scheduling algorithms (like the two suggested below) to decide the operation of the lift
- Prioritise requests based on factors such as direction of travel, proximity to the call, and waiting times
- Implement a real-time simulation: you need to simulate real-time scenarios with multiple passengers requesting the lift simultaneously. You should also have real-world constraints such as capacity limits and the time taken to move between floors
- Generate charts showing the efficiency of the algorithms: you have to come up with scenarios for your simulation and run experiments showing the statistics of at least two scheduling approaches
- (Optional) Develop a GUI to visualise the simulation and lift’s movements (up to 5% extra)
- (Optional) Develop an approach where buildings have multiple lifts. Note that all the requirements above apply for this extension (simulation, results/charts, etc) (up to 5% extra)

Data Structures You’re Expected to Use

In this project, you’re expected to use (and implement) several data structures. You are not expected to use pre-defined data structures in your language of choice (Java or Python)

- Queues: used to manage floor requests and the order they arrived. You may need to have queues for requests to move up and down, and maybe consider a separation per floor to deal with limits
- Priority Queues: You should consider that some requests may have a priority due to long wait times, consider priority floors. Note that you also need to consider the fact that in real-world scenarios, not every person makes a request. You can have several people on one floor but just one request
- Heaps: Probably the best way to implement priority queues

Algorithms to Implement

You can implement other algorithms, but at least two must exist. If you implement extra algorithms, your specification must discuss why they are different and present statistics/charts about their performance. They must be improvements to the ones required below.

- SCAN algorithm. This algorithm has the lift move in one direction and only changes direction when it reaches the bottom or top floors. It serves all requests while travelling in that direction
- LOOK algorithm. Like SCAN, but the direction can be changed at any point if no more requests exist in the current direction
- Real-time algorithm. Adapt the algorithms above to handle arrival of requests in real-time. On the above you can assume a particular set of requests at the start of the simulation.

Marking Scheme

You will be marked in the following way:

- Specification Document (20%)
    - The specification should clearly outline the requirements, objectives and scope of the coursework
    - It must include detail design of your implementation
    - All assumptions and limitations should be stated
    - Provide specific specifications for SCAN, LOOK and variations (if applicable), using pseudocode
    - Explain how these algorithms were adapted for real-time handling.
    - Group collaboration statement outlining each group member's contribution reflecting a fair distribution of the work
    - The Generative AI Statement should include a detailed disclosure of any use of Generative AI. You should include prompts and where/how they were used.
    - Discuss any additional algorithms implemented (if applicable).
- Implementation (50%)
    - Data structures implementation (20%)
    - Algorithms implementation (20%)
    - Code quality and Documentation (10%)
    - Note that your code is expected to run/compile correctly. You will automatically be deducted half of the marks here if your code does not run/compile correct. Meaning that the entire section will be worth 25% (maximum)
- Simulation and Testing (20%)
    - Development of realistic scenarios for multiple passengers and requests
    - Implementation of constraints like lift capacity and travel time between floors
    - Collection of data for the generation of charts comparing the efficiency of the lift algorithms
    - Analysis of the results, discussing what factors left to a good or bad performance
    - Identification of limitations in the algorithms implemented
- Video Presentation (10%)
    - Clear explanation of project objectives, design choices, implementation and results
    - Demonstration of understanding the details of the implementation
    - All group members should participate in the video

    - Video should include a demo of the execution of the implementation
    - Clarity and professionalism.
- Extra Points
    - GUI (up to 5%)
    - Multiple Lifts (up to 5%)

Specification Structure

You can deviate of this structure slightly, but it's recommended you do not unless you talk to the lecturers and/or PTAs, and they indicate you should. Note that in each section below we suggest possible content to them, but you can expand and change according to your results/implementation.

Title Page (Title of project, module number, group members, IDs of group members

1. Introduction (Background, purpose of the project, overview of the report
2. Objectives (Request handling, algorithm implementation, real-time simulation)
3. System Overview (High-level description, key features, diagram with the architecture)
4. Specification (Data structures used, algorithms specifications, implementation details)
5. Results (Description of simulation scenarios, performance metrics, comparison of algorithms)
6. Discussion (Interpretation of the results, limitations of your system, future work).

Suggested Folder Structure for the .zip file

```
/[StudentID]
│
├── specification
│   ├── coursework_spec.pdf
│   └── additional_docs/
│
├── sources
│   ├── main.py (or main.java)
│   ├── module1.py
│   └── ...
│
├── presentation
│   └── presentation_link.txt
│
├── results
│   ├── simulation_report.pdf
│   ├── charts/
│   └── data/
│
└── README.md
```

Key points

- **Root Folder:** Named after a student ID (one student per group)
- **specification:** Contains the main PDF and related docs.

- **sources:** All source code files.
- **presentation:** Stores YouTube presentation URL or assets.
- **results:** Includes simulation report, charts, and data.
- **README.md:** Overview and navigation instructions.