|  |
| --- |
| Institutt for Informatikk  **Index ADT**  Oblig 1  Erling Heimstad Willassen  INF-1101, Vår 2025 |

Contents

[1 Introduction 1](#_Toc190446508)

[2 Technical background 1](#_Toc190446509)

[3 Design 1](#_Toc190446510)

[4 Implementation 2](#_Toc190446511)

[4.1 Classes 2](#_Toc190446512)

[4.2 Functions 2](#_Toc190446513)

[5 Result 3](#_Toc190446514)

[6 Discussion 4](#_Toc190446515)

[7 Conclusion 6](#_Toc190446516)

[8 Source 6](#_Toc190446517)

# Introduction

In this assignment students was tasked to implement a linked-list and an "Abstract Data-type" (ADT) that satifies at atleast an average time complexity of O(log n). The ADT structure should be able to store arbitary values that are searchable by a key. In this assignment, the student implemented a double linked-list and a hashmap as the requirement for linked-list and ADT, respectively.

# Technical background

The assignment required the linked-list and ADT to be written in C. The programming language C, in contrast to e.g. Python, don't have linked-list or hashmaps already as part of its library, and must be written in code for implementation. Therefore, one has to create their own structure in C to make a linked-list or hashmap.

For the assignment, the students acquire pre-codes for their implementation of ADT. In the pre-code, several code and functions are already written by the course lectures, but does not have functionality without working ADT structures. The student has to modify or implement ADT's in the pre-code for it to be compiled correctly.

After implementation of ADT, compiling and running the executable, the user can write a signel word in terminal and get the frequency of the word in a dictionary text file from the pre-code. The code works by using the word as "key" for the hashmap, and retrieves the frequency of said word from the hashmap. The linked-list is initally used map each word and its frequency before getting mapped to a hashmap for search algorithm.

# Design

# Implementation

# Discussion

- **\*\*Introduction\*\*** - Introduce the solution that you have come up with

- **\*\*Design/Methods\*\*** - Describe how and what has been

- **\*\*Results\*\*** - Describe your experiments and the results from these

- **\*\*Discussion\*\*** - Discuss your results

- **\*\*Conclusion\*\*** - Conclude your report

The most crucial part of the report is the results and discussion. As mentioned, you will need to prove through experiments that your `map` implementation has **\*\*O(log n)\*\*** time complexity, and the experimental data should be presented in a readable form.

In summation, the requirements for this assignment are:

- Implement a *\*linked list\** satisfying the `list.h` interface

- Implement a *\*map\** satisfying the `map.h` interface with a average search complexity of O(log n)

- Design and implement experiments/benchmarks and gather data about your maps performance

- Write a report and present your design, and present your experimental results.

- Argue based on your results that the implementation satisfies the requirement for average search time complexity

- Argue for what trade-offs your application makes with regards to space and time complexity

- Har en tilnærmest O(1) søke funksjon på average (litt høyere) grunnet lenketliste.

- Hashmap er rask, men tar mye plass. Må allokere minne til noder som er minimum størrelsen av array, og hvis det er kollisjonen må det bli allokert mer minne (siden det er lenketliste).

- Kunne ha laget en dynamisk array, nå er den fsat på 100000 størrelse i array

# Conclusion

# Source

[1] «Boids», *Wikipedia*. 7. februar 2023. Åpnet: 3. mars 2023. [Online]. Tilgjengelig på: https://en.wikipedia.org/w/index.php?title=Boids&oldid=1137932972

[2] «about - pygame wiki». Åpnet: 14. februar 2025. [Online]. Tilgjengelig på: https://www.pygame.org/wiki/about

[3] «Boids (Flocks, Herds, and Schools: a Distributed Behavioral Model)». Åpnet: 14. februar 2025. [Online]. Tilgjengelig på: http://www.red3d.com/cwr/boids/

[4] «Boids-algorithm». Åpnet: 14. februar 2025. [Online]. Tilgjengelig på: https://vanhunteradams.com/Pico/Animal\_Movement/Boids-algorithm.html

[5] *pyreverse: set of tools to reverse enginering Python code*. Åpnet: 14. februar 2025. [Online]. Tilgjengelig på: http://www.logilab.org/projects/pyreverse