# Searching with datastructures

## Introduction

Searching values and storing them in memory is a key concept in computer science. For optimal speed and complexity, data is stored in a lot of different data structures, for example a tree or a skiplist. Every data structure has its own advantages, some perform better on insertion, while other perform better on finding a specific value. In this document we will describe how we will compare data structures. We will start by proposing a research question and some sub-questions. Secondly we specify the problem we want to research and the scope of our project. After that we explain our experiments in terms of criteria, test data and scenarios.

## Research questions

We want our research to include multiple data structures. We want our research to compare those data structures in different scenario’s, since each data structure has its stronger and weaker points. Data structures will be compared using the duration of the different actions. Using this information, our research question is stated as:   
“Which data structure has on given actions the shortest duration?”  
This question can be specified in multiple sub-questions, all derived from the given actions. For action x we add the sub-question:  
“Which data structure has the shortest duration on x?”  
Since we also want to compare the general usability of the data structures, we will research (although not in depth) combinations of these actions. This results in a new sub-question:  
“Which data structure performs best (using duration) on interleaved actions?”

## Scope and assumptions

Since our research capacity is limited, we cannot research and compare all existing data structures for searching. Since we want to keep our research broad, we will take data structures which differ a lot from each other. We take four data structures, with different qualities. First we take the naïve approach: the list. Next we take the tree, which is another commonly used data structure for searching. Next we take the fast hashtable, also used in databases. At last we want to include the min/max-heap in our experiments, which is mainly used for extracting the min of max value, stored in the heap.

Here we specify our things.

## Criteria

## Test data

## Scenarios

Voor het opslaan van gegevens en het zoeken van die gegevens met behulp van een key zijn in het vak Datastructuren een heleboel verschillende datastructuren behandeld:

AVL-bomen

Rood-zwart-bomen

Hashtabellen met chaining

Hashtabellen met open addressing

Skip lists

Gewone Boom

Onderzoeksvraag:

Welke datastructuur heeft op gewenste acties de snelste looptijd.

Probleemomschrijving:

-Veel acties

-Veel datastructuren

-Welke wanneer

Aannames:

-Acties

-Opbouwen van datastructuur

-Zoeken van een willekeurig getal in de datastructuur

-Minimum/maximum opzoeken/verwijderen.

-Toevoegen/Verwijderen

-Datastructuren

-Min-maxheap (gecombineerd) (Sam)

-Lijst (William) evt. inclusief sortedlist in C#

-Gebalanceerde boom (kiezen) (Erik)

-Hashlijst (kiezen) (Gerben)

Criteria:

-Snelheid op:

-Opbouwen van datastructuur

-Zoeken van een willekeurig getal in de datastructuur

-Minimum/maximum opzoeken/verwijderen.

-Toevoegen/Verwijderen

Testdata:

- Grote invoer

- Kleine invoer

- Gesorteerd/ongesorteerd

Scenario's

-Voor elke actie

- Voor elke datastructuur

- voor verschillende testdata

- de snelheid