# Searching with data structures

## 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

Within our research it is our intention 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 followed:  
“Which data structure has the shortest duration on given actions?”

This question is the overall idea in our research and can be specified in multiple sub-questions. The first sub-question is focussed on specific 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 second 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 have chosen the following data structures:

- Lists

- Balanced trees

- Hash tables

- Min-max heaps

The data structures we have chosen are all fast in specific actions. We will look at how fast they are and what this means for the other actions and overall speed of the data structure.

Lists are the naïve approach when we think about search data structures. We will sort the list when creating it and keep it sorted with every action. A balanced tree is another commonly used data structure for searching because of its overall speed. Within our research we will make use of the AVL tree to represent the balanced tree. We have chosen for the AVL tree because it is similar to the red-black tree, but known to be faster on lookups. The third structure we will be using is the hash table. This structure is known to be fast with lookups, insertions and deletes. Our last data structure we want to include in our experiments is the min-max heap, which is mainly used for extracting the minimum or maximum value.

There are many actions to perform on a data structure, but we want to put our focus on the main actions that every data structure uses. Therefore we have chosen for the following actions:

- Build

- Search

- Insert

- Delete

- GetMin

- GetMax

- ExtractMin

- ExtractMax

We will shortly describe what every action will do to create a shared source of knowledge.

Build

In the build function we will set up our data structure so that we can use it for all the other actions. We give the function all the desired values and sort it in the way desired for this data structure.

Search

In the search function we will try to find the object for a given key. If the object is not present in the set of items we will return an empty object.

Insert

In the insert function we will insert the object in a way desired for the given data structure.

Delete

In the delete function we will try to delete the object for a given key and restore the objects within the data structure to the desired sequence.

GetMin

In the getMin function we will search and return the object with the lowest key.

GetMax

In the getMax function we will search and return the object with the highest key.

ExtractMin

In the extractMin function we will search and return the object with the lowest key and take it out of the set of items. We will restore the data structure to the desired sequence.

ExtractMax

In the extractMax function we will search and return the object with the highest key and take it out of the set of items. We will restore the data structure to the desired sequence.

For our research we assume that the data structures we will build are all correct so that we will receive correct output. Another assumption is that computer we will be doing our experiments on has enough memory. The last assumption we make is that our program will end in finite time.

## Criteria

In our research we will look at how every data structure performs on every action. We could split this up into 32 criteria, but this does not add to the research. Therefore we have decided to combine the data structures for every individual action, but still make the distinction in our research. This way we come up with the following criteria:

- How fast a build for every single data structure?

- How fast a search for every single data structure?

- How fast an insert for every single data structure?

- How fast a delete for every single data structure?

- How fast a getMin for every single data structure?

- How fast a getMax for every single data structure?

- How fast an extractMin for every single data structure?

- How fast an extractMax for every single data structure?

Measuring our criteria will be done by using the stopwatch in C#. We will insert test-cases and see how fast the data structure returns the desired answer.

## 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