Searching with data structures

Gerben Aalvanger
Studentnumber: 3987051
Utrecht University

Erik Visser
Studentnumber: 3470806
Utrecht University

William Kos
Studentnumber: 3933083
Utrecht University

Sam van der Wal
Studentnumber: 3962652
Utrecht University

14 februari 2015
Teacher: Peter de Waal

1 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.

2 Research question

Within our research it is our intention to include multiple data structures. We want our research to compare those data structures in different scenarios, since each data structure has its stronger and weaker points. Data structures will be compared by the duration of the different actions. Using this information, our research question becomes: “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?”

3 Scope & 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 nave 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 data structure we will be using is the hash table. This structure is known to be fast with lookups, insertions and deletes, but will not be fast on extracting min/max values. 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, but has difficulties with deletion and searching of other values.

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

For our research we assume that the data structures we will build are all correct so that we receive correct output. Another assumption is that the 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.

4 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 is a build for every single data structure?
- How fast is a search for every single data structure?
- How fast is an insert for every single data structure?
- How fast is a delete for every single data structure?
- How fast is a getMin for every single data structure?
- How fast is a getMax for every single data structure?
- How fast is an extractMin for every single data structure?
- How fast is 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.

5 Test data

A lot of different inputs are possible, but since research capacity (in time and researchers) is limited, our research will be limited to a certain amount of inputs. At first the assumption is made that all input is unique, no double values (or keys) are stored in the data structures. This will simplify the implementation of the data structures, but will not influence the running time a lot.

For each action evaluated, we will define the different types of data we test on.

Building the data structure
Since the data structure is generated from a list, the form of the list should be defined. A list has two useful properties: the size (length) of the list and the order of the list (ordered, reverse ordered or random).

Searching
Searching in the data structure has two properties: the key where one is looking for and the amount of keys stored in the data structure. The searched key is between the minimum and maximum key used. Since we are interested in the difference between explicitly searching for the maximum/minimum and implicitly searching for the minimum/maximum, we will search for the minimum, maximum and a random value. We will use a data structure with a high amount of elements.

Insertion
Inserting a value has two properties: the key to insert and the amount of keys stored in the data structure. The inserted key is between the minimum and maximum integer value. Since we are interested in minimum and maximum values, we want to insert both a value higher than the maximum and a value lower than the currently stored minimum to the data structure. Besides this values, we also want to add random values.

Deletion
Deleting a value is influenced by two properties: the key to delete and the amount of keys stored in the data structure. The deleted key is between the minimum and maximum integer value. Since we are interested in minimum and maximum values, we want to delete both the minimum and maximum value. Besides this values, we also want to delete random values.

Other actions
The other actions (Get/Extract Min/Max) are only influenced by the amount of elements in the data structure. We will use a data structure with a high amount of elements.

6 Scenarios

Our research consists of three different stages. In the first stage we will start implementing the algorithms we will be testing in the experiments and make sure that these algorithms contain no errors. This first stage is expected to take place in the first and second week. The second stage is about creating the test data and executing our first experiments with this data so we can have a look at some preliminary results. We will run statistics like the average and variance on these first results so we can come up with a first answer to our research question. These two stages together form the foundation of our research. After these stages we will review what we have found so far and how this influences the rest of our research. In the third stage we will implement ideas created in the first and second stage. During this stage we will do more thorough experiments to get to our final results and write our paper about the research.

In our experiments we will make every data structure execute the actions on the same test sets. These test sets contain a set of actions that can be performed. The first test set contains 10.000 elements, the second contains 100.000 elements and the third 1.000.000 elements. The actions will be tested with the data specified in chapter 5, Test data. Every test set will be conducted 30 times. Tests which exceed the time threshold of 10 minutes will be seen as too slow and will not be tested any further. We will look at the standard deviation and average of the performed actions on special cases.

For our research we have the following hypotheses:

- The build action will be executed the fastest by the hash table and slowest by the Min-Max tree.
- The search action will be executed the fastest by the hash table and the AVL tree and the slowest by the list and the Min-Max tree.
- The insert action will be executed the fastest by the hash table, the AVL tree and the Min-Max tree and slowest by the list.
- The delete action will be executed the fastest by the hash table and the AVL tree and the slowest by the Min-Max tree and the list.
- The getMin action will be executed the fastest by the list and the Min-Max tree and the slowest by the hash table.
- The extractMin action will be executed the fastest by the AVL tree and the Min-Max tree and the slowest by the hash table.
- The getMax action will be executed the fastest by the list and the Min-Max tree and the slowest by hash table.
- The extractMax action will be executed the fastest by the list and the Min-Max tree and the slowest by the hash table.