HP35 Calculator Report.

Adrian Jonsson Sjödin

Fall 2022

Introduction

The goal of this assignment is to evaluate how the time complexity changes when we apply different searching algorithms, as opposed to simply searching through an array from start to end.

Task 1

Set up a benchmark for the given method that searches through an unsorted array, and determine the relationship of how the time increases for a growing number of elements in the array.

Method

The method to benchmark was already given and required no further modification. What was left to implement for this task was solely the benchmarking of the given method. The implementation of this can be seen in the code bellow.

```java
public static double benchmarkUnsortedSearch(int maxArraySize) {
  Random rnd = new Random();
  int[] arrayToSearch = new int[maxArraySize];
  double sum = 0;
  for (int i = 0; i < 1_000_000; i++) {
    int key = rnd.nextInt(maxArraySize);
    long timeStart = System.nanoTime();
    searchUnsorted(arrayToSearch, key);
    sum += (double) (System.nanoTime() - timeStart) ;
  }
  return sum / 1000000;
}
```

Stack	**Number of calculations**	**Time in ns**	**Time increase**
Static	1 000	2439	-
Dynamic	1 000	10010	4.1
Static	10 000	2251	-
Dynamic	10 000	6410	2.8
Static	1 000 000	1702	-
Dynamic	1 000 000	5033	2.9

Table 1: Stack benchmarks

Result

To benchmark the different stacks we measured the best time out of n runs that it took for the calculator to calculate an expression with 500 values followed by 499 add operations.

Discussion

Task 2

Method

Discussion