UFOAssignment3

Mads Andersen
Benjamin Højgaard

April 2021

1 Task 1

Find a point in your program that can be optimized (for speed), for example by using a profiler

- We used the letterfrequencies project. We investigated whether try/-catch blocks were as efficient/fast as if/else statements for control flow. Turns out, it's only if you catch an exception 10% of the time, it's slower. By using our Stopwatch class, we figured out that it was reading the file that took the longest, so we investigated how to make the FileReader faster and the solution was a BufferedReader.

Make a measurement of the point to optimize, for example by running a number of times, and calculating the mean and standard deviation (see the paper from Sestoft).

- We chose to optimize the tallyChars method. We ran it 10 times and got the following results:
  - Run 1: 89,31 milliseconds
  - Run 2: 90,62 milliseconds
  - Run 3: 88,10 milliseconds
  - Run 4: 86,86 milliseconds
  - Run 5: 91,64 milliseconds
  - Run 6: 85,62 milliseconds
  - Run 7: 86,77 milliseconds
  - Run 8: 88,35 milliseconds
  - Run 9: 89,33 milliseconds
  - Run 10: 89,15 milliseconds

  We used [1] to calculate the mean and standard deviation:

  Standard deviation: 8.37

  Mean deviation: 86.42

2 Hand-in

Short introduction of the program and the part to be optimized

- We used the letterfrequencies project. The project takes a .txt file and count how many times each letter occurs. The part to be optimized is the method that takes a reader and adds the letters to a hashmap (tallyChars).

Documentation of the current performance

- See point 2 under Task 1.

Explanation of bottleneck(s)

- A bottleneck is when a part of software is slower than the rest so the other parts have to wait.

A hypothesis of what causes the problem

- The Reader that reads the .txt file is slow.

A changed program with better performance

- Our new method takes a BufferedReader instead of a FileReader.

```
private static void tallyChars2(BufferedReader reader,
Map<Integer, Long> freq) throws IOException {
    int b;
    while ((b = reader.read()) != -1) {
        try {
            freq.put(b, freq.get(b) + 1);
        } catch (NullPointerException np) {
            freq.put(b, 1L);
        };
    }
}
```

Documentation of the new performance

```
TIME FOR FileReader: 86.430201
TIME FOR BufferedReader: 47.174602
```

Figure 1: Run times

References

[1] Standard Deviation Calculator, "https://www.calculator.net/standard-deviation-calculator.html"