Assignment 2

Daniel Lindholm, Nikolaj Nielsen, Michael Lundsgaard,
Adam Saidane, Sebastian Lundsgaard-Larsen, Emil Hermansen,
Jacob Borg

November 17, 2020

1 Performance

Method	Min	Max	Median	Mean	Std.dev.
Original program: tallyChars	0.0316749	0.1283925	0.0366601	0.04345826	0.0143686
Improved: tallyChars2	0.0175185	0.0825488	0.0191022	0.02384565	0.00937236
Improved v2: tallyChars3	0.0179792	0.0674372	0.0181885	0.0225562	0.00705107
Improved v3: tallyChars4	0.0047943	0.0336532	0.00624575	0.0072654	0.00292975

2 Bottlenecks in the Program

There are several bottlenecks in the program. The first problem is that we are reading directly from a reader class, instead of using a buffered reader. When filereader calls .read() it has to make new call to read from the file system. If we instead use a BufferedReader and call .read(), we read directly from the buffer, preventing us from having to open up the file again. Changing this almost doubles the speed of the program.

3 Cause of problem

We can then furthermore increase the speed of the program by over double our new the already faster code, by not using a hashmap. Since we are only reading characters as there numeric value, such as 34, 58, 90 etc. We can then instead create an integer array of size 255. where the index represent a character, and

the value represent how many times we have seen that character. We then for each value we get from .read() increment the value on that index.

The reason a haspmap is so slow, is that whenever we have to increment how many times we have seen a specific character, we have to first use the .get() method on that hashpmap which can take a long time, it is O(n) time but it is an amortized time, and a lot slower than looking, on a specific index of an array. And because we don't know if we have seen the that index in the haspmap before, we first have to check if it has been seen before. so we end up having to call the .get() twice for every character, making it even slower.

To stay true to the structure of the original program. We convert the produced array to a hashmap, and even though we do this additional action it is still faster than the original, because we only add to the hashmap 255 times insead, of the total length of characters in the file.