P4 Part 1 Analysis

Henry Barklam (hgb13)

Marcus Deans (md374)

CS201: Data Structures and Algorithms

5 March 2020

We have adhered to the Duke Community Standard in completing this assignment.

dna length = 636

cutting at enzyme gaattc

-----

Class splicee recomb time appends

-----

StringStrand: 256 886 0.000 2

StringStrand: 512 1,142 0.000 2

StringStrand: 1,024 1,654 0.000 2

StringStrand: 2,048 2,678 0.000 2

StringStrand: 4,096 4,726 0.000 2

StringStrand: 8,192 8,822 0.000 2

StringStrand: 16,384 17,014 0.000 2

StringStrand: 32,768 33,398 0.000 2

StringStrand: 65,536 66,166 0.000 2

StringStrand: 131,072 131,702 0.000 2

StringStrand: 262,144 262,774 0.000 2

StringStrand: 524,288 524,918 0.001 2

StringStrand: 1,048,576 1,049,206 0.001 2

StringStrand: 2,097,152 2,097,782 0.002 2

StringStrand: 4,194,304 4,194,934 0.006 2

StringStrand: 8,388,608 8,389,238 0.011 2

StringStrand: 16,777,216 16,777,846 0.019 2

StringStrand: 33,554,432 33,555,062 0.047 2

StringStrand: 67,108,864 67,109,494 0.070 2

StringStrand: 134,217,728 134,218,358 0.156 2

The following result was not obtained using StringBuilderStrand so was excluded for better comparison between the two.

StringStrand: 268,435,456 268,436,086 2.218 2

Using mySource = mySource + mySource + mySource;

dna length = 1,920

cutting at enzyme gaattc

-----

Class splicee recomb time appends

-----

StringStrand: 256 2,670 0.000 6

StringStrand: 512 3,438 0.000 6

StringStrand: 1,024 4,974 0.000 6

StringStrand: 2,048 8,046 0.000 6

StringStrand: 4,096 14,190 0.000 6

StringStrand: 8,192 26,478 0.000 6

StringStrand: 16,384 51,054 0.000 6

StringStrand: 32,768 100,206 0.000 6

StringStrand: 65,536 198,510 0.001 6

StringStrand: 131,072 395,118 0.002 6

StringStrand: 262,144 788,334 0.003 6

StringStrand: 524,288 1,574,766 0.007 6

StringStrand: 1,048,576 3,147,630 0.020 6

StringStrand: 2,097,152 6,293,358 0.019 6

StringStrand: 4,194,304 12,584,814 0.043 6

StringStrand: 8,388,608 25,167,726 0.053 6

StringStrand: 16,777,216 50,333,550 0.159 6

StringStrand: 33,554,432 100,665,198 0.553 6

StringStrand: 67,108,864 201,328,494 0.387 6

StringStrand: 134,217,728 402,655,086 0.940 6

dna length = 636

cutting at enzyme gaa-----

Class splicee recomb time appends

StringBuilderStrand: 256 886 0.000 2

StringBuilderStrand: 512 1,142 0.000 2

StringBuilderStrand: 1,024 1,654 0.000 2

StringBuilderStrand: 2,048 2,678 0.000 2

StringBuilderStrand: 4,096 4,726 0.000 2

StringBuilderStrand: 8,192 8,822 0.000 2

StringBuilderStrand: 16,384 17,014 0.000 2

StringBuilderStrand: 32,768 33,398 0.000 2

StringBuilderStrand: 65,536 66,166 0.000 2

StringBuilderStrand: 131,072 131,702 0.000 2

StringBuilderStrand: 262,144 262,774 0.000 2

StringBuilderStrand: 524,288 524,918 0.001 2

StringBuilderStrand: 1,048,576 1,049,206 0.002 2

StringBuilderStrand: 2,097,152 2,097,782 0.008 2

StringBuilderStrand: 4,194,304 4,194,934 0.010 2

StringBuilderStrand: 8,388,608 8,389,238 0.015 2

StringBuilderStrand: 16,777,216 16,777,846 0.030 2

StringBuilderStrand: 33,554,432 33,555,062 0.042 2

StringBuilderStrand: 67,108,864 67,109,494 0.092 2

StringBuilderStrand: 134,217,728 134,218,358 0.197 2

ttc

Using mySource = mySource + mySource + mySource;

dna length = 1,920

cutting at enzyme gaattc

-----

Class splicee recomb time appends

-----

StringBuilderStrand: 256 2,670 0.000 6

StringBuilderStrand: 512 3,438 0.000 6

StringBuilderStrand: 1,024 4,974 0.000 6

StringBuilderStrand: 2,048 8,046 0.000 6

StringBuilderStrand: 4,096 14,190 0.000 6

StringBuilderStrand: 8,192 26,478 0.000 6

StringBuilderStrand: 16,384 51,054 0.001 6

StringBuilderStrand: 32,768 100,206 0.001 6

StringBuilderStrand: 65,536 198,510 0.001 6

StringBuilderStrand: 131,072 395,118 0.002 6

StringBuilderStrand: 262,144 788,334 0.005 6

StringBuilderStrand: 524,288 1,574,766 0.012 6

StringBuilderStrand: 1,048,576 3,147,630 0.033 6

StringBuilderStrand: 2,097,152 6,293,358 0.054 6

StringBuilderStrand: 4,194,304 12,584,814 0.071 6

StringBuilderStrand: 8,388,608 25,167,726 0.134 6

StringBuilderStrand: 16,777,216 50,333,550 0.348 6

StringBuilderStrand: 33,554,432 100,665,198 1.331 6

StringBuilderStrand: 67,108,864 201,328,494 2.475 6

StringBuilderStrand: 134,217,728 402,655,086 1.378 6

**Question 1**: ***are the benchmark timings for StringStrand consistent*** with the explanation below that the time to execute cutAndSplice is **O(b2S)**?

Note that the value of **b** is half the number of calls to append since each cut (except the first) is modeled by two calls of append in the method **cutAndSplice** -- see the code. This means that **b2** will be constant in the benchmark, but S will vary.

***What patterns, if any, do you find in the runtimes for StringStrand?*** In answering this question *include some data from the first few runs* of the Benchmark program where you use

**mySource = mySource + mySource + mySource;**

After the call to **mySource = dnaFromScanner(...**

This will triple the number of breaks which you'll see in the number of calls to append printed by the benchmark program.

**Question 1:**

The benchmark data that we collected using cutAndSplice with StringStrand is consistent with the explanation that the time to execute is O(b^2 s). B represents half the number of calls to append, and as the data size increases for the program, there are more calls to end. Therefore, b will increase and thus b^2 will increase exponentially. This is observed in the data and demonstrates why the benchmark timings are consistent with runtime O(n^2).

In the runtimes using mySource = mySource + mySource + mySource, it was observed that the differences in runtime are initially smaller with the lower values. As the size increased, it was seen that the runtime increased rapidly at an increasing rate. However, there were certain cases such as when the runtime was 0.043 and the subsequent runtime was 0.053, which was a minimal difference, whereas the next runtime was 0.159 which was a significantly increased, with the runtime almost tripling. This increase seemed somewhat irregular and was surprising. Finally, it was observed that the runtime decreased between the third last and second last values even as the size of the data increased, from 0.553 to 0.387, which was similarly surprising. More examination is needed to elucidate the cause of these peculiarities in the runtimes.

**Question 2:**

The benchmark data that we collected using cutAndSplice with StringStrand is consistent with the explanation that the time to execute is O(bS) in big-O notation. ‘b’ increases linearly because b is half the number of calls to append, therefore the data size increases with more calls to append. This additionally clarifies why the benchmark timings make sense regarding assigning it a runtime O(n).

P4 Part 2 Analysis

Henry Barklam (hgb13)

Marcus Deans (md374)

CS201: Data Structures and Algorithms

23 March 2020

We have adhered to the Duke Community Standard in completing this assignment.

dna length = 1,920

cutting at enzyme gaattc

-----

Class splicee recomb time appends

-----

LinkStrand: 256 2,670 0.000 6

LinkStrand: 512 3,438 0.000 6

LinkStrand: 1,024 4,974 0.000 6

LinkStrand: 2,048 8,046 0.000 6

LinkStrand: 4,096 14,190 0.000 6

LinkStrand: 8,192 26,478 0.000 6

LinkStrand: 16,384 51,054 0.000 6

LinkStrand: 32,768 100,206 0.000 6

LinkStrand: 65,536 198,510 0.000 6

LinkStrand: 131,072 395,118 0.000 6

LinkStrand: 262,144 788,334 0.000 6

LinkStrand: 524,288 1,574,766 0.000 6

LinkStrand: 1,048,576 3,147,630 0.000 6

LinkStrand: 2,097,152 6,293,358 0.000 6

LinkStrand: 4,194,304 12,584,814 0.000 6

LinkStrand: 8,388,608 25,167,726 0.000 6

LinkStrand: 16,777,216 50,333,550 0.000 6

LinkStrand: 33,554,432 100,665,198 0.000 6

LinkStrand: 67,108,864 201,328,494 0.000 6

LinkStrand: 134,217,728 402,655,086 0.000 6

LinkStrand: 268,435,456 805,308,270 0.000 6

LinkStrand: 536,870,912 1,610,614,638 0.000 6

**Question 3:** Explain why the time for LinkStrand does not change much at all over all the runs in the benchmark program. Explain why you think memory is exhausted as the specific strand size you see in your timings -- as compared to exhaustion for StringBuilder.

With LinkStrand, the Benchmark program had sufficient memory to test a splice four times the size of the greatest possible with StringBuilderStrand, and twice that of StringStrand. This is markedly more efficient and the reason is the same as why the time for LinkStrand is the same over the entire program. LinkStrand is considerably more efficient due to the use of linkedlists to store the information effectively. Specifically, the longer runtime of StringBuilderStrand and StringStrand is due to the method of concatenation; in these two slower programs, the characters are appended bit by bit which for large splices leads to high memory usage and subsequent failure of the benchmark program, as well as a high runtime. As noted above, StringBuilderStrand is O(bS) runtime for creating the new strand and StringStrand is O(b^2S); demonstrating why there is failure when the number of splices is large. By contrast, LinkStrand can use pointers so that the splice that is being used in the run is pointed to by multiple nodes instead of having to be re-referenced every single time. Combined with the usage of nodes for the remaining data for the rest of the DNA strand and it is apparent that LinkStrand is significantly more efficient that the prior versions.