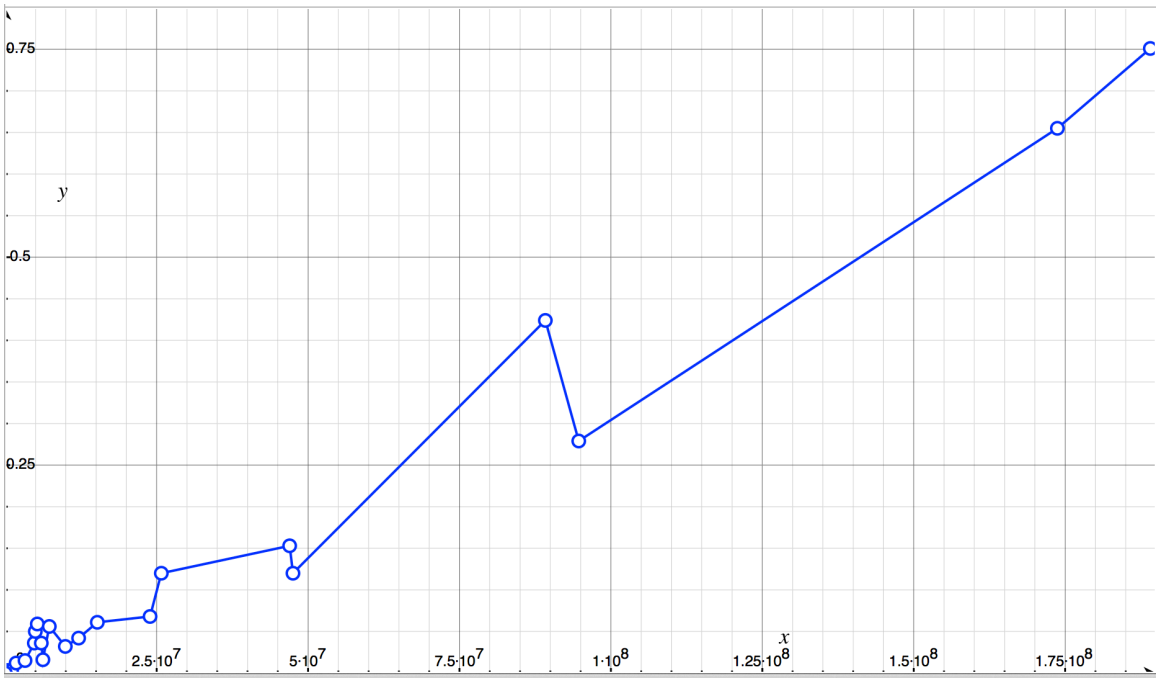
Analysis-Dna Assignment
Katrina Zhu

Benchmark Part 1

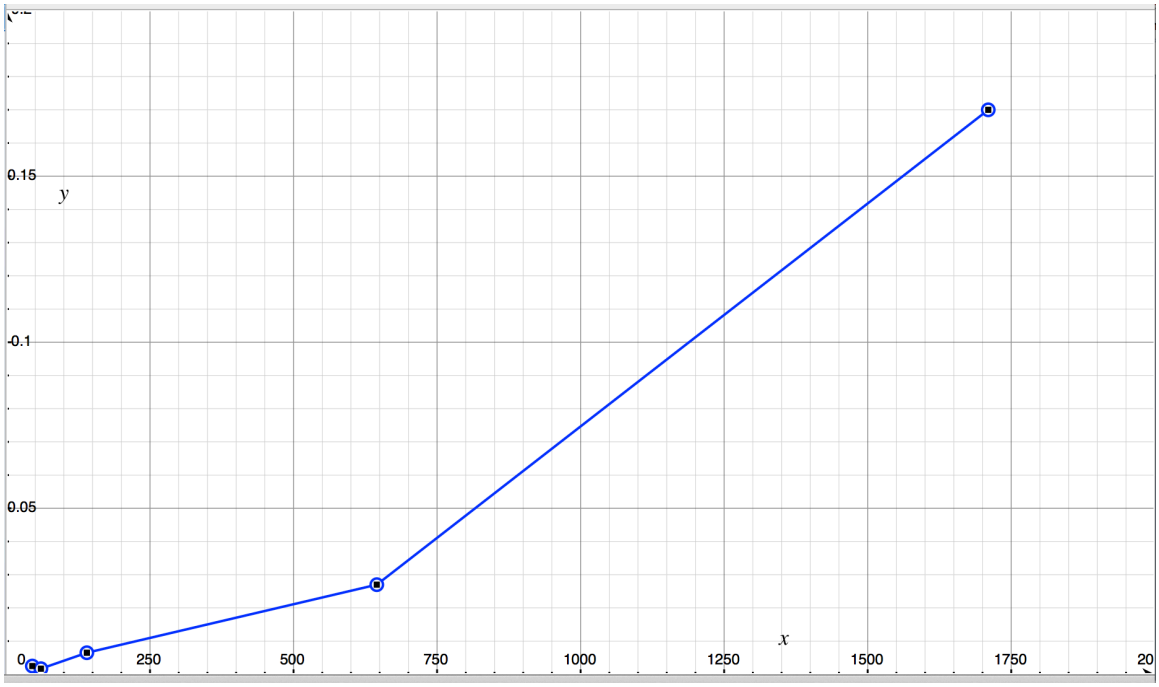


This graph combines data of running SimpleStrand on both ecoli.txt and ecoli_small.txt. The x-axis is the length of the recombined strand and the y-axis is the time that cutAndSplice takes. We can see that there is a fairly linear relationship between the two, meaning that the runtime is O(N), where N is the length of the recombined strand.

- This is the case because cutAndSplice uses the append method to add Strings, and the append method contains the method StringBuilder.append(String s), which has a runtime of O(N). Thus, SimpleStrand.append has a runtime of O(N); thus, cutAndSplice has a runtime of O(N).

Benchmark Part 2
When run on ecoli.txt with the following updated MB:
2048 MB→262144 splicee
4096 MB→524288 splicee
8129 MB→1048576 splicee
16384 MB→1048576 splicee
32768 MB→1048576 splicee
524288 MB→1048576 splicee
The largest splicee my machine can manage is 1,048,576 characters long.

Benchmark Part 3



The above graph models the relation between the number of breaks, on the x axis, and the time that LinkStrand.cutAndSplice takes, on the y axis. The number of breaks, or B, was calculated by dividing the number of appends by 2. I got various different data points by using ecoli, ecoli_small, and combinations of the two. This graph shows an approximate linear relation between the number of breaks and time; thus, the runtime of cutAndSplice in LinkStrand is O(B).

- This is the case because the Append method's runtime is O(1), and the number of times append is called is directly proportional to the number of breaks. Append's runtime is O(1) because linked lists only deal with whole nodes; thus, they do not need to worry about the length of the strings within these nodes. Therefore, unlike SimpleStrand's Append method, LinkStrand's Append method is independent of length; the only factor affecting time is how many times it is called.