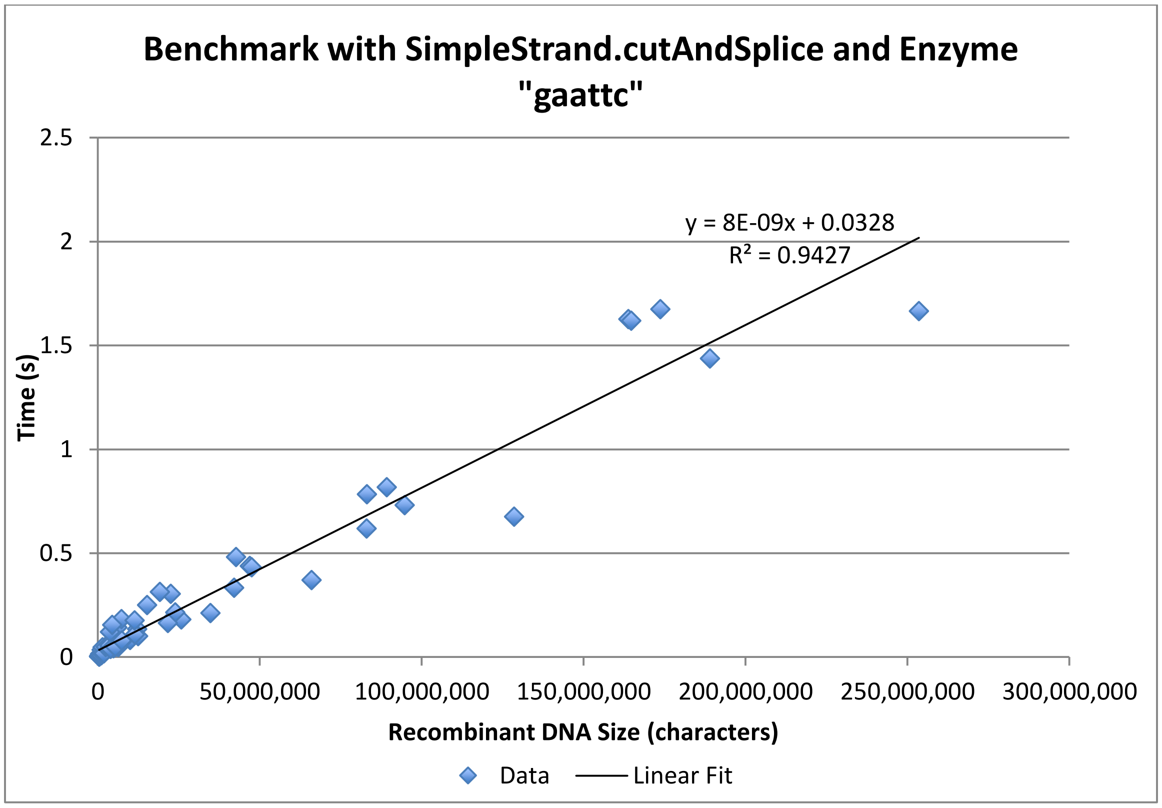
Xing Su
CompSci 201
3/5/13

# DNA Analysis

## Part A

**Benchmark the code you're given in `SimpleStrand.cutAndSplice` that correctly finds all occurrences of an enzyme and replaces the enzyme with another strand of DNA. The benchmarking is done by the class `DNABenchMark`. Your benchmarking report must show that the algorithm/code in `SimpleStrand.cutAndSplice` is O(N) where N is the size of the recombined strand returned.**

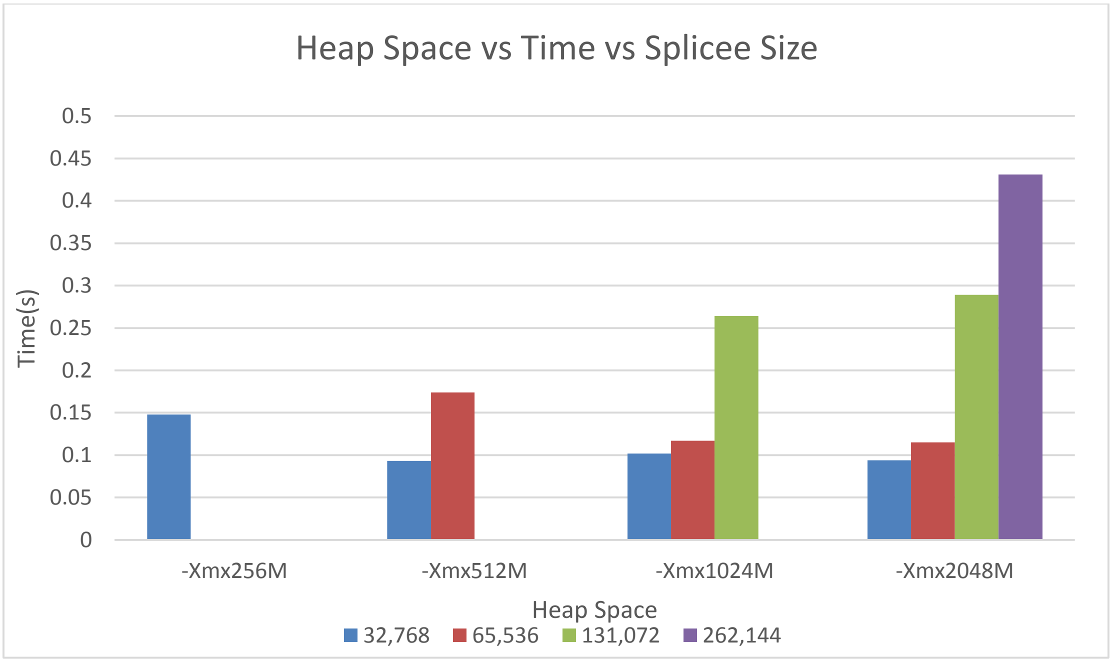


As we can see from the above, as the size of recombinant DNA increase, the time needed to execute increases linearly (linear fit has an $R^2$ value of .9427), thus indicating O(N) runtime.

This is because each time a string (DNA strand) is appended, a string is added to it. Because Strings are immutable and each time a string is appended, a new string is created and the original content is copied over, which means that the longer the string appended is, the longer the running time will be.

## Part B

**Test your benchmarking and O(N) simulation by running out of memory and then re-running the simulation with more memory.**



As we can see above, running the DNAbenchmark with more heap space each time will allow the next order-2 splice length to execute before running out of memory. This makes sense as the more memory the program is allocated, the more memory it has to store and thus successfully execute the splice operations. With each doubling of heap space, the running time of the previous max strand stays fairly consistent.

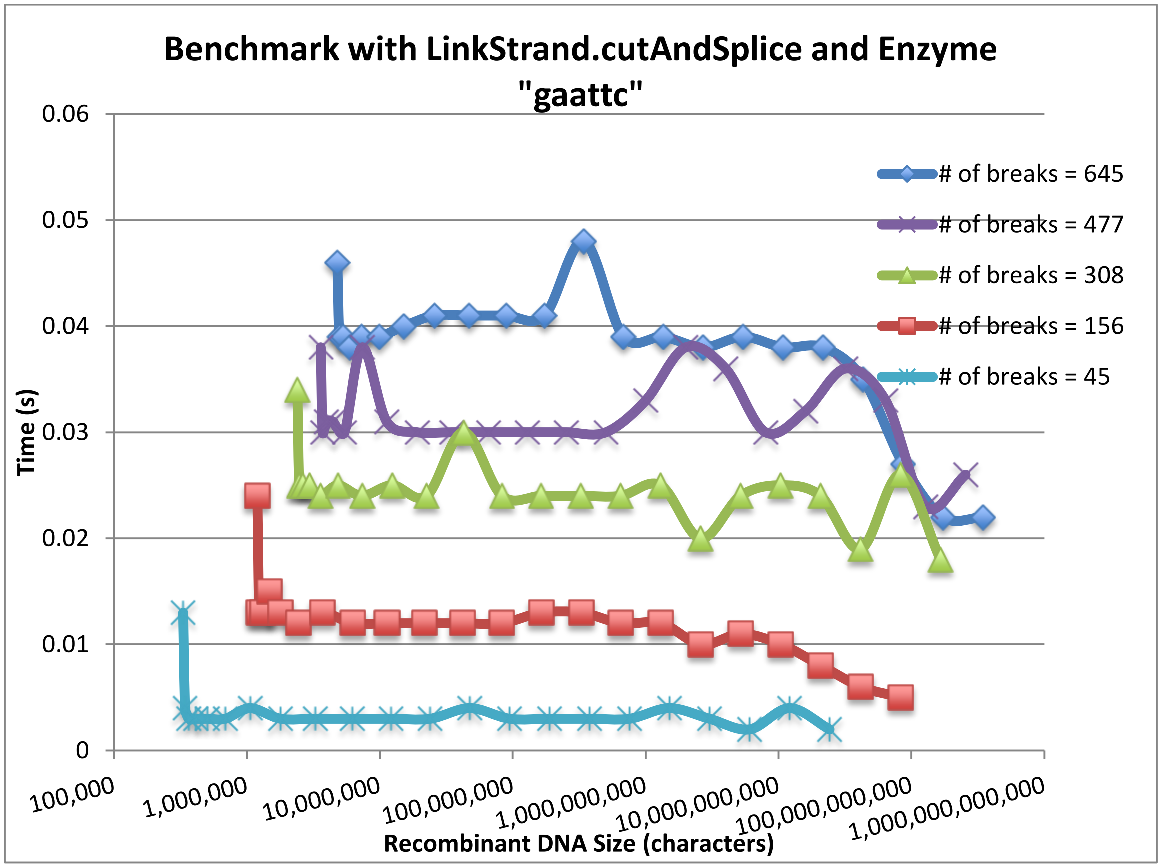
## Part C

**Design, code, and test `LinkStrand` that implements the `IDnaStrand` interface so that your implementation passes the JUnit tests in `TestStrand`.**

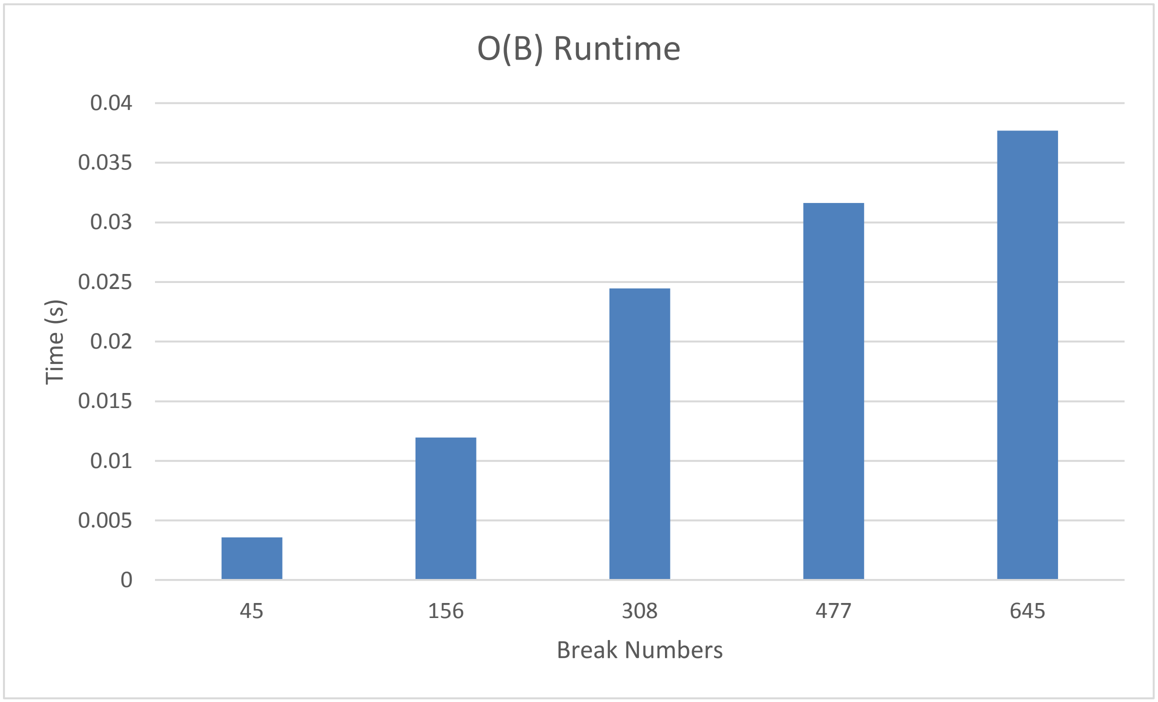
See code.

## Part D

**You must run *virtual experiments* to show that your `LinkStrand` linked-list implementation isO(B) for a strand with B breaks as described below.**



As we can see from above, the running time no longer depend on the recombinant DNA sizes. The variation in the running times as the size of recombinant DNA increase are purely due to the fluctuations of the system clock and other processes.



Here we can see a clear linearly increasing trend: as the number of breaks increase, the average time needed to execute the splice operation linearly increase, thus the O(B) runtime.