Joel Mire

DNA Analysis

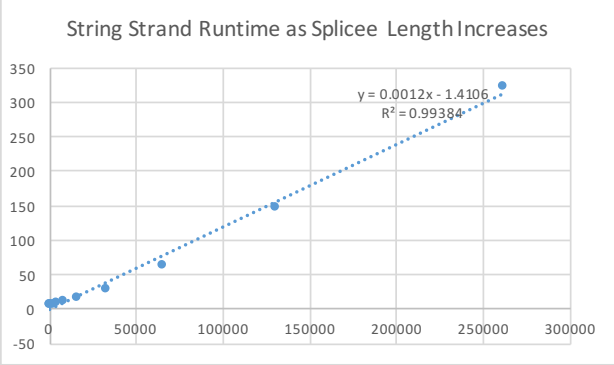
*see code at the bottom of this document for explanation of code used for tests. b refers to number of times an enzyme is cut and replaced by a splicee. s refers to the length of the splicee.

Each of the three hypotheses is not interested in the O(n) iteration through the list, but rather, b and s. Thus each hypothesis will combine the big Oh runtimes of the b and s components relevant to the given strand type.

Non Linked Lists Hypothesis

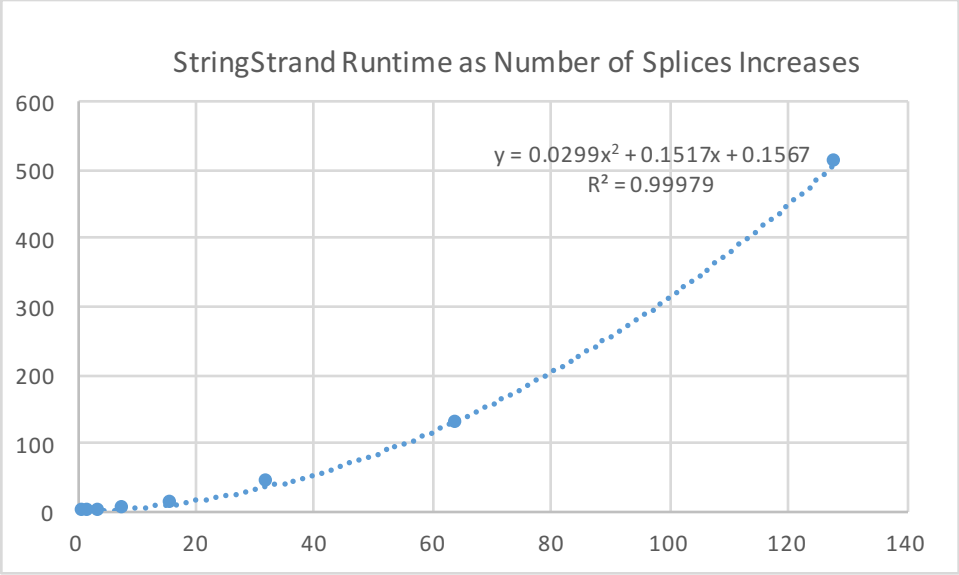
StringStrand

```
dna length = 4,639,221
cutting at enzyme gaattc
-----
Class                   splicee             recomb    time    appends
-----
StringStrand:              256              4,800,471       4.465   1290
StringStrand:              512              4,965,591       4.641   1290
StringStrand:            1,024              5,295,831       4.957   1290
StringStrand:            2,048              5,956,311       6.429   1290
StringStrand:            4,096              7,277,271       7.719   1290
StringStrand:            8,192              9,919,191       9.725   1290
StringStrand:           16,384             15,203,031       14.690  1290
StringStrand:           32,768             25,770,711       27.208  1290
StringStrand:           65,536             46,906,071       63.388  1290
StringStrand:          131,072             89,176,791       147.382 1290
StringStrand:          262,144            173,718,231       321.301 1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.util.Arrays.copyOf(Arrays.java:3332)
        at java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:124)
        at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:448)
        at java.lang.StringBuilder.append(StringBuilder.java:136)
        at StringStrand.append(StringStrand.java:70)
        at IDnaStrand.cutAndSplice(IDnaStrand.java:40)
        at DNABenchmark.strandSpliceBenchmark(DNABenchmark.java:67)
        at DNABenchmark.main(DNABenchmark.java:110)
```



S follows a linear relationship, 0(s)

b	runtime
1024	0.1
2048	0.2397
4096	0.6481
8192	2.4161
16384	9.8361
32768	41.1525
65536	128.6742
131072	510.048



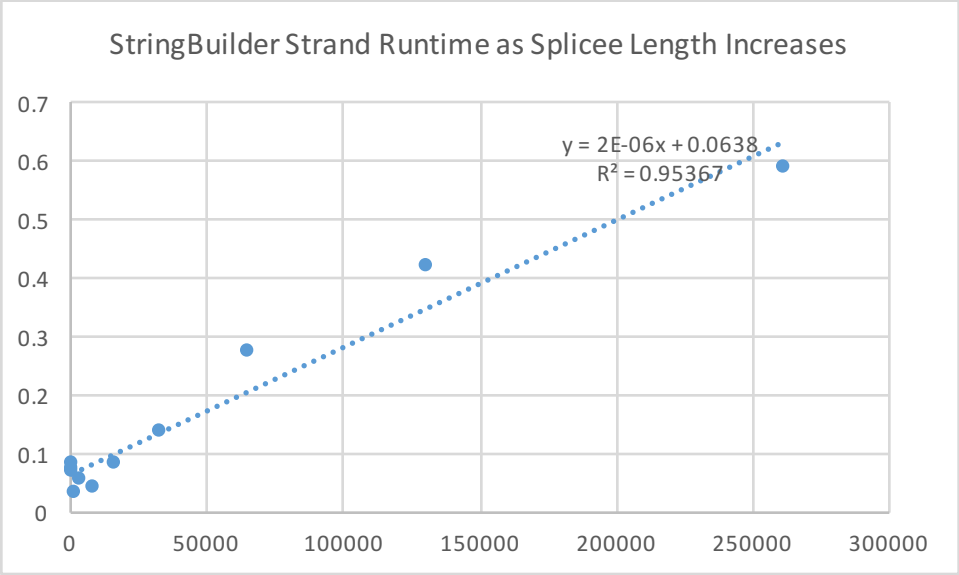
b's leading exponent is x^2, so O(b^2)

Thus O(s) and O(b^2) show that the big Oh for StringStrand is O(b^2s).

The process of appending two strings is quite inefficient in this implementation.

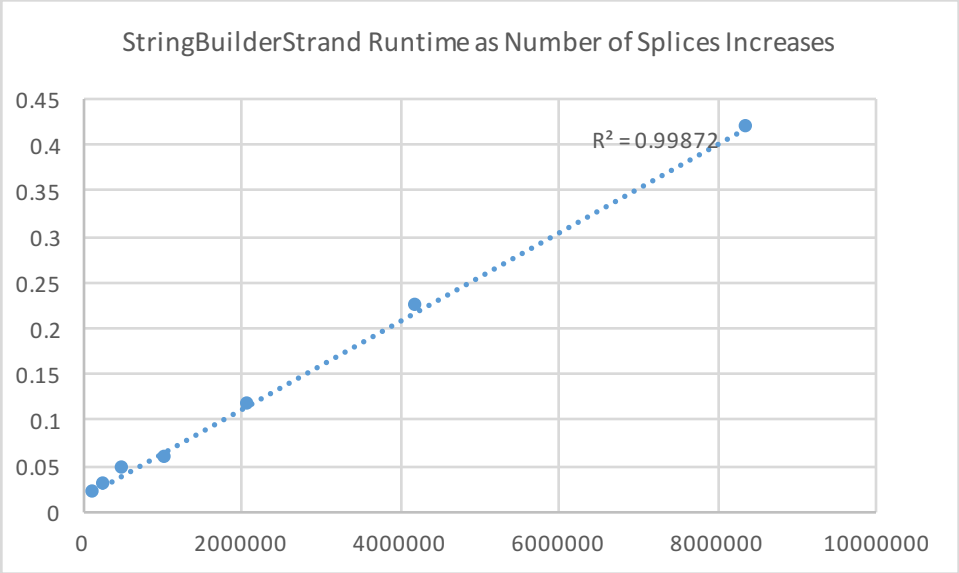
StringBuilderStrand

```
dna length = 4,639,221
cutting at enzyme gaattc
-----
Class                   splicee          recomb  time   appends
-----
StringBuilderStrand:                       256          4,800,471     0.081  1290
StringBuilderStrand:                       512          4,965,591     0.069  1290
StringBuilderStrand:                     1,024          5,295,831     0.071  1290
StringBuilderStrand:                     2,048          5,956,311     0.030  1290
StringBuilderStrand:                     4,096          7,277,271     0.053  1290
StringBuilderStrand:                     8,192          9,919,191     0.041  1290
StringBuilderStrand:                    16,384         15,203,031     0.083  1290
StringBuilderStrand:                    32,768         25,770,711     0.138  1290
StringBuilderStrand:                    65,536         46,906,071     0.274  1290
StringBuilderStrand:                   131,072         89,176,791     0.417  1290
StringBuilderStrand:                   262,144        173,718,231     0.588  1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
       at java.util.Arrays.copyOf(Arrays.java:3332)
       at
java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:124)
       at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:448)
       at java.lang.StringBuilder.append(StringBuilder.java:136)
       at StringBuilderStrand.append(StringBuilderStrand.java:70)
       at IDnaStrand.cutAndSplice(IDnaStrand.java:40)
       at DNABenchmark.strandSpliceBenchmark(DNABenchmark.java:67)
       at DNABenchmark.main(DNABenchmark.java:110)
```



S follows a linear relationship, 0(s)

b	runtime
131072	0.0204
262144	0.0282
524288	0.0471
1048576	0.0568
2097152	0.1161
4194304	0.2228
8388608	0.4169



b follows a linear relationship, 0(b)

Thus O(s) and O(b) show that the big Oh for StringBuilderStrand is O(bs).

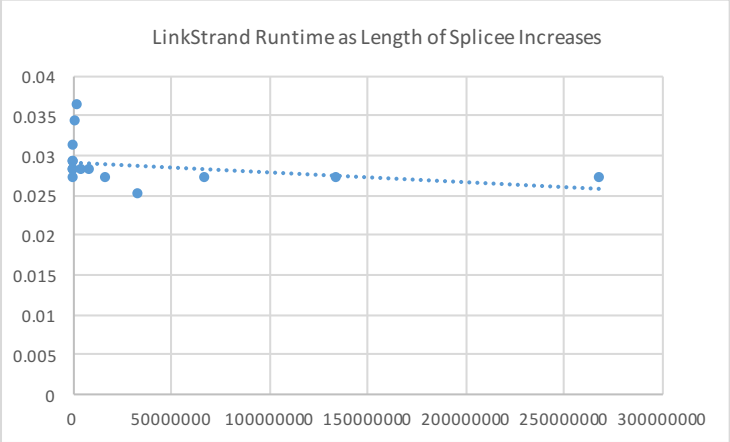
String builder does not have to iterate through both strings in order to join the two, and thus is faster by an order of b.

Linked Lists Hypothesis

LinkStrand

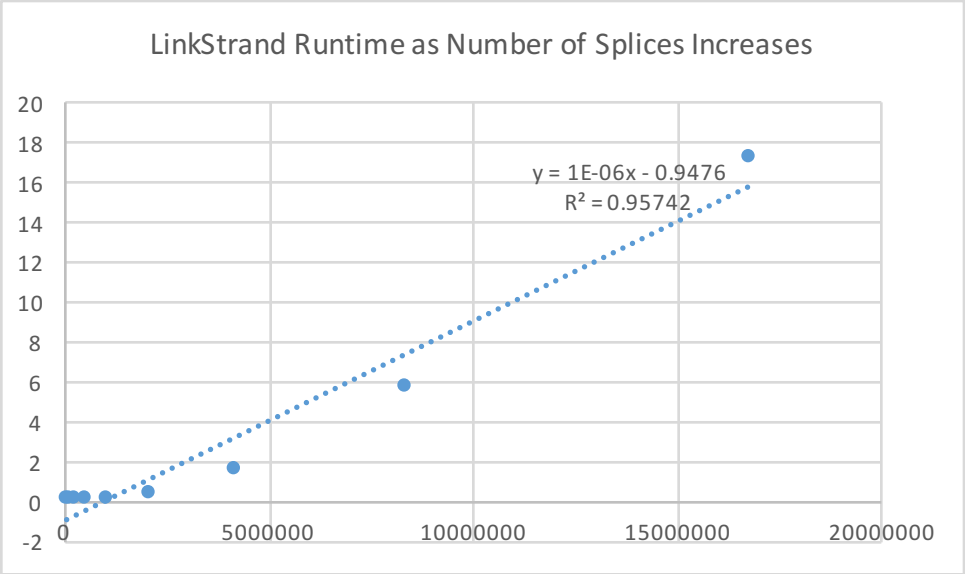
dna length = 4,639,221
cutting at enzyme gaattc

```
-----
Class                   splicee             recomb    time    appends
-----
LinkStrand:                 256             4,800,471       0.039   1290
LinkStrand:                 512             4,965,591       0.036   1290
LinkStrand:               1,024             5,295,831       0.036   1290
LinkStrand:               2,048             5,956,311       0.027   1290
LinkStrand:               4,096             7,277,271       0.033   1290
LinkStrand:               8,192             9,919,191       0.037   1290
LinkStrand:              16,384            15,203,031       0.029   1290
LinkStrand:              32,768            25,770,711       0.027   1290
LinkStrand:              65,536            46,906,071       0.029   1290
LinkStrand:             131,072            89,176,791       0.031   1290
LinkStrand:             262,144           173,718,231       0.028   1290
LinkStrand:             524,288           342,801,111       0.028   1290
LinkStrand:           1,048,576           680,966,871       0.034   1290
LinkStrand:           2,097,152         1,357,298,391       0.036   1290
LinkStrand:           4,194,304         2,709,961,431       0.028   1290
LinkStrand:           8,388,608         5,415,287,511       0.028   1290
LinkStrand:          16,777,216        10,825,939,671       0.027   1290
LinkStrand:          33,554,432        21,647,243,991       0.025   1290
LinkStrand:          67,108,864        43,289,852,631       0.027   1290
LinkStrand:         134,217,728        86,575,069,911       0.027   1290
LinkStrand:         268,435,456       173,145,504,471       0.027   1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.util.Arrays.copyOf(Arrays.java:3332)
        at java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:124)
        at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:448)
        at java.lang.StringBuilder.append(StringBuilder.java:136)
        at DNABenchmark.main(DNABenchmark.java:107)
```



The near constant time results suggests that s follows a constant relationship, so O(1) at some C constant

b	runtime
65536	0.0205
131072	0.0251
262144	0.0299
524288	0.062
1048576	0.0875
2097152	0.346
4194304	1.4906
8388608	5.6325
16777216	17.1801



b follows a linear relationship, so O(n).

Thus O(1) and O(b) show that the big Oh for StringBuilderStrand is O(b).

*Note that here, as in all of the hypotheses, the O(N) sequential iteration is ignored for big Oh values…
Because the splicee is a single object with many nodes in the DNA strand ultimately referencing it, the complete runtime would be O(N + b).

Comment on Code for DNA Benchmark:

```
for (int j = 8; j <= 32; j++){
                StringBuilder b = new StringBuilder("");
                int spSize = (int) Math.pow(2, j);
                for (int k = 0; k < spSize; k++) {
                     b.append("c");
                }
                String splicee = b.toString();
                String results = strandSpliceBenchmark(ENZYME,
splicee, strandType);
                System.out.println(results);
           }
```

the outer for loop in the code above controls the size s of a given splicee. This code from DNA benchmark allows us to test runtime with the independent variable being s.

DNA Benchmark, more generally, tests by first scanning in a synthetic DNA strand averaging a number of trials of cut/splice operations.

Comments on Code for DNA Analysis:

```
public class DNAAnalysis {

     private static final String ENZYME = "gaattc";
     private static final String SPLICEE = "ttttttttttttttttttttttt";
     private static final int NUM_TRIALS = 10;

     public static void test() {
          for (int b = 1; b < 80000000; b *= 2){
                StringBuilder dna = new StringBuilder();
                for (int i = 0; i < b; i++)
                    dna.append(ENZYME);
                while (dna.length() < 10000000){
                     dna.append("x");
                }
```

```
//              IDnaStrand strand = new StringStrand(dna.toString());
//              IDnaStrand strand = new
StringBuilderStrand(dna.toString());
                IDnaStrand strand = new LinkStrand(dna.toString());
                double totalTime = 0;
                for (int trial = 0; trial < NUM_TRIALS; trial++) {
                    double start = System.nanoTime();
                    strand.cutAndSplice(ENZYME, SPLICEE);
                    totalTime += (System.nanoTime() - start) / 1e9;
                }
                totalTime /= NUM_TRIALS;
                System.out.printf("%6d\t%6.4f\n", b, totalTime);
          }
     }

     public static void main(String[] args){
          test();
     }

}
```

My created DNA Analysis class tests b, the number of times a cut/splice operation will occur in a given DNA strand of invariant length N. To achieve the same size strand, even as number of b splices adds to the strand, a while loop brings the size of the synthetic DNA up to a constant length with an arbitrary char value unique from the Enzyme that signals a cut/splice operation. It averages using a class variable NUM_TRIALS to stabilize results as it iterates through different values of b to measure runtime.