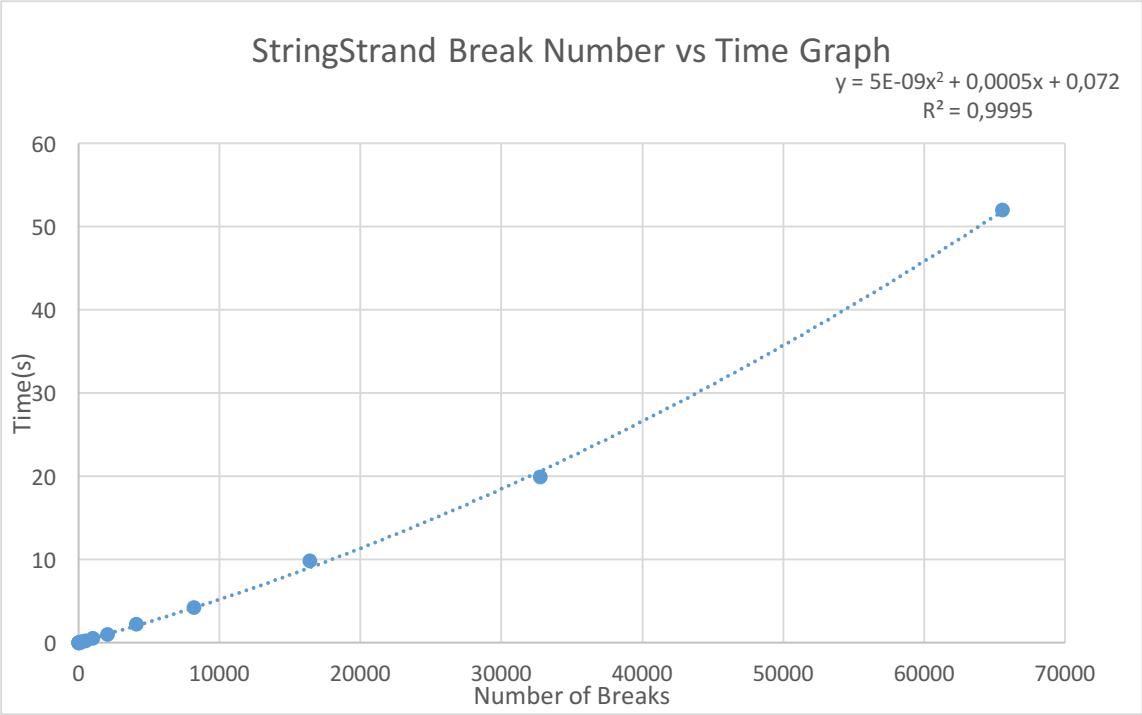
ANALYSIS

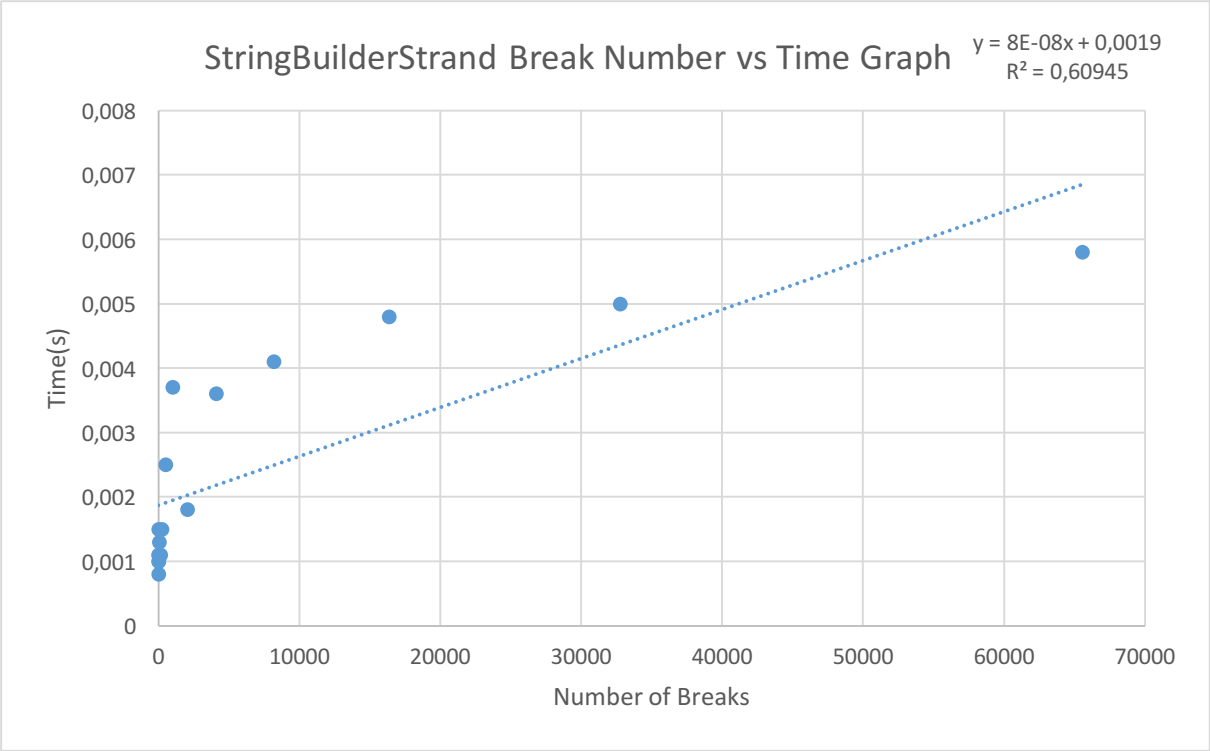


StringStrand

Number of Breaks	Time(s)
1	0.0055
2	0.001
4	0.0035
8	0.0105
16	0.0133
32	0.0354
64	0.0601
128	0.1128
256	0.1529
512	0.2696
1024	0.5455
2048	1.0509
4096	2.2837
8192	4.2481
16384	9.8873
32768	19.917
65536	51.9755

This is the graph and data of StringStrand's timing with varying break numbers(b) with fixed N. As it can be seen from the graph, String Strand follows an x^2 trend with really high R^2 value of 0.9995. Thus cut and splice takes multiple of O(b^2) time for String Strand. As it's known that it varies linearly with the change of splice's length, its big Oh is deduced to be O(b^2). Thus the hypothesis is correct.

ANALYSIS

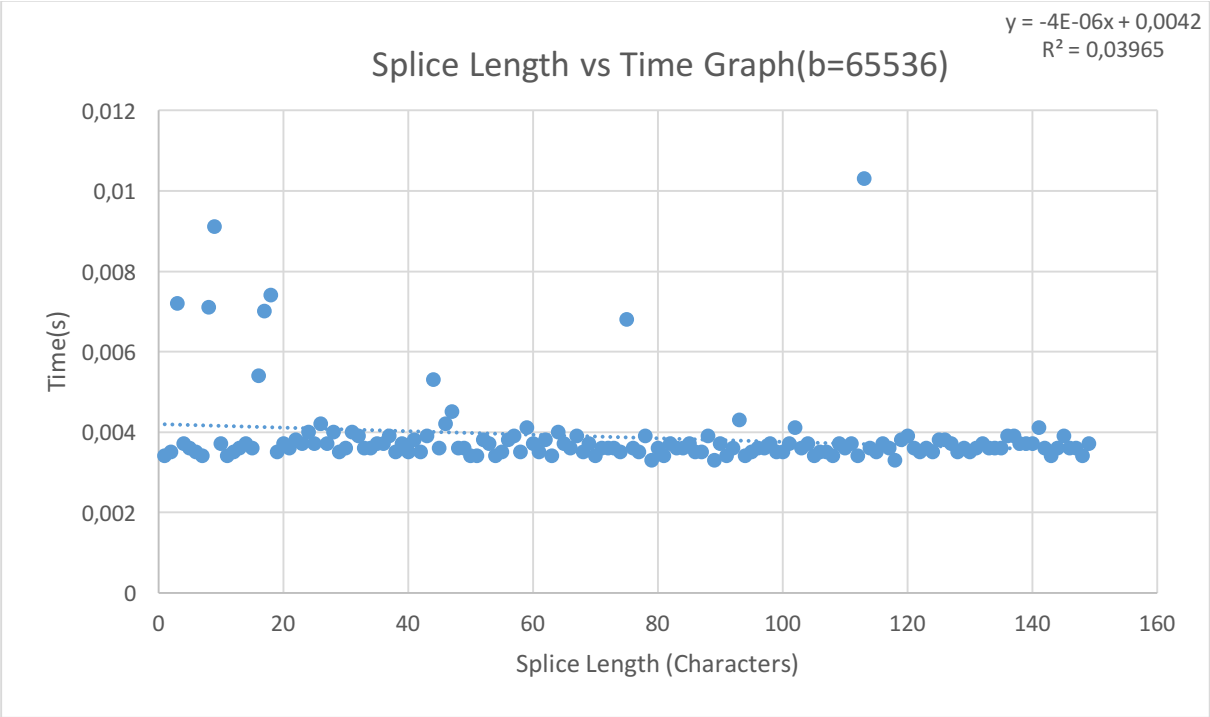


Number of Breaks	Time(s)
1	0.001
2	0.0008
4	0.0015
8	0.0011
16	0.001
32	0.0011
64	0.0013
128	0.0011
256	0.0015
512	0.0025
1,024	0.0037
2,048	0.0018
4,096	0.0036
8,192	0.0041
16,384	0.0048
32768	0.005
65536	0.0058

This is the graph and data of StringBuilderStrand's break number vs time. The figures used for timing are very small especially for small number of breaks. Thus small fluctuations that are not directly related to actual time complexity causes inaccuracy and resulting to the low R^2 value found in the graph. However, if one looks at the data for the highest six number of breaks, it is clear that String Builder Strand follows O(b). As it's known that big-Oh also depends on the length of Splice, the final big-oh is O(bS) and hypothesis is correct. This is much faster than StringStrand.

ANALYSIS

For B = 65536



Length of Splice	Time(s)
1	0.0034
2	0.0035
3	0.0072
4	0.0037
5	0.0036
6	0.0035
7	0.0034
8	0.0071
9	0.0091
10	0.0037
11	0.0034
12	0.0035
13	0.0036
14	0.0037
15	0.0036
16	0.0054
17	0.007
18	0.0074
19	0.0035
20	0.0037
21	0.0036

This is the graph and data for CutandSplice for LinkStrand with varying splice length and constant break number(65536). Other than a few randomly fluctuated data, graph suggests no relation between splice length and time complexity. Thus the part of the hypothesis that suggests LinkStrand's time complexity does not depend on the splice length is true.

22	0.0038
23	0.0037
24	0.004
25	0.0037
26	0.0042
27	0.0037
28	0.004
29	0.0035
30	0.0036
31	0.004
32	0.0039
33	0.0036
34	0.0036
35	0.0037
36	0.0037
37	0.0039
38	0.0035
39	0.0037
40	0.0035
41	0.0038
42	0.0035
43	0.0039
44	0.0053
45	0.0036
46	0.0042
47	0.0045
48	0.0036
49	0.0036
50	0.0034
51	0.0034
52	0.0038
53	0.0037
54	0.0034
55	0.0035
56	0.0038
57	0.0039
58	0.0035
59	0.0041
60	0.0037
61	0.0035
62	0.0038
63	0.0034

ANALYSIS

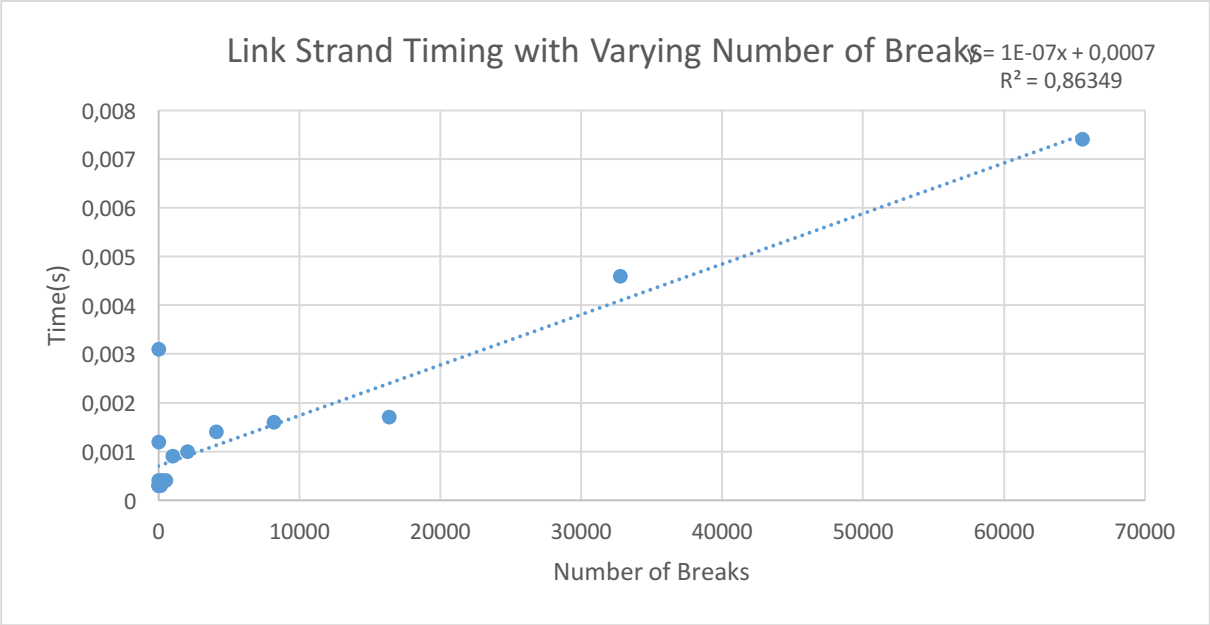
64	0.004
65	0.0037
66	0.0036
67	0.0039
68	0.0035
69	0.0037
70	0.0034
71	0.0036
72	0.0036
73	0.0036
74	0.0035
75	0.0068
76	0.0036
77	0.0035
78	0.0039
79	0.0033
80	0.0036
81	0.0034
82	0.0037
83	0.0036
84	0.0036
85	0.0037
86	0.0035
87	0.0035
88	0.0039
89	0.0033
90	0.0037
91	0.0034
92	0.0036
93	0.0043
94	0.0034
95	0.0035
96	0.0036
97	0.0036
98	0.0037
99	0.0035
100	0.0035
101	0.0037
102	0.0041
103	0.0036
104	0.0037
105	0.0034

ANALYSIS

106	0.0035
107	0.0035
108	0.0034
109	0.0037
110	0.0036
111	0.0037
112	0.0034
113	0.0103
114	0.0036
115	0.0035
116	0.0037
117	0.0036
118	0.0033
119	0.0038
120	0.0039
121	0.0036
122	0.0035
123	0.0036
124	0.0035
125	0.0038
126	0.0038
127	0.0037
128	0.0035
129	0.0036
130	0.0035
131	0.0036
132	0.0037
133	0.0036
134	0.0036
135	0.0036
136	0.0039
137	0.0039
138	0.0037
139	0.0037
140	0.0037
141	0.0041
142	0.0036
143	0.0034
144	0.0036
145	0.0039
146	0.0036
147	0.0036

ANALYSIS

148	0.0034
149	0.0037



Number of Breaks	Time(s)
1	0.0031
2	0.0012
4	0.0003
8	0.0003
16	0.0004
32	0.0003
64	0.0003
128	0.0003
256	0.0004
512	0.0004
1024	0.0009
2048	0.001
4096	0.0014
8192	0.0016
16384	0.0017
32768	0.0046
65536	0.0074

This is the graph and the data for running times of cutandSplice for linkStrand with varying number of breaks. The graph, especially for high numbers of breaks, clearly suggests that there is a linear relation between number of breaks and time of execution. R^2 value of 0.86 supports this conclusion. Thus both parts of the hypothesis are correct and the time complexity is O(b).

ANALYSIS

CODE FOR NONLINKED EXPERIMENT (NLLH):

```
import java.lang.*;
import java.util.*;
public class NonlinkedExperiment {
     public static void main(String args[]) {
          String original = "CGA";
          String evolved = "GGGTTTAAA";
          String repeatingPiece = "x";
          double dnaLength = 6*Math.pow(10,5);
          ArrayList<String> StringCollection = new
ArrayList<String>(); //b is equal to index*2
          for(int b=1; b<Math.pow(10, 5); b*=2) {
               StringBuilder dna = new StringBuilder();
               ArrayList<Integer> randomList = new
ArrayList<Integer>();
               for(int i=0; i<b;i++) {
                    randomList.add(1);
               }
               for(int i=0; i<(dnaLength-
b*original.length());i++) {
                    randomList.add(0);
               }
               Collections.shuffle(randomList);
               for(int i=0;i<((dnaLength-
b*original.length())+b);i++) {
                    if(randomList.get(i)==1) {
                         dna.append(original);
                    }
                    else {
                         dna.append(repeatingPiece);
                    }
               }
               StringCollection.add(dna.toString());

          }

          System.out.println("StringStrand");
          for(int i=0; i<StringCollection.size();i++) {
               int bNo = (int) Math.pow(2, i);
               int NumTrials = 4;
               String dnaString = StringCollection.get(i);
               double totalTime = 0.0;
```

```
                for(int trialNo=0; trialNo<NumTrials;
trialNo++) {
                     IDnaStrand strand = new
StringStrand(dnaString);
                    double start = System.nanoTime();
                    strand.cutAndSplice(original, evolved);
                    totalTime += (System.nanoTime() - start) /
1e9;
                }
                totalTime = totalTime/NumTrials;
                System.out.printf("%6d\t%6.4f\n", bNo,
totalTime);

           }

           System.out.println("");
           System.out.println("StringBuilderStrand");
           for(int i=0; i<StringCollection.size();i++) {
                int bNo = (int) Math.pow(2, i);
                int NumTrials = 4;
                String dnaString = StringCollection.get(i);
                double totalTime = 0.0;
                for(int trialNo=0; trialNo<NumTrials;
trialNo++) {
                     IDnaStrand strand = new
StringBuilderStrand(dnaString);
                    double start = System.nanoTime();
                    strand.cutAndSplice(original, evolved);
                    totalTime += (System.nanoTime() - start) /
1e9;
                }
                totalTime = totalTime/NumTrials;
                System.out.printf("%6d\t%6.4f\n", bNo,
totalTime);
           }

     }


}
```

ANALYSIS

CODE FOR LINKSTRAND EXPERIMENT (LSH):

```
import java.util.ArrayList;
import java.util.Collections;

public class LinkStrandExperiment {
	public static void main(String args[]) {
		String original = "CGA";
		String evolved = "GGGTTTAAA";
		String repeatingPiece = "x";
		double dnaLength = 6*Math.pow(10,5);
		ArrayList<String> StringCollection = new
ArrayList<String>(); //b is equal to index*2
		for(int b=1; b<Math.pow(10, 5); b*=2) {
			StringBuilder dna = new StringBuilder();
			ArrayList<Integer> randomList = new
ArrayList<Integer>();
			for(int i=0; i<b;i++) {
				randomList.add(1);
			}
			for(int i=0; i<(dnaLength-b*original.length());i++)
{
				randomList.add(0);
			}
			Collections.shuffle(randomList);
			for(int i=0;i<((dnaLength-
b*original.length())+b);i++) {
				if(randomList.get(i)==1) {
					dna.append(original);
				}
				else {
					dna.append(repeatingPiece);
				}
			}
			StringCollection.add(dna.toString());

		}

		System.out.println("LinkStrand");
		for(int i=0; i<StringCollection.size();i++) {
			int bNo = (int) Math.pow(2, i);
			int NumTrials = 4;
			String dnaString = StringCollection.get(i);
			double totalTime = 0.0;
			for(int trialNo=0; trialNo<NumTrials; trialNo++) {
				IDnaStrand strand = new LinkStrand(dnaString);
			    double start = System.nanoTime();
```

```
                    strand.cutAndSplice(original, evolved);
                    totalTime += (System.nanoTime() - start) / 1e9;
                }
                totalTime = totalTime/NumTrials;
                System.out.printf("%6d\t%6.4f\n", bNo, totalTime);

            }
            String evolved2 =
"AAGGGTTTAAAGGTATGATGATGATAGATGGAAAATTTTGGAGATGAGTA";
            String longEvolved = evolved2+evolved2+evolved2;
            System.out.println("");
            System.out.println("StringBuilderStrand Changing
Splice");
            for(int k=1; k<longEvolved.length();k++) {
                String evolvedInp = longEvolved.substring(0, k);

                for(int i=0; i<StringCollection.size();i++) {
                    int bNo = (int) Math.pow(2, i);
                    int NumTrials = 4;
                    String dnaString = StringCollection.get(i);
                    double totalTime = 0.0;
                    for(int trialNo=0; trialNo<NumTrials;
trialNo++) {
                        IDnaStrand strand = new
LinkStrand(dnaString);
                        double start = System.nanoTime();
                        strand.cutAndSplice(original, evolvedInp);
                        totalTime += (System.nanoTime() - start) /
1e9;
                    }
                    totalTime = totalTime/NumTrials;
                    if(bNo==65536) {
                        System.out.printf("%6d\t%6.4f\n", k,
totalTime);
                    }
                }
            }


    }
}
```

ANALYSIS

EXPLANATION OF THE CODE(NLLH):

Part 1:
The code starts by defining the string blocks of the DNAs, and the constants that will be used in the program. One of the constants is N, lengths of the DNAs. Using this constant, program creates DNAs with different numbers of breaks, and 'x' strings(used to represent other unchanged parts of the dna) that adds up to N. To randomize the distribution of enzymes, first a list with 1s and 0s are constructed with each 1 corresponding to one enzyme and 0s are corresponding to the 'x' strings. Then this list is shuffled and DNA string is constructed with using this random order of the list. Then each DNA with different numbers of breaks are collected in a string list. This first part might seem confusing, however what is basically accomplished is DNA strings with randomly distributed different number of breaks are created and stored for later use in the code.

Part 2:
In part two, for String Builder Strand and String Strand, for strings with different breaks the time it takes for cutandSplice is calculated through taking average of total time it takes for running it four times. The results are printed out for both with corresponding b numbers. The strings with different breaks are called from the list they were stored in part1 and are used as dna input for the cutandSplice command.

EXPLANATION OF THE CODE(LSH):

Part 1:
Part 1 is exactly same with the part 1 of NLLH.

Part 2:
In part 2, the time it takes for Link Strand to execute cutAndSplice command for varying number of breaks are calculated in the same way as in part 2 of NLLH.

Part 3:
In part 3, a long string is created with 149 characters. Then with using a for loop and substring different lengths of this string is used as splice in LinkStrand. With printing out only results where b=65536, the trend of LinkStrand's time complexity that depends on the length of splice is outputed to the data.

ANALYSIS

BENCHMARK RESULTS:

STRINGSTRAND:

```
dna length = 4,639,221
cutting at enzyme gaattc
-----
Class                 splicee          recomb     time  appends
-----
StringStrand:               256       4,800,471  5.037 1290
StringStrand:               512       4,965,591  5.222 1290
StringStrand:             1,024       5,295,831  5.654 1290
StringStrand:             2,048       5,956,311  7.886 1290
StringStrand:             4,096       7,277,271  7.051 1290
StringStrand:             8,192       9,919,191  9.403 1290
StringStrand:            16,384      15,203,031  16.420     1290
StringStrand:            32,768      25,770,711  34.080     1290
StringStrand:            65,536      46,906,071  91.667     1290
StringStrand:           131,072      89,176,791  161.418    1290
StringStrand:           262,144     173,718,231  328.351    1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap
space
      at java.util.Arrays.copyOf(Arrays.java:3332)
      at
java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStrin
gBuilder.java:124)
      at
java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:44
8)
      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)
```

STRINGBUILDERSTRAND

```
dna length = 4,639,221
cutting at enzyme gaattc
-----
Class                 splicee          recomb     time  appends
-----
StringBuilderStrand:            256       4,800,471  0.058 1290
```

ANALYSIS

```
StringBuilderStrand:              512        4,965,591  0.049 1290
StringBuilderStrand:            1,024        5,295,831  0.073 1290
StringBuilderStrand:            2,048        5,956,311  0.033 1290
StringBuilderStrand:            4,096        7,277,271  0.050 1290
StringBuilderStrand:            8,192        9,919,191  0.037 1290
StringBuilderStrand:           16,384       15,203,031  0.080 1290
StringBuilderStrand:           32,768       25,770,711  0.095 1290
StringBuilderStrand:           65,536       46,906,071  0.214 1290
StringBuilderStrand:          131,072       89,176,791  0.361 1290
StringBuilderStrand:          262,144      173,718,231  0.512 1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap
space
      at java.util.Arrays.copyOf(Arrays.java:3332)
      at
java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStrin
gBuilder.java:124)
      at
java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:44
8)
      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)


LINKSTRAND

dna length = 4,639,221
cutting at enzyme gaattc
-----
Class                splicee          recomb     time  appends
-----
LinkStrand:              256        4,800,471  0.024 1290
LinkStrand:              512        4,965,591  0.027 1290
LinkStrand:            1,024        5,295,831  0.034 1290
LinkStrand:            2,048        5,956,311  0.028 1290
LinkStrand:            4,096        7,277,271  0.022 1290
LinkStrand:            8,192        9,919,191  0.023 1290
LinkStrand:           16,384       15,203,031  0.023 1290
LinkStrand:           32,768       25,770,711  0.023 1290
LinkStrand:           65,536       46,906,071  0.027 1290
LinkStrand:          131,072       89,176,791  0.022 1290
LinkStrand:          262,144      173,718,231  0.021 1290
LinkStrand:          524,288      342,801,111  0.022 1290
LinkStrand:        1,048,576      680,966,871  0.023 1290
LinkStrand:        2,097,152    1,357,298,391  0.030 1290
```

ANALYSIS

```
LinkStrand:            4,194,304    2,709,961,431  0.027 1290
LinkStrand:            8,388,608    5,415,287,511  0.022 1290
LinkStrand:           16,777,216   10,825,939,671  0.025 1290
LinkStrand:           33,554,432   21,647,243,991  0.021 1290
LinkStrand:           67,108,864   43,289,852,631  0.021 1290
LinkStrand:          134,217,728   86,575,069,911  0.023 1290
LinkStrand:          268,435,456  173,145,504,471  0.021 1290
Exception in thread "main" java.lang.OutOfMemoryError: Java heap
space
      at java.util.Arrays.copyOf(Arrays.java:3332)
      at
java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStrin
gBuilder.java:124)
      at
java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:44
8)
      at java.lang.StringBuilder.append(StringBuilder.java:136)
      at DNABenchmark.main(DNABenchmark.java:107)
```