Joe Cusano
Jgc28

StringBuilderStrand

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

Class	splicee	recomb	time	appends
StringBuilderStrand:	256	4,800,471	0.025	1290
StringBuilderStrand:	512	4,965,591	0.026	1290
StringBuilderStrand:	1,024	5,295,831	0.025	1290
StringBuilderStrand:	2,048	5,956,311	0.024	1290
StringBuilderStrand:	4,096	7,277,271	0.026	1290
StringBuilderStrand:	8,192	9,919,191	0.022	1290
StringBuilderStrand:	16,384	15,203,031	0.024	1290
StringBuilderStrand:	32,768	25,770,711	0.049	1290
StringBuilderStrand:	65,536	46,906,071	0.020	1290
StringBuilderStrand:	131,072	89,176,791	0.098	1290
StringBuilderStrand:	262,144	173,718,231	0.243	1290
StringBuilderStrand:	524,288	342,801,111	0.271	1290

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

String Strand

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

Class	splicee	recomb	time	appends
StringStrand:	256	4,800,471	0.353	1290
StringStrand:	512	4,965,591	0.323	1290
StringStrand:	1,024	5,295,831	0.336	1290
StringStrand:	2,048	5,956,311	0.377	1290
StringStrand:	4,096	7,277,271	0.478	1290
StringStrand:	8,192	9,919,191	0.673	1290
StringStrand:	16,384	15,203,031	1.092	1290
StringStrand:	32,768	25,770,711	1.932	1290
StringStrand:	65,536	46,906,071	3.726	1290
StringStrand:	131,072	89,176,791	8.703	1290
StringStrand:	262,144	173,718,231	20.217	1290
StringStrand:	524,288	342,801,111	41.024	1290

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

Link Strand

Class	splicee	recomb	time	appends
LinkStrand:	256	4,800,471	0.021	1290
LinkStrand:	512	4,965,591	0.015	1290
LinkStrand:	1,024	5,295,831	0.003	1290
LinkStrand:	2,048	5,956,311	0.004	1290
LinkStrand:	4,096	7,277,271	0.004	1290
LinkStrand:	8,192	9,919,191	0.005	1290
LinkStrand:	16,384	15,203,031	0.003	1290
LinkStrand:	32,768	25,770,711	0.003	1290
LinkStrand:	65,536	46,906,071	0.004	1290
LinkStrand:	131,072	89,176,791	0.003	1290
LinkStrand:	262,144	173,718,231	0.003	1290
LinkStrand:	524,288	342,801,111	0.003	1290
LinkStrand:	1,048,576	680,966,871	0.004	1290
LinkStrand:	2,097,152	1,357,298,391	0.006	1290
LinkStrand:	4,194,304	2,709,961,431	0.007	1290
LinkStrand:	8,388,608	5,415,287,511	0.004	1290
LinkStrand:	16,777,216	10,825,939,671	0.007	1290
LinkStrand:	33,554,432	21,647,243,991	0.004	1290
LinkStrand:	67,108,864	43,289,852,631	0.007	1290
LinkStrand:	134,217,728	86,575,069,911	0.004	1290
LinkStrand:	268,435,456	173,145,504,471	0.005	1290
LinkStrand:	536,870,912	346,286,373,591	0.008	1290

Question 1: ***are the benchmark timings for `StringStrand` consistent*** with the explanation below that the time to execute cutAndSplice is $O(b^2S)$?

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 b^2 will be constant in the benchmark, but S will vary.

The benchmark timings are consistent with the $O(B^2S)$ runtime explanation for *StringStrand*. As you double the splicee (S), the time doubles as well, which evidences this runtime explanation. For example, when splicee increases from 16,384 to 32,768, time increases from 1.092 to 1.932.

B^2 does not affect runtime in this istance because it remains constant.

Question 2: are the benchmark timings for **`StringBuilderStrand`** consistent with the explanation below that the time to execute cutAndSplice is $O(bS)$?

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 **`b`** will be constant in the benchmark, but S will vary.

The benchmark timings for StringBuilderStrand evidence the runtime complexity being O(bS). b is constant in the benchmark so will not affect runtime. S does not have a large affect on timings when splicee is less than 70,000 because of O(bS). For example, when splicee increases from 32,768 to 65,536, time actually drops from .049 to .020.

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 at the specific strand size you see in your timings -- as compared to exhaustion for String and StringBuilder.

Run times do not vary because the size of the splicee is not significant because it can be linked via a new node. For example, when splicee increases from 33,554,432 to 67,108,864 time stays very constant, going from .004 to .007.

Memory is exhausted at a larger strand size because LinkStrand uses less memory than StringStrand and StringBuilderStrand. In LinkStrand, Nodes can point to the same splicee for the same values, and do not have to create new nodes if nodes have the same value.