Helen Xu

P5 Analysis

1. **BenchmarkForAutocomplete**

"threeletterwords.txt"

init time: 0.004082 for BruteAutocomplete
init time: 0.003972 for BinarySearchAutocomplete
init time: 0.07922 for HashListAutocomplete

search	size	#match	BruteAutoc	BinarySear	HashListAu
	17576	50	0.00484380	0.00814940	0.00009360
	17576	50	0.00124760	0.00440780	0.00001010
a	676	50	0.00089790	0.00034720	0.00000820
a	676	50	0.00095700	0.00029830	0.00000780
b	676	50	0.00073110	0.00029360	0.00000710
c	676	50	0.00055320	0.00016010	0.00000610
g	676	50	0.00049220	0.00022470	0.00001070
ga	26	50	0.00031540	0.00004080	0.00000500
go	26	50	0.00039300	0.00009360	0.00000530
gu	26	50	0.00049220	0.00008030	0.00000630
x	676	50	0.00023670	0.00014880	0.00000550
y	676	50	0.00025670	0.00015430	0.00000600
z	676	50	0.00019010	0.00014720	0.00000550
aa	26	50	0.00016190	0.00005180	0.00000500
az	26	50	0.00022050	0.00003700	0.00000850
za	26	50	0.00029780	0.00004840	0.00000670
zz	26	50	0.00028970	0.00004580	0.00000690
zqzqwwx	0	50	0.00014040	0.00003650	0.00000280

size in bytes=246064 for BruteAutocomplete
size in bytes=246064 for BinarySearchAutocomplete
size in bytes=676268 for HashListAutocomplete

“fourletterwords.txt”

init time: 0.04179 for BruteAutocomplete
init time: 0.02593 for BinarySearchAutocomplete
init time: 0.6866 for HashListAutocomplete

search	size	#match	BruteAutoc	BinarySear	HashListAu
	456976	50	0.00861850	0.02696840	0.00008870
	456976	50	0.00375610	0.00429030	0.00000790
a	17576	50	0.00384830	0.00028230	0.00000780
a	17576	50	0.00408630	0.00031780	0.00000770
b	17576	50	0.00342690	0.00024430	0.00000770
c	17576	50	0.00336050	0.00028080	0.00000900
g	17576	50	0.00335490	0.00024810	0.00000820
ga	676	50	0.00332570	0.00009290	0.00000620

go	676	50	0.00329790	0.00006950	0.00000580
gu	676	50	0.00351090	0.00007770	0.00000720
x	17576	50	0.00354880	0.00033920	0.00000830
y	17576	50	0.00353690	0.00027210	0.00000710
z	17576	50	0.00345260	0.00021670	0.00000820
aa	676	50	0.00333590	0.00006740	0.00000710
az	676	50	0.00344910	0.00007080	0.00001360
za	676	50	0.00347230	0.00007840	0.00000780
zz	676	50	0.00356520	0.00007470	0.00000790
zqzqwwx	0	50	0.00302640	0.00008730	0.00000350

size in bytes=7311616 for BruteAutocomplete
size in bytes=7311616 for BinarySearchAutocomplete
size in bytes=25845100 for HashListAutocomplete

“alexa.txt”

init time: 0.3698 for BruteAutocomplete
init time: 1.122 for BinarySearchAutocomplete
init time: 4.617 for HashListAutocomplete

search	size	#match	BruteAutoc	BinarySear	HashListAu
	1000000	50	0.01946120	0.06000880	0.00008960
	1000000	50	0.02483850	0.05983370	0.00001210
a	69464	50	0.01486160	0.00422800	0.00001300
a	69464	50	0.01011910	0.00230960	0.00000980
b	56037	50	0.00952170	0.00197510	0.00001080
c	65842	50	0.02736430	0.00560020	0.00006870
g	37792	50	0.01879050	0.00271520	0.00001540
ga	6664	50	0.01462720	0.00046710	0.00000970
go	6953	50	0.01811570	0.00072160	0.00001290
gu	2782	50	0.01680750	0.00039870	0.00001100
x	6717	50	0.01690610	0.00066060	0.00001240
y	16765	50	0.02041210	0.00141570	0.00001360
z	8780	50	0.01395200	0.00068370	0.00001120
aa	718	50	0.01190900	0.00011570	0.00000840
az	889	50	0.01302210	0.00014000	0.00001420
za	1718	50	0.01311580	0.00029180	0.00001060
zz	162	50	0.01456540	0.00011220	0.00001110
zqzqwwx	0	50	0.01443090	0.00015660	0.00000660

size in bytes=38204230 for BruteAutocomplete
size in bytes=38204230 for BinarySearchAutocomplete
size in bytes=420937488 for HashListAutocomplete

2. **“alexa.txt” with matches=10000**

init time: 0.2704 for BruteAutocomplete
init time: 1.335 for BinarySearchAutocomplete

init time: 4.569 for HashListAutocomplete

search	size	#match	BruteAutoc	BinarySear	HashListAu
	1000000	10000	0.02678960	0.10658590	0.00012680
	1000000	10000	0.02495640	0.08058850	0.00001240
a	69464	10000	0.01966340	0.01910780	0.00001140
a	69464	10000	0.01748440	0.01894020	0.00001610
b	56037	10000	0.01750380	0.01704310	0.00001100
c	65842	10000	0.01722790	0.01852680	0.00001240
g	37792	10000	0.02249450	0.01482650	0.00001120
ga	6664	10000	0.01672510	0.00355300	0.00000850
go	6953	10000	0.01919690	0.00356080	0.00000930
gu	2782	10000	0.01409620	0.00148480	0.00001090
x	6717	10000	0.01523940	0.00355240	0.00001020
y	16765	10000	0.01833150	0.00856630	0.00001110
z	8780	10000	0.01671350	0.00463720	0.00001040
aa	718	10000	0.01244940	0.00032870	0.00000810
az	889	10000	0.01363790	0.00041440	0.00001350
za	1718	10000	0.01224800	0.00082340	0.00000880
zz	162	10000	0.01138860	0.00010080	0.00000680
zqzqwwx	0	10000	0.01159300	0.00010370	0.00000430

size in bytes=38204230 for BruteAutocomplete
size in bytes=38204230 for BinarySearchAutocomplete
size in bytes=420937488 for HashListAutocomplete

The number of matches seems to have the largest effect on the runtimes of BinarySearch, especially at the beginning, when the runtime was noticeably longer with the altered match size of 10000, but other than that, especially for BruteAutocomplete and HashListAutocomplete, match size does not seem to have too much of an effect on runtime.

3. BruteAutocomplete.topMatches() uses a LinkedList rather than an ArrayList because it is more efficient to use a LinkedList when adding to the front. In an ArrayList, adding to the front involves shifting all the elements back, while adding to the front of a LinkedList is an O(1) operation. The PriorityQueue uses Comparator.comparing(Term::getWeight) to get the top k heaviest matches since we want the top k heaviest elements in the priority queue of size k, so by sorting in increasing order of weight, each time we need to add a new element to the priority queue, we can simply call pq.remove() to remove the smallest element, leaving us with the top k heaviest elements in the priority queue.

4. HashListAutocomplete uses more memory than the other implementations because in initialize(), all possible prefixes are mapped to terms to save the time of having to recalculate all over again each time. Since each combination is stored in the instance variable myMap, HashListAutocomplete takes up more memory in exchange for having a faster runtime.