Kenneth Marenco

Analysis for Autocomplete

1.

a) threeletterwords

init time: 0.01328 for BruteAutocomplete

init time: 0.03017 for BinarySearchAutocomplete

init time: 0.1000 for HashListAutocomplete

search size #match BruteAutoc BinarySear HashListAu

17576 50 0.00786930 0.15370600 0.00039880

17576 50 0.00100740 0.00289650 0.00001370

a 676 50 0.00066460 0.00030100 0.00000860

a 676 50 0.00067560 0.00037170 0.00000880

b 676 50 0.00071980 0.00037610 0.00000880

c 676 50 0.00069220 0.00033030 0.00001940

g 676 50 0.00065970 0.00036320 0.00000780

ga 26 50 0.00072090 0.00103710 0.00000920

go 26 50 0.00055380 0.00016130 0.00001510

gu 26 50 0.00062240 0.00011970 0.00000760

x 676 50 0.00065850 0.00024170 0.00000850

y 676 50 0.00040060 0.00030770 0.00000820

z 676 50 0.00064180 0.00033970 0.00000850

aa 26 50 0.00075200 0.00006890 0.00000780

az 26 50 0.00031810 0.00007450 0.00000740

za 26 50 0.00032680 0.00006060 0.00000770

zz 26 50 0.00031820 0.00006830 0.00000760

zqzqwwx 0 50 0.00029910 0.00019670 0.00000380

size in bytes=246064 for BruteAutocomplete

size in bytes=246064 for BinarySearchAutocomplete

size in bytes=354276 for HashListAutocomplete

b) fourletterwords

init time: 0.08043 for BruteAutocomplete

init time: 0.04928 for BinarySearchAutocomplete

init time: 1.554 for HashListAutocomplete

search size #match BruteAutoc BinarySear HashListAu

456976 50 0.01350080 0.03234990 0.00054690

456976 50 0.00780780 0.00361190 0.00001060

a 17576 50 0.00814780 0.00030790 0.00002240

a 17576 50 0.00811290 0.00026490 0.00000880

b 17576 50 0.00614970 0.00037590 0.00000930

c 17576 50 0.00556700 0.00034190 0.00001060

g 17576 50 0.00556610 0.00036580 0.00000910

ga 676 50 0.00595110 0.00012910 0.00000790

go 676 50 0.00533280 0.00010590 0.00000990

gu 676 50 0.00705860 0.00016430 0.00000910

x 17576 50 0.00577760 0.00040260 0.00001060

y 17576 50 0.00554220 0.00031440 0.00001040

z 17576 50 0.00562680 0.00031980 0.00001020

aa 676 50 0.00630170 0.00008470 0.00000910

az 676 50 0.00602250 0.00020250 0.00001170

za 676 50 0.00738590 0.00011380 0.00000840

zz 676 50 0.00538320 0.00011280 0.00000960

zqzqwwx 0 50 0.00632420 0.00007870 0.00000390

size in bytes=7311616 for BruteAutocomplete

size in bytes=7311616 for BinarySearchAutocomplete

size in bytes=11075636 for HashListAutocomplete

c) alexa

init time: 0.7075 for BruteAutocomplete

init time: 3.280 for BinarySearchAutocomplete

init time: 12.61 for HashListAutocomplete

search size #match BruteAutoc BinarySear HashListAu

1000000 50 0.05207380 0.05828170 0.00056050

1000000 50 0.02050500 0.01551550 0.00001150

a 69464 50 0.01796410 0.00148380 0.00002840

a 69464 50 0.01837390 0.00139500 0.00001180

b 56037 50 0.01804490 0.00113730 0.00001150

c 65842 50 0.01809420 0.00137060 0.00001100

g 37792 50 0.01758390 0.00093550 0.00001180

ga 6664 50 0.01786370 0.00035410 0.00001090

go 6953 50 0.01636350 0.00034340 0.00001130

gu 2782 50 0.02937210 0.00023170 0.00001130

x 6717 50 0.01758530 0.00034150 0.00001130

y 16765 50 0.01817630 0.00050070 0.00001140

z 8780 50 0.01785660 0.00024810 0.00000830

aa 718 50 0.01569360 0.00010710 0.00000960

az 889 50 0.01744080 0.00010750 0.00000900

za 1718 50 0.01339650 0.00014110 0.00000880

zz 162 50 0.01339860 0.00006860 0.00000880

zqzqwwx 0 50 0.01554010 0.00010470 0.00000430

size in bytes=38204230 for BruteAutocomplete

size in bytes=38204230 for BinarySearchAutocomplete

size in bytes=98824414 for HashListAutocomplete

2.

Below you will see the results of running BenchMark again with a match # of 10,000. The initialization time for all the autocomplete methods decreased with the increase in match #. However, by analyzing the results for each search, we can see that the runtime for each autocomplete class increased, excluding HashListAutocomplete. This is due to increased size of match #, but since HashList utilizes a HashMap, it still runs at constant time leading to similar times as the 50 #match.

init time: 0.4725 for BruteAutocomplete

init time: 2.392 for BinarySearchAutocomplete

init time: 8.867 for HashListAutocomplete

search size #match BruteAutoc BinarySear HashListAu

1000000 10000 0.04130800 0.10879450 0.00036810

1000000 10000 0.03010290 0.10176520 0.00001820

a 69464 10000 0.02254260 0.02095140 0.00002420

a 69464 10000 0.02229380 0.02077950 0.00000990

b 56037 10000 0.02339100 0.02128730 0.00001320

c 65842 10000 0.02282040 0.02165050 0.00001130

g 37792 10000 0.02274580 0.01615910 0.00002980

ga 6664 10000 0.02077850 0.00419140 0.00000980

go 6953 10000 0.02002010 0.00443720 0.00000960

gu 2782 10000 0.01730320 0.00157350 0.00000890

x 6717 10000 0.01996200 0.00434390 0.00000900

y 16765 10000 0.02373090 0.01023510 0.00001080

z 8780 10000 0.02469150 0.00583820 0.00000860

aa 718 10000 0.01658310 0.00035840 0.00000830

az 889 10000 0.01601470 0.00047380 0.00000840

za 1718 10000 0.01667860 0.00089930 0.00000970

zz 162 10000 0.01482280 0.00008660 0.00000810

zqzqwwx 0 10000 0.01757520 0.00011420 0.00000420

size in bytes=38204230 for BruteAutocomplete

size in bytes=38204230 for BinarySearchAutocomplete

size in bytes=98824414 for HashListAutocomplete

3.

The BruteAutocomplete.topMatches uses a LinkedList because it requires less memory and allows constant time insertions and removals while an ArrayList would require much more run time when adding or removing elements or shifting elements. The reason to use the PriorityQueue with Camparator.comparing(Term::getWeight) is to sort the queue with elements with the highest weights to one end that way it would be simple task to extract the k heaviest matches.

4.

HashListAutocomplete uses a HashMap which intrinsically costs more memory to store keys with a corresponding value (in this case, ArrayList<Term>). This is vital when trying to reduce runtime since finding values for maps are constant time.