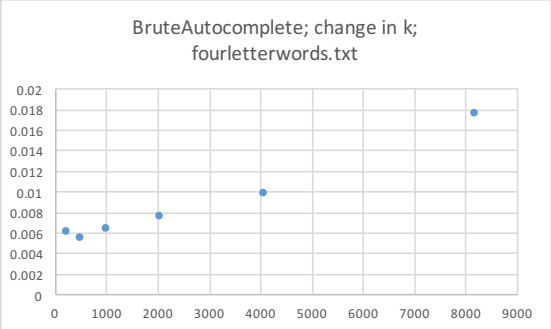
BruteAutocomplete

fourletterwords.txt

```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwords.txt.
Benchmarking BruteAutocomplete...
Found 456976 words
Time to initialize - 0.139636311
Time for topMatch("") - 0.001795190781
Time for topMatch("nenk") - 0.004442384037
Time for topMatch("n") - 0.003564854281
Time for topMatch("ne") - 0.003346401787
Time for topMatch("notarealword") - 0.003252189904
Time for topKMatches("", 256) -  0.00594049599
Time for topKMatches("", 512) -  0.00546317382
Time for topKMatches("", 1024) -  0.00625415295
Time for topKMatches("", 2048) -  0.00750980926
Time for topKMatches("", 4096) -  0.00975125786
Time for topKMatches("", 8192) -  0.01753570366
Time for topKMatches("nenk", 256) -  0.00546092532
Time for topKMatches("nenk", 512) -  0.0055313072
Time for topKMatches("nenk", 1024) -  0.00435245438
Time for topKMatches("nenk", 2048) -  0.0045462009
Time for topKMatches("nenk", 4096) -  0.00423897464
Time for topKMatches("nenk", 8192) -  0.00420877921
Time for topKMatches("n", 256) -  0.00435825538
Time for topKMatches("n", 512) -  0.0044600049
Time for topKMatches("n", 1024) -  0.00486548734
Time for topKMatches("n", 2048) -  0.00523481566
Time for topKMatches("n", 4096) -  0.00610729347
Time for topKMatches("n", 8192) -  0.00714070216
Time for topKMatches("ne", 256) -  0.0043002058
Time for topKMatches("ne", 512) -  0.00424523068
Time for topKMatches("ne", 1024) -  0.00421960122
Time for topKMatches("ne", 2048) -  0.00426125638
Time for topKMatches("ne", 4096) -  0.00423289882
Time for topKMatches("ne", 8192) -  0.00425980759
Time for topKMatches("notarealword", 256) -  0.0035369897
Time for topKMatches("notarealword", 512) -  0.0035961971
Time for topKMatches("notarealword", 1024) -  0.00350650262
Time for topKMatches("notarealword", 2048) -  0.00347382649
Time for topKMatches("notarealword", 4096) -  0.00344580295
Time for topKMatches("notarealword", 8192) -  0.00345118817
```



If we have n terms, m of which start with the prefix, then O(N) + O(M log M). We have big oh log m instead of log k. Based on this data, as k increases by a factor of two, runtime increases, but no clear pattern emerges. The final data point for k = 8192 may be an outlier, because it marks the biggest increase in runtime. It is likely that this is a linear relationship. K does not appear to have drastic effect on the runtime, which makes

sense because runtime is determined by all m terms based on the prefix, not just the k size delimiter for brute autocomplete.

Additionally, as the prefix size increases, there is no drastic change in runtimes based on these data.

fourletterwordshalf.txt

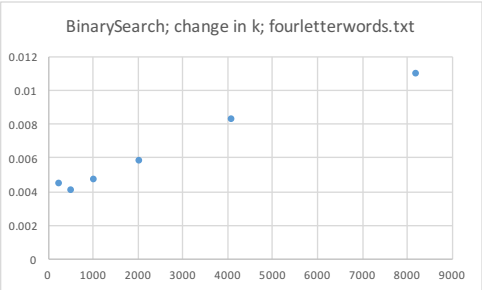
```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwordshalf.txt.
Benchmarking BruteAutocomplete...
Found 228488 words
Time to initialize - 0.046385762
Time for topMatch("") - 9.12525081E-4
Time for topMatch("aenk") - 0.003741430869
Time for topMatch("a") - 0.001142871382
Time for topMatch("ae") - 9.1811557E-4
Time for topMatch("notarealword") - 0.002473942537
Time for topKMatches("", 256) -  0.00441283952
Time for topKMatches("", 512) -  0.00403461888
Time for topKMatches("", 1024) -  0.00464842211
Time for topKMatches("", 2048) -  0.0057881786
Time for topKMatches("", 4096) -  0.00822307799
Time for topKMatches("", 8192) -  0.01091386766
Time for topKMatches("aenk", 256) -  0.00273239812
Time for topKMatches("aenk", 512) -  0.00216050752
Time for topKMatches("aenk", 1024) -  0.00230426825
Time for topKMatches("aenk", 2048) -  0.00217278667
Time for topKMatches("aenk", 4096) -  0.00222978161
Time for topKMatches("aenk", 8192) -  0.00214216372
Time for topKMatches("a", 256) -  0.00226927095
Time for topKMatches("a", 512) -  0.00246535643
Time for topKMatches("a", 1024) -  0.00271210309
Time for topKMatches("a", 2048) -  0.00314467692
Time for topKMatches("a", 4096) -  0.00382730629
Time for topKMatches("a", 8192) -  0.00488751689
Time for topKMatches("ae", 256) -  0.00217259869
Time for topKMatches("ae", 512) -  0.00217099892
Time for topKMatches("ae", 1024) -  0.00226904048
Time for topKMatches("ae", 2048) -  0.00224298261
Time for topKMatches("ae", 4096) -  0.00219752789
Time for topKMatches("ae", 8192) -  0.00221914051
Time for topKMatches("notarealword", 256) -  0.00181407188
Time for topKMatches("notarealword", 512) -  0.00191625548
Time for topKMatches("notarealword", 1024) -  0.00175502679
Time for topKMatches("notarealword", 2048) -  0.00186323674
Time for topKMatches("notarealword", 4096) -  0.00175119035
Time for topKMatches("notarealword", 8192) -  0.00183187685
```

In the data file with N/2 words, runtime decreases. This makes sense one aspect of the runtime for BruteAutocomplete should be O(n), so naturally a decrease in n will decrease runtime at these small n values.

BinarySearchTree

fourletterwords.txt

```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwords.txt.
Benchmarking BinarySearchAutocomplete...
Found 456976 words
Time to initialize - 0.05810979
Time for topMatch("") - 7.82747784E-4
Time for topMatch("nenk") - 1.872503E-6
Time for topMatch("n") - 1.9931332E-5
Time for topMatch("ne") - 2.453617E-6
Time for topMatch("notarealword") - 4.884324E-6
Time for topKMatches("", 256) -  0.03836013143
Time for topKMatches("", 512) -  0.04189562787
Time for topKMatches("", 1024) -  0.04675649246
Time for topKMatches("", 2048) -  0.05040715356
Time for topKMatches("", 4096) -  0.0501919282
Time for topKMatches("", 8192) -  0.05056349329
Time for topKMatches("nenk", 256) -  1.35092E-6
Time for topKMatches("nenk", 512) -  8.5751E-7
Time for topKMatches("nenk", 1024) -  9.9394E-7
Time for topKMatches("nenk", 2048) -  1.35261E-6
Time for topKMatches("nenk", 4096) -  1.85225E-6
Time for topKMatches("nenk", 8192) -  3.21764E-6
Time for topKMatches("n", 256) -  0.00143765935
Time for topKMatches("n", 512) -  0.00169654414
Time for topKMatches("n", 1024) -  0.00199537704
Time for topKMatches("n", 2048) -  0.00231012112
Time for topKMatches("n", 4096) -  0.00285268311
Time for topKMatches("n", 8192) -  0.00345132793
Time for topKMatches("ne", 256) -  4.893397E-5
Time for topKMatches("ne", 512) -  6.413663E-5
Time for topKMatches("ne", 1024) -  7.416234E-5
Time for topKMatches("ne", 2048) -  7.315405E-5
Time for topKMatches("ne", 4096) -  7.361065E-5
Time for topKMatches("ne", 8192) -  7.466884E-5
Time for topKMatches("notarealword", 256) -  1.28088E-6
Time for topKMatches("notarealword", 512) -  1.19888E-6
Time for topKMatches("notarealword", 1024) -  2.0696E-6
Time for topKMatches("notarealword", 2048) -  1.46429E-6
Time for topKMatches("notarealword", 4096) -  1.89461E-6
Time for topKMatches("notarealword", 8192) -  3.00522E-6
```



These data for Binary Search show a more significant relationship between the size of k and runtime. Because binary search tree top matches uses a priority queue to delimit size (thus changing log m to log k where k is the size of the priority queue), the linear relationship between k and runtime makes sense. As k increases, runtime increases as determined by O(log N + M log M) for top matches given an initial sort of O(N log N).

fourletterwordshalf.txt

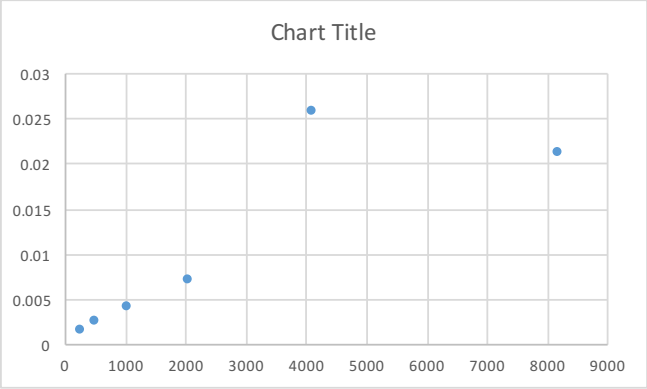
```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwordshalf.txt.
Benchmarking BinarySearchAutocomplete...
Found 228488 words
Time to initialize - 0.021347088
Time for topMatch("") - 3.94839666E-4
Time for topMatch("aenk") - 1.224401E-6
Time for topMatch("a") - 2.0619973E-5
Time for topMatch("ae") - 2.523001E-6
Time for topMatch("notarealword") - 4.821394E-6
Time for topKMatches("", 256) -  0.01842769274
Time for topKMatches("", 512) -  0.02004085981
Time for topKMatches("", 1024) -  0.02239847858
Time for topKMatches("", 2048) -  0.02477566162
Time for topKMatches("", 4096) -  0.02769993277
Time for topKMatches("", 8192) -  0.03541788837
Time for topKMatches("aenk", 256) -  1.50035E-6
Time for topKMatches("aenk", 512) -  7.7978E-7
Time for topKMatches("aenk", 1024) -  8.8391E-7
Time for topKMatches("aenk", 2048) -  1.19641E-6
Time for topKMatches("aenk", 4096) -  1.60188E-6
Time for topKMatches("aenk", 8192) -  2.91256E-6
Time for topKMatches("a", 256) -  0.00138467712
Time for topKMatches("a", 512) -  0.00164623072
Time for topKMatches("a", 1024) -  0.00190165802
Time for topKMatches("a", 2048) -  0.00213757446
Time for topKMatches("a", 4096) -  0.00269010451
Time for topKMatches("a", 8192) -  0.00326417048
Time for topKMatches("ae", 256) -  5.589775E-5
Time for topKMatches("ae", 512) -  7.804439E-5
Time for topKMatches("ae", 1024) -  7.781719E-5
Time for topKMatches("ae", 2048) -  7.489968E-5
Time for topKMatches("ae", 4096) -  7.592671E-5
Time for topKMatches("ae", 8192) -  7.656954E-5
Time for topKMatches("notarealword", 256) -  1.18417E-6
Time for topKMatches("notarealword", 512) -  9.58E-7
Time for topKMatches("notarealword", 1024) -  1.05577E-6
Time for topKMatches("notarealword", 2048) -  1.23465E-6
Time for topKMatches("notarealword", 4096) -  1.66245E-6
Time for topKMatches("notarealword", 8192) -  2.73734E-6
```

For n/2 words, runtime decreases based on this limited data, which follows the big oh expectations based on the log N portion of O(log N + M log M).

TrieAutocomplete

fourletterwords.txt

```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwords.txt.
Benchmarking TrieAutocomplete...
Found 456976 words
Time to initialize - 0.175467018
Created 475255 nodes
Time for topMatch("") - 6.170332E-6
Time for topMatch("nenk") - 5.70497E-7
Time for topMatch("n") - 3.769125E-6
Time for topMatch("ne") - 2.689533E-6
Time for topMatch("notarealword") - 5.43038E-7
Time for topKMatches("", 256) -  0.00155099338
Time for topKMatches("", 512) -  0.00266200835
Time for topKMatches("", 1024) -  0.00412311797
Time for topKMatches("", 2048) -  0.00710941492
Time for topKMatches("", 4096) -  0.02587571399
Time for topKMatches("", 8192) -  0.02127232868
Time for topKMatches("nenk", 256) -  3.12631E-6
Time for topKMatches("nenk", 512) -  1.40153E-6
Time for topKMatches("nenk", 1024) -  1.8836E-6
Time for topKMatches("nenk", 2048) -  1.8247E-6
Time for topKMatches("nenk", 4096) -  1.3188E-6
Time for topKMatches("nenk", 8192) -  2.66339E-6
Time for topKMatches("n", 256) -  3.1586033E-4
Time for topKMatches("n", 512) -  5.941852E-4
Time for topKMatches("n", 1024) -  0.00100696927
Time for topKMatches("n", 2048) -  0.00155059181
Time for topKMatches("n", 4096) -  0.00212178275
Time for topKMatches("n", 8192) -  0.00352521894
Time for topKMatches("ne", 256) -  4.100326E-5
Time for topKMatches("ne", 512) -  6.806586E-5
Time for topKMatches("ne", 1024) -  1.0309844E-4
Time for topKMatches("ne", 2048) -  8.73908E-5
Time for topKMatches("ne", 4096) -  8.646068E-5
Time for topKMatches("ne", 8192) -  8.761521E-5
Time for topKMatches("notarealword", 256) -  2.28543E-6
Time for topKMatches("notarealword", 512) -  7.1757E-7
Time for topKMatches("notarealword", 1024) -  7.4256E-7
Time for topKMatches("notarealword", 2048) -  8.1756E-7
Time for topKMatches("notarealword", 4096) -  7.7734E-7
Time for topKMatches("notarealword", 8192) -  7.6656E-7
```



Firstly, it appears that the data point for k = 4096 may be an outlier. The data follows a strict linear relationship except for this one datapoint, so for the purposes of my analysis, I will exclude it.

Once the trie is created, big oh is O(N) + O(M) for upper bound.

If k is smaller than m, which occurs in the data, then the trie can be more optimal than O(M)

The total number of nodes derives from the construction of the trie, which is O(NW), W length longest word

It is important to note that searching for a single word is o(w) in a trie…a benefit of the construction.

fourletterwordshalf.txt

```
Opening - /Users/joelmire/Documents/CS201/autocompletef17-start/data/fourletterwordshalf.txt.
Benchmarking TrieAutocomplete...
Found 228488 words
Time to initialize - 0.078995996
Created 237628 nodes
Time for topMatch("") - 6.114503E-6
Time for topMatch("aenk") - 5.54457E-7
Time for topMatch("a") - 5.872504E-6
Time for topMatch("ae") - 2.775911E-6
Time for topMatch("notarealword") - 1.85217E-7
Time for topKMatches("", 256) -  0.00127727189
Time for topKMatches("", 512) -  0.00153027438
Time for topKMatches("", 1024) -  0.00294334377
Time for topKMatches("", 2048) -  0.00522435614
Time for topKMatches("", 4096) -  0.00977420002
Time for topKMatches("", 8192) -  0.01687317473
Time for topKMatches("aenk", 256) -  2.82651E-6
Time for topKMatches("aenk", 512) -  2.15004E-6
Time for topKMatches("aenk", 1024) -  2.3308E-6
Time for topKMatches("aenk", 2048) -  2.10477E-6
Time for topKMatches("aenk", 4096) -  2.48941E-6
Time for topKMatches("aenk", 8192) -  2.13512E-6
Time for topKMatches("a", 256) -  3.3508479E-4
Time for topKMatches("a", 512) -  6.2121581E-4
Time for topKMatches("a", 1024) -  0.00103365385
Time for topKMatches("a", 2048) -  0.00170511765
Time for topKMatches("a", 4096) -  0.00227472772
Time for topKMatches("a", 8192) -  0.00329115974
Time for topKMatches("ae", 256) -  3.390159E-5
Time for topKMatches("ae", 512) -  6.160284E-5
Time for topKMatches("ae", 1024) -  7.768812E-5
Time for topKMatches("ae", 2048) -  7.787603E-5
Time for topKMatches("ae", 4096) -  7.873897E-5
Time for topKMatches("ae", 8192) -  7.889576E-5
Time for topKMatches("notarealword", 256) -  1.81808E-6
Time for topKMatches("notarealword", 512) -  5.4654E-7
Time for topKMatches("notarealword", 1024) -  5.2159E-7
Time for topKMatches("notarealword", 2048) -  4.8616E-7
Time for topKMatches("notarealword", 4096) -  4.5667E-7
Time for topKMatches("notarealword", 8192) -  5.251E-7
```

For n/2 words, there is not an observable significant difference in runtimes. I suspect that because tries have such fast runtimes, there is no way to determine empirical evidence that supports the big oh analysis at these N. Tries are so fast on the runtime side, but memory usage, which is not included in this analysis, is more interesting.