1.

Data Table:

Table 1

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking BinarySearchAutocomplete...
Found 333333 words
Time to initialize - 0.468793666
Time for topMatch("") - 0.001185248525
Time for topMatch("khombu") - 5.613142E-6
Time for topMatch("k") - 1.5383398E-5
Time for topMatch("kh") - 6.51359E-7
Time for topMatch("notarealword") - 1.520434E-6
Time for topKMatches("", 1) - 0.001483993748
Time for topKMatches("", 4) - 0.001462924784
Time for topKMatches("", 7) - 0.001459384324
Time for topKMatches("khombu", 1) - 6.20142E-7
Time for topKMatches("khombu", 4) - 6.80375E-7
Time for topKMatches("khombu", 7) - 5.87746E-7
Time for topKMatches("k", 1) - 3.3141085E-5
Time for topKMatches("k", 4) - 3.4781218E-5
Time for topKMatches("k", 7) - 3.5304194E-5
Time for topKMatches("kh", 1) - 1.721198E-6
Time for topKMatches("kh", 4) - 1.766072E-6
Time for topKMatches("kh", 7) - 1.837282E-6
Time for topKMatches("notarealword", 1) - 6.5922E-7
Time for topKMatches("notarealword", 4) - 6.07357E-7
Time for topKMatches("notarealword", 7) - 6.15525E-7

Table 2

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking BinarySearchAutocomplete...
Found 333333 words
Time to initialize - 0.463663873
Time for topMatch("") - 0.001417594329
Time for topMatch("khombu") - 5.767479E-6
Time for topMatch("k") - 3.8474597E-5
Time for topMatch("kh") - 8.02273E-7
Time for topMatch("notarealword") - 2.105794E-6
Time for topKMatches("", 1) - 0.002794196574
Time for topKMatches("", 4) - 0.003112180322
Time for topKMatches("", 7) - 0.0016499008
Time for topKMatches("", 10) - 0.001596722168
Time for topKMatches("", 13) - 0.001587941347
Time for topKMatches("", 16) - 0.001446606318

Time for topKMatches("", 19) - 0.001489676075
Time for topKMatches("", 22) - 0.001518017075
Time for topKMatches("", 25) - 0.001515711343
Time for topKMatches("khombu", 1) - 6.89123E-7
Time for topKMatches("khombu", 4) - 6.48434E-7
Time for topKMatches("khombu", 7) - 6.62644E-7
Time for topKMatches("khombu", 10) - 5.95916E-7
Time for topKMatches("khombu", 13) - 5.98195E-7
Time for topKMatches("khombu", 16) - 6.35278E-7
Time for topKMatches("khombu", 19) - 6.1876E-7
Time for topKMatches("khombu", 22) - 6.12082E-7
Time for topKMatches("khombu", 25) - 6.36757E-7
Time for topKMatches("k", 1) - 3.5772959E-5
Time for topKMatches("k", 4) - 3.9122849E-5
Time for topKMatches("k", 7) - 3.9063156E-5
Time for topKMatches("k", 10) - 3.6689784E-5
Time for topKMatches("k", 13) - 4.3520875E-5
Time for topKMatches("k", 16) - 3.7618974E-5
Time for topKMatches("k", 19) - 4.0017867E-5
Time for topKMatches("k", 22) - 3.3819664E-5
Time for topKMatches("k", 25) - 3.7362922E-5
Time for topKMatches("kh", 1) - 2.275612E-6
Time for topKMatches("kh", 4) - 2.131858E-6
Time for topKMatches("kh", 7) - 1.889336E-6
Time for topKMatches("kh", 10) - 1.77158E-6
Time for topKMatches("kh", 13) - 1.667659E-6
Time for topKMatches("kh", 16) - 1.506715E-6
Time for topKMatches("kh", 19) - 1.478383E-6
Time for topKMatches("kh", 22) - 1.433815E-6
Time for topKMatches("kh", 25) - 1.613208E-6
Time for topKMatches("notarealword", 1) - 5.77217E-7
Time for topKMatches("notarealword", 4) - 5.81633E-7
Time for topKMatches("notarealword", 7) - 5.67544E-7
Time for topKMatches("notarealword", 10) - 5.74077E-7
Time for topKMatches("notarealword", 13) - 5.6795E-7
Time for topKMatches("notarealword", 16) - 5.68087E-7
Time for topKMatches("notarealword", 19) - 5.67516E-7
Time for topKMatches("notarealword", 22) - 6.04018E-7
Time for topKMatches("notarealword", 25) - 5.78095E-7

Analysis:

The big O for topKMatches for BinarySearchAutocomplete is log n + m log k + k log k. This is supported by the data in the table, since as k increases, the running time increases. For example, as k increases from 1 to 22, while the length of the prefix and terms stay constant, running time increases from 3.5772959E-5 to 3.3819664E-5. To explain the big O, log n comes

from the firstIndexOf and lastIndexOf methods, which use binary heap. For binary heaps, everything below the node must be bigger than it. The number of times one needs to swap numbers so that this structure is maintained is $\log_2 n = \log n$. m log k comes from the fact that there are k things in the parameter, and the program looks through every match so log k is multiplied with m. k log k comes from the fact that we take k things out and put them into an array, so it is k multiplied by log k.

2.

BruteAutocomplete:

Data Table 1

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking BruteAutocomplete...
Found 333333 words
Time to initialize - 0.008839648
Time for topMatch("") - 0.002151984512
Time for topMatch("khombu") - 0.002984595043
Time for topMatch("k") - 0.002113641746
Time for topMatch("kh") - 0.002247514707
Time for topMatch("notarealword") - 0.002029355929
Time for topKMatches("", 1) - 0.002125533251
Time for topKMatches("", 4) - 0.002095510885
Time for topKMatches("", 7) - 0.002204989049
Time for topKMatches("khombu", 1) - 0.003019464971
Time for topKMatches("khombu", 4) - 0.002878861421
Time for topKMatches("khombu", 7) - 0.002743077513
Time for topKMatches("k", 1) - 0.002019100765
Time for topKMatches("k", 4) - 0.001973991499
Time for topKMatches("k", 7) - 0.002078123178
Time for topKMatches("kh", 1) - 0.002220168597
Time for topKMatches("kh", 4) - 0.00213331255
Time for topKMatches("kh", 7) - 0.002068937328
Time for topKMatches("notarealword", 1) - 0.001808855293
Time for topKMatches("notarealword", 4) - 0.001806709769
Time for topKMatches("notarealword", 7) - 0.00183850643

Data Table 2

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking BruteAutocomplete...
Found 333333 words
Time to initialize - 0.0092131
Time for topMatch("") - 0.002400329117
Time for topMatch("khombu") - 0.003922127502

Time for topMatch("k") - 0.003135758945
Time for topMatch("kh") - 0.002333811571
Time for topMatch("notarealword") - 0.002021137053
Time for topKMatches("", 1) - 0.0021207913
Time for topKMatches("", 4) - 0.002056963629
Time for topKMatches("", 7) - 0.002065781446
Time for topKMatches("", 10) - 0.002157585976
Time for topKMatches("", 13) - 0.001998061798
Time for topKMatches("", 16) - 0.001964640199
Time for topKMatches("", 19) - 0.001950313491
Time for topKMatches("", 22) - 0.001917617909
Time for topKMatches("", 25) - 0.00197822335
Time for topKMatches("khombu", 1) - 0.002721979029
Time for topKMatches("khombu", 4) - 0.002694420335
Time for topKMatches("khombu", 7) - 0.002747091101
Time for topKMatches("khombu", 10) - 0.002730975174
Time for topKMatches("khombu", 13) - 0.002706933152
Time for topKMatches("khombu", 16) - 0.002856782447
Time for topKMatches("khombu", 19) - 0.002856842302
Time for topKMatches("khombu", 22) - 0.002680199879
Time for topKMatches("khombu", 25) - 0.002711918297
Time for topKMatches("k", 1) - 0.002008003575
Time for topKMatches("k", 4) - 0.002040152359
Time for topKMatches("k", 7) - 0.002001028621
Time for topKMatches("k", 10) - 0.002123430137
Time for topKMatches("k", 13) - 0.002357648468
Time for topKMatches("k", 16) - 0.002094141373
Time for topKMatches("k", 19) - 0.002211785437
Time for topKMatches("k", 22) - 0.00219662073
Time for topKMatches("k", 25) - 0.002105370883
Time for topKMatches("kh", 1) - 0.002167598497
Time for topKMatches("kh", 4) - 0.002161143194
Time for topKMatches("kh", 7) - 0.002106608646
Time for topKMatches("kh", 10) - 0.002145815896
Time for topKMatches("kh", 13) - 0.00216311513
Time for topKMatches("kh", 16) - 0.002145491294
Time for topKMatches("kh", 19) - 0.002121608924
Time for topKMatches("kh", 22) - 0.002157189237
Time for topKMatches("kh", 25) - 0.00223403789
Time for topKMatches("notarealword", 1) - 0.001979809343
Time for topKMatches("notarealword", 4) - 0.001879814294
Time for topKMatches("notarealword", 7) - 0.001800297932
Time for topKMatches("notarealword", 10) - 0.001890464993
Time for topKMatches("notarealword", 13) - 0.002000757823
Time for topKMatches("notarealword", 16) - 0.001966749318
Time for topKMatches("notarealword", 19) - 0.002065923802

Time for topKMatches("notarealword", 22) - 0.001934216677
Time for topKMatches("notarealword", 25) - 0.00199283336

Analysis:

The runtime of topKMatches() varies with k with the following relationship: (n-m) + m log k+ k log k. Algorithmically, this is because the program goes through all elements, generating n-m. For m log k , the program goes through every match to add it to the priority queue, so m is multiplied by log k. For k log k, program takes the things out k times to put them into a string array, resulting in k being multiplied by log k. This is supported by empirical data, as shown in Data Table 1 and 2. Given this relationship between the runtime of topKMatches () and k, as k increases runtime should also increase, which is the case in both data tables. For example, in Table 2, as k increases from 22 to 25 while the length of the prefix and number of terms stay the same, runtime increases from 0.001934216677 to 0.00199283336.

BinarySearchAutocomplete

The runtime of topKMatches() varies with k with the following relationship: log n + m log k + k log k.

This can be referenced in Question 1. The reason for the relationship has been explained algorithmically and backed up by empirical data in Question 1.

TrieAutocomplete

Data Table 1

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking TrieAutocomplete...
Found 333333 words
Time to initialize - 0.407235108
Created 805917 nodes
Time for topMatch("") - 1.732419E-6
Time for topMatch("khombu") - 8.16373E-7
Time for topMatch("k") - 1.025109E-6
Time for topMatch("kh") - 5.74224E-7
Time for topMatch("notarealword") - 7.30001E-7
Time for topKMatches("", 1) - 1.7655226E-5
Time for topKMatches("", 4) - 8.529076E-6
Time for topKMatches("", 7) - 7.610322E-6
Time for topKMatches("khombu", 1) - 4.77841E-7
Time for topKMatches("khombu", 4) - 4.40026E-7
Time for topKMatches("khombu", 7) - 4.12901E-7
Time for topKMatches("k", 1) - 5.22442E-6
Time for topKMatches("k", 4) - 4.622014E-6
Time for topKMatches("k", 7) - 4.319905E-6

Time for topKMatches("kh", 1) - 2.064116E-6
Time for topKMatches("kh", 4) - 1.964422E-6
Time for topKMatches("kh", 7) - 2.121608E-6
Time for topKMatches("notarealword", 1) - 2.88807E-7
Time for topKMatches("notarealword", 4) - 2.85031E-7
Time for topKMatches("notarealword", 7) - 3.01017E-7

Data Table 2

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/words-333333.txt.
Benchmarking TrieAutocomplete...
Found 333333 words
Time to initialize - 0.371304608
Created 805917 nodes
Time for topMatch("") - 1.775199E-6
Time for topMatch("khombu") - 8.67348E-7
Time for topMatch("k") - 1.011574E-6
Time for topMatch("kh") - 5.6975E-7
Time for topMatch("notarealword") - 7.37175E-7
Time for topKMatches("", 1) - 1.9214334E-5
Time for topKMatches("", 4) - 8.252157E-6
Time for topKMatches("", 7) - 7.904643E-6
Time for topKMatches("", 10) - 7.633179E-6
Time for topKMatches("", 13) - 8.045796E-6
Time for topKMatches("", 16) - 8.073366E-6
Time for topKMatches("", 19) - 7.964617E-6
Time for topKMatches("", 22) - 7.569862E-6
Time for topKMatches("", 25) - 7.925873E-6
Time for topKMatches("khombu", 1) - 4.75072E-7
Time for topKMatches("khombu", 4) - 4.47186E-7
Time for topKMatches("khombu", 7) - 4.15798E-7
Time for topKMatches("khombu", 10) - 4.82897E-7
Time for topKMatches("khombu", 13) - 4.29388E-7
Time for topKMatches("khombu", 16) - 4.47537E-7
Time for topKMatches("khombu", 19) - 4.3892E-7
Time for topKMatches("khombu", 22) - 4.45045E-7
Time for topKMatches("khombu", 25) - 4.06862E-7
Time for topKMatches("k", 1) - 4.962708E-6
Time for topKMatches("k", 4) - 4.628616E-6
Time for topKMatches("k", 7) - 4.216693E-6
Time for topKMatches("k", 10) - 4.098093E-6
Time for topKMatches("k", 13) - 4.263865E-6
Time for topKMatches("k", 16) - 4.073709E-6
Time for topKMatches("k", 19) - 4.341007E-6
Time for topKMatches("k", 22) - 4.15988E-6
Time for topKMatches("k", 25) - 4.172283E-6

Time for topKMatches("kh", 1) - 2.512303E-6
Time for topKMatches("kh", 4) - 2.632584E-6
Time for topKMatches("kh", 7) - 1.934316E-6
Time for topKMatches("kh", 10) - 1.970613E-6
Time for topKMatches("kh", 13) - 1.91773E-6
Time for topKMatches("kh", 16) - 1.921526E-6
Time for topKMatches("kh", 19) - 1.925222E-6
Time for topKMatches("kh", 22) - 2.051201E-6
Time for topKMatches("kh", 25) - 1.993777E-6
Time for topKMatches("notarealword", 1) - 2.77014E-7
Time for topKMatches("notarealword", 4) - 2.74396E-7
Time for topKMatches("notarealword", 7) - 2.74652E-7
Time for topKMatches("notarealword", 10) - 2.83516E-7
Time for topKMatches("notarealword", 13) - 3.12743E-7
Time for topKMatches("notarealword", 16) - 2.91207E-7
Time for topKMatches("notarealword", 19) - 2.73732E-7
Time for topKMatches("notarealword", 22) - 3.62593E-7
Time for topKMatches("notarealword", 25) - 3.38616E-7

Analysis:

The runtime of topKMatches() varies with k with the following relationship, with lower bound (best case scenario) k log k and upper bound (worst case scenario): x(logx + log k). Algorithmically the lower bound is k log k because the program runs a total of k times, resulting in k being multiplied by log k. The upper bound is x (logx + log k) because x is the number of nodes rooted at prefix, and that's why it is multiplied by (log x + log k). The empirical data from Data Table 1 and 2 doesn't always support this relationship, however. For example, in Data Table 2 when k increases from 1 to 4 while prefix and terms are set, runtime decreases from 2.77014E-7 to 2.74396E-7. This is caused by the small number of trails performed. With a larger number of trails, the empirical data will correspond more closely with the theoretical runtimes.

3.

The empirical runtimes for both BruteAutocomplete and BinarySearchAutocomplete are available in the data tables that can be referenced in Question 1 and 2. The theoretical runtimes for BruteAutocomplete and BinarySearchAutocomplete are:

BruteAutocomplete

topKMatches: (n-m) + m log k+ k log k
topMatches: n

BinarySearchAutocomplete
topKMatches: log n + m log k + k log k
topMatches: rlog n + m

Analysis:

No, BinarySearchAutocomplete is not always guaranteed to perform better than BruteAutocomplete. This is because for topKMatches(), for example, when the prefix is sufficiently short or empty, the BinarySearchAutocomplete doesn't perform better than BruteAutocomplete. In fact, in that case the BinarySearchAutocomplete would take a longer time than BruteAutocomplete. This is supported by empirical data. For the same prfix "" and k = 1, the running time for BruteAutocomplete is 0.002125533251 while that for BinarySearchComplete is 0.002794196574.

4.

BruteAutocomplete

Data Table 1 (with fourletterwords.txt)

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwords.txt.
Benchmarking BruteAutocomplete...
Found 456976 words
Time to initialize - 0.012970713
Time for topMatch("") - 0.003845889333
Time for topMatch("nenk") - 0.005927341714454976
Time for topMatch("n") - 0.005476693471553611
Time for topMatch("ne") - 0.005689018860227273
Time for topMatch("notarealword") - 0.004333404639
Time for topKMatches("", 1) - 0.005723849480549199
Time for topKMatches("", 4) - 0.006142695825766871
Time for topKMatches("", 7) - 0.005674251333333333
Time for topKMatches("", 10) - 0.0055792787670011145
Time for topKMatches("", 13) - 0.00581744038255814
Time for topKMatches("", 16) - 0.005601280352743561
Time for topKMatches("", 19) - 0.005814460111627907
Time for topKMatches("", 22) - 0.0066244210516556296
Time for topKMatches("", 25) - 0.005801683625290023
Time for topKMatches("nenk", 1) - 0.005947385550535077
Time for topKMatches("nenk", 4) - 0.005687507485227273
Time for topKMatches("nenk", 7) - 0.00556000228
Time for topKMatches("nenk", 10) - 0.006002864552220888
Time for topKMatches("nenk", 13) - 0.006181400315203956
Time for topKMatches("nenk", 16) - 0.005997111752997602
Time for topKMatches("nenk", 19) - 0.005861877048065651
Time for topKMatches("nenk", 22) - 0.005909951309327036
Time for topKMatches("nenk", 25) - 0.006040825422705314
Time for topKMatches("n", 1) - 0.00603499056212304
Time for topKMatches("n", 4) - 0.006552617835078534

Time for topKMatches("n", 7) - 0.006175253113580247
Time for topKMatches("n", 10) - 0.006313440517676768
Time for topKMatches("n", 13) - 0.00622838443159204
Time for topKMatches("n", 16) - 0.006260675707133918
Time for topKMatches("n", 19) - 0.006036590271411339
Time for topKMatches("n", 22) - 0.005836057618436406
Time for topKMatches("n", 25) - 0.005752013770114943
Time for topKMatches("ne", 1) - 0.005431755191096634
Time for topKMatches("ne", 4) - 0.005556506516666667
Time for topKMatches("ne", 7) - 0.006096715444579781
Time for topKMatches("ne", 10) - 0.006005087960384153
Time for topKMatches("ne", 13) - 0.005739175862385321
Time for topKMatches("ne", 16) - 0.006162734353448275
Time for topKMatches("ne", 19) - 0.006270059573934837
Time for topKMatches("ne", 22) - 0.005656559943438914
Time for topKMatches("ne", 25) - 0.00584435039135514
Time for topKMatches("notarealword", 1) - 0.004287809403
Time for topKMatches("notarealword", 4) - 0.004210368878
Time for topKMatches("notarealword", 7) - 0.004219188769
Time for topKMatches("notarealword", 10) - 0.004260709572
Time for topKMatches("notarealword", 13) - 0.004360452022
Time for topKMatches("notarealword", 16) - 0.004452004979
Time for topKMatches("notarealword", 19) - 0.00440096955
Time for topKMatches("notarealword", 22) - 0.004372842902
Time for topKMatches("notarealword", 25) - 0.004316975978

Data Table 2 (with fourletterwordshalf.txt)

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwordshalf.txt.
Benchmarking BruteAutocomplete...
Found 228488 words
Time to initialize - 0.009351562
Time for topMatch("") - 0.001885116594
Time for topMatch("aenk") - 0.002668525571
Time for topMatch("a") - 0.002288306914
Time for topMatch("ae") - 0.002276082397
Time for topMatch("notarealword") - 0.001797196169
Time for topKMatches("", 1) - 0.002419050223
Time for topKMatches("", 4) - 0.002104069789
Time for topKMatches("", 7) - 0.002138907984
Time for topKMatches("", 10) - 0.002246180905
Time for topKMatches("", 13) - 0.002339559279
Time for topKMatches("", 16) - 0.002194988836
Time for topKMatches("", 19) - 0.002065111663

Time for topKMatches("", 22) - 0.002190852691
Time for topKMatches("", 25) - 0.00207310421
Time for topKMatches("aenk", 1) - 0.002888659472
Time for topKMatches("aenk", 4) - 0.002943552573
Time for topKMatches("aenk", 7) - 0.003105582897
Time for topKMatches("aenk", 10) - 0.002986225462
Time for topKMatches("aenk", 13) - 0.003104484715
Time for topKMatches("aenk", 16) - 0.003434021701
Time for topKMatches("aenk", 19) - 0.003138572893
Time for topKMatches("aenk", 22) - 0.002960904515
Time for topKMatches("aenk", 25) - 0.003069039624
Time for topKMatches("a", 1) - 0.003322705383
Time for topKMatches("a", 4) - 0.003420335486
Time for topKMatches("a", 7) - 0.003285147921
Time for topKMatches("a", 10) - 0.003117127522
Time for topKMatches("a", 13) - 0.003025488043
Time for topKMatches("a", 16) - 0.003157107666
Time for topKMatches("a", 19) - 0.003115040272
Time for topKMatches("a", 22) - 0.003134881064
Time for topKMatches("a", 25) - 0.003464488404
Time for topKMatches("ae", 1) - 0.003202868192
Time for topKMatches("ae", 4) - 0.003168341196
Time for topKMatches("ae", 7) - 0.003129749415
Time for topKMatches("ae", 10) - 0.003183023494
Time for topKMatches("ae", 13) - 0.003094882791
Time for topKMatches("ae", 16) - 0.003041348861
Time for topKMatches("ae", 19) - 0.003172865715
Time for topKMatches("ae", 22) - 0.00324132288
Time for topKMatches("ae", 25) - 0.003287817513
Time for topKMatches("notarealword", 1) - 0.002866023028
Time for topKMatches("notarealword", 4) - 0.002736395973
Time for topKMatches("notarealword", 7) - 0.003287447608
Time for topKMatches("notarealword", 10) - 0.002962825122
Time for topKMatches("notarealword", 13) - 0.003095023401
Time for topKMatches("notarealword", 16) - 0.002835258214
Time for topKMatches("notarealword", 19) - 0.002822788578
Time for topKMatches("notarealword", 22) - 0.003192698439
Time for topKMatches("notarealword", 25) - 0.003032906421

Theoretical runtimes:

topKMatches: (n-m) + m log k+ k log k
topMatches: n

Analysis:

According to the theoretical runtimes, increasing the size of the source (n) should increase both runtimes for topKMatches() and topMatches() for BruteAutocomplete. This is supported by empirical data. For example, the running time for topKMathches () with prefix that has a length of 2 and k = 25 is 0.003287817513 for fourletterwordshalf.txt and is 0.00584435039135514 for fourletterwords.txt, which is twice as long.

On the other hand, increasing the size of the prefix decreases m. With fewer matches, it takes longer to do each compare and each add (which is reflected by the increase in the value of r). Therefore, increasing the size of prefix will decrease topKMatches but will not affect topMatches. This is not entirely supported by empirical data. For example, the runtime for topKMatches() with “ae” is 0.003287817513, while that with a longer prefix, “notarealword”, is 0.003032906421. However, the runtime for topMatches() changes slightly as the length of prefix changes. This can be due to the fact that there are only a small number of trails in the colllection of data.

BinarySearchAutocomplete

Data Table 1 (with fourletterwords.txt)

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwords.txt.
Benchmarking BinarySearchAutocomplete...
Found 456976 words
Time to initialize - 0.030611598
Time for topMatch("") - 5.50301262E-4
Time for topMatch("nenk") - 5.360576E-6
Time for topMatch("n") - 1.4835388E-5
Time for topMatch("ne") - 8.61168E-7
Time for topMatch("notarealword") - 4.060175E-6
Time for topKMatches("", 1) - 0.001335865278
Time for topKMatches("", 4) - 0.001366841553
Time for topKMatches("", 7) - 0.001344350458
Time for topKMatches("", 10) - 0.001455723911
Time for topKMatches("", 13) - 0.001604266778
Time for topKMatches("", 16) - 0.001424049822
Time for topKMatches("", 19) - 0.001418264772
Time for topKMatches("", 22) - 0.001363451523
Time for topKMatches("", 25) - 0.001391781769
Time for topKMatches("nenk", 1) - 7.88948E-7
Time for topKMatches("nenk", 4) - 7.63817E-7
Time for topKMatches("nenk", 7) - 7.12435E-7
Time for topKMatches("nenk", 10) - 7.29843E-7
Time for topKMatches("nenk", 13) - 6.89453E-7
Time for topKMatches("nenk", 16) - 7.05263E-7
Time for topKMatches("nenk", 19) - 6.90159E-7

Time for topKMatches("nenk", 22) - 6.89604E-7
Time for topKMatches("nenk", 25) - 7.18169E-7
Time for topKMatches("n", 1) - 5.106104E-5
Time for topKMatches("n", 4) - 5.3053128E-5
Time for topKMatches("n", 7) - 4.9235522E-5
Time for topKMatches("n", 10) - 5.5800402E-5
Time for topKMatches("n", 13) - 5.1842001E-5
Time for topKMatches("n", 16) - 5.228443E-5
Time for topKMatches("n", 19) - 5.1220961E-5
Time for topKMatches("n", 22) - 5.3150674E-5
Time for topKMatches("n", 25) - 4.900018E-5
Time for topKMatches("ne", 1) - 4.784966E-6
Time for topKMatches("ne", 4) - 3.901321E-6
Time for topKMatches("ne", 7) - 3.171046E-6
Time for topKMatches("ne", 10) - 2.95873E-6
Time for topKMatches("ne", 13) - 2.84801E-6
Time for topKMatches("ne", 16) - 2.810841E-6
Time for topKMatches("ne", 19) - 3.0129E-6
Time for topKMatches("ne", 22) - 2.892102E-6
Time for topKMatches("ne", 25) - 2.768374E-6
Time for topKMatches("notarealword", 1) - 5.80583E-7
Time for topKMatches("notarealword", 4) - 5.72046E-7
Time for topKMatches("notarealword", 7) - 5.69916E-7
Time for topKMatches("notarealword", 10) - 5.75999E-7
Time for topKMatches("notarealword", 13) - 5.70817E-7
Time for topKMatches("notarealword", 16) - 5.84424E-7
Time for topKMatches("notarealword", 19) - 6.41064E-7
Time for topKMatches("notarealword", 22) - 6.37451E-7
Time for topKMatches("notarealword", 25) - 5.86429E-7

Data Table 2 (with fourletterwordshalf.txt)

Opening -
/Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwordshalf.txt.
Benchmarking BinarySearchAutocomplete...
Found 228488 words
Time to initialize - 0.022177502
Time for topMatch("") - 2.3225987E-4
Time for topMatch("aenk") - 4.604589E-6
Time for topMatch("a") - 1.5583992E-5
Time for topMatch("ae") - 7.80039E-7
Time for topMatch("notarealword") - 3.153979E-6
Time for topKMatches("", 1) - 6.89909688E-4
Time for topKMatches("", 4) - 6.90518513E-4
Time for topKMatches("", 7) - 6.31196738E-4

Time for topKMatches("", 10) - 6.59681459E-4
Time for topKMatches("", 13) - 8.59374987E-4
Time for topKMatches("", 16) - 6.76347015E-4
Time for topKMatches("", 19) - 7.50898574E-4
Time for topKMatches("", 22) - 7.52308788E-4
Time for topKMatches("", 25) - 7.45000659E-4
Time for topKMatches("aenk", 1) - 7.42301E-7
Time for topKMatches("aenk", 4) - 7.2156E-7
Time for topKMatches("aenk", 7) - 7.03563E-7
Time for topKMatches("aenk", 10) - 6.6694E-7
Time for topKMatches("aenk", 13) - 7.40344E-7
Time for topKMatches("aenk", 16) - 6.76534E-7
Time for topKMatches("aenk", 19) - 6.54055E-7
Time for topKMatches("aenk", 22) - 6.41748E-7
Time for topKMatches("aenk", 25) - 6.39452E-7
Time for topKMatches("a", 1) - 5.4492066E-5
Time for topKMatches("a", 4) - 6.2725472E-5
Time for topKMatches("a", 7) - 5.4792121E-5
Time for topKMatches("a", 10) - 5.8177931E-5
Time for topKMatches("a", 13) - 5.5243946E-5
Time for topKMatches("a", 16) - 6.2179819E-5
Time for topKMatches("a", 19) - 5.6857604E-5
Time for topKMatches("a", 22) - 5.932686E-5
Time for topKMatches("a", 25) - 6.7649517E-5
Time for topKMatches("ae", 1) - 2.924459E-6
Time for topKMatches("ae", 4) - 2.906644E-6
Time for topKMatches("ae", 7) - 2.79212E-6
Time for topKMatches("ae", 10) - 2.959862E-6
Time for topKMatches("ae", 13) - 2.964277E-6
Time for topKMatches("ae", 16) - 2.88446E-6
Time for topKMatches("ae", 19) - 3.053637E-6
Time for topKMatches("ae", 22) - 2.812154E-6
Time for topKMatches("ae", 25) - 2.977194E-6
Time for topKMatches("notarealword", 1) - 4.45833E-7
Time for topKMatches("notarealword", 4) - 4.60322E-7
Time for topKMatches("notarealword", 7) - 4.307E-7
Time for topKMatches("notarealword", 10) - 4.36103E-7
Time for topKMatches("notarealword", 13) - 4.97074E-7
Time for topKMatches("notarealword", 16) - 4.24219E-7
Time for topKMatches("notarealword", 19) - 4.26023E-7
Time for topKMatches("notarealword", 22) - 4.84623E-7
Time for topKMatches("notarealword", 25) - 4.3774E-7

Theoretical runtimes:

topKMatches: log n + m log k + k log k
topMatches: rlog n + m

Analysis:

According to the theoretical runtimes, increasing the size of the source (n) should increase both the runtimes for topKMatches() and topMatches(), since log(n) increases as n increases. This is supported by empirical data. With prefix "notarealword" and k = 25, the runtime for topKMatches() is 4.3774E-7 for fourletterwordshalf.txt while it is 5.86429E-7 for fourletterwords.txt. Similarly, both with prefix of length 2, the runtime for topMatch() for fourletterwordshalf.txt is 7.80039E-7 while that for fourletterwords.txt is 8.61168E-7.

On the other hand, increasing the size of the prefix decreases m. With fewer matches, it takes longer to do each compare and each add (which is reflected by the increase in the value of r). Therefore, it decreases the runtimes for both topKMatches() and topMatches(). However, this is not supported by the data. This is because the trail number is low, so the data is not representative.

TrieAutocomplete

Data Table 1 (with fourletterwords.txt)

Opening - /Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwords.txt.
Benchmarking TrieAutocomplete...
Found 456976 words
Time to initialize - 0.106356024
Created 475255 nodes
Time for topMatch("") - 3.4977E-6
Time for topMatch("nenk") - 7.78251E-7
Time for topMatch("n") - 1.502256E-6
Time for topMatch("ne") - 1.215503E-6
Time for topMatch("notarealword") - 7.28703E-7
Time for topKMatches("", 1) - 2.0051775E-5
Time for topKMatches("", 4) - 8.622767E-6
Time for topKMatches("", 7) - 8.378842E-6
Time for topKMatches("", 10) - 8.160821E-6
Time for topKMatches("", 13) - 8.93297E-6
Time for topKMatches("", 16) - 8.529728E-6
Time for topKMatches("", 19) - 8.449749E-6
Time for topKMatches("", 22) - 9.951133E-6
Time for topKMatches("", 25) - 8.144114E-6
Time for topKMatches("nenk", 1) - 3.98633E-7
Time for topKMatches("nenk", 4) - 3.28402E-7
Time for topKMatches("nenk", 7) - 3.27358E-7

Time for topKMatches("nenk", 10) - 3.46417E-7
Time for topKMatches("nenk", 13) - 3.25985E-7
Time for topKMatches("nenk", 16) - 3.25217E-7
Time for topKMatches("nenk", 19) - 3.26223E-7
Time for topKMatches("nenk", 22) - 3.25991E-7
Time for topKMatches("nenk", 25) - 3.54709E-7
Time for topKMatches("n", 1) - 6.519585E-6
Time for topKMatches("n", 4) - 6.39931E-6
Time for topKMatches("n", 7) - 7.936137E-6
Time for topKMatches("n", 10) - 6.540992E-6
Time for topKMatches("n", 13) - 6.370051E-6
Time for topKMatches("n", 16) - 6.087659E-6
Time for topKMatches("n", 19) - 6.223508E-6
Time for topKMatches("n", 22) - 6.860009E-6
Time for topKMatches("n", 25) - 6.177182E-6
Time for topKMatches("ne", 1) - 2.997319E-6
Time for topKMatches("ne", 4) - 2.785426E-6
Time for topKMatches("ne", 7) - 2.783508E-6
Time for topKMatches("ne", 10) - 2.907865E-6
Time for topKMatches("ne", 13) - 2.897187E-6
Time for topKMatches("ne", 16) - 3.043958E-6
Time for topKMatches("ne", 19) - 2.90339E-6
Time for topKMatches("ne", 22) - 2.936203E-6
Time for topKMatches("ne", 25) - 2.907172E-6
Time for topKMatches("notarealword", 1) - 2.51506E-7
Time for topKMatches("notarealword", 4) - 2.21886E-7
Time for topKMatches("notarealword", 7) - 2.47563E-7
Time for topKMatches("notarealword", 10) - 2.49213E-7
Time for topKMatches("notarealword", 13) - 2.38752E-7
Time for topKMatches("notarealword", 16) - 2.45674E-7
Time for topKMatches("notarealword", 19) - 2.28069E-7
Time for topKMatches("notarealword", 22) - 2.47173E-7
Time for topKMatches("notarealword", 25) - 2.24401E-7

Data Table 2 (with fourletterwordshalf.txt)

Opening -
/Users/xiaoyuyanglian/Documents/workspace2/autocomplete/data/fourletterwordshalf.txt.
Benchmarking TrieAutocomplete...
Found 228488 words
Time to initialize - 0.083623558
Created 237628 nodes
Time for topMatch("") - 2.159518E-6
Time for topMatch("aenk") - 5.90543E-7
Time for topMatch("a") - 9.87922E-7

Time for topMatch("ae") - 1.03275E-6
Time for topMatch("notarealword") - 1.88303E-7
Time for topKMatches("", 1) - 1.7205135E-5
Time for topKMatches("", 4) - 7.455398E-6
Time for topKMatches("", 7) - 7.203903E-6
Time for topKMatches("", 10) - 8.704695E-6
Time for topKMatches("", 13) - 7.486672E-6
Time for topKMatches("", 16) - 7.165075E-6
Time for topKMatches("", 19) - 7.122363E-6
Time for topKMatches("", 22) - 7.516501E-6
Time for topKMatches("", 25) - 6.91328E-6
Time for topKMatches("aenk", 1) - 4.37483E-7
Time for topKMatches("aenk", 4) - 3.46947E-7
Time for topKMatches("aenk", 7) - 3.4522E-7
Time for topKMatches("aenk", 10) - 3.38445E-7
Time for topKMatches("aenk", 13) - 3.23095E-7
Time for topKMatches("aenk", 16) - 3.26007E-7
Time for topKMatches("aenk", 19) - 3.23201E-7
Time for topKMatches("aenk", 22) - 3.23045E-7
Time for topKMatches("aenk", 25) - 3.56569E-7
Time for topKMatches("a", 1) - 5.816919E-6
Time for topKMatches("a", 4) - 6.028857E-6
Time for topKMatches("a", 7) - 5.938475E-6
Time for topKMatches("a", 10) - 5.934933E-6
Time for topKMatches("a", 13) - 6.332221E-6
Time for topKMatches("a", 16) - 7.200633E-6
Time for topKMatches("a", 19) - 6.128827E-6
Time for topKMatches("a", 22) - 6.212356E-6
Time for topKMatches("a", 25) - 6.342166E-6
Time for topKMatches("ae", 1) - 3.738865E-6
Time for topKMatches("ae", 4) - 3.706237E-6
Time for topKMatches("ae", 7) - 3.759317E-6
Time for topKMatches("ae", 10) - 3.835758E-6
Time for topKMatches("ae", 13) - 3.674619E-6
Time for topKMatches("ae", 16) - 3.835251E-6
Time for topKMatches("ae", 19) - 3.530885E-6
Time for topKMatches("ae", 22) - 4.423616E-6
Time for topKMatches("ae", 25) - 3.642452E-6
Time for topKMatches("notarealword", 1) - 7.3178E-8
Time for topKMatches("notarealword", 4) - 7.2743E-8
Time for topKMatches("notarealword", 7) - 7.1621E-8
Time for topKMatches("notarealword", 10) - 8.2568E-8
Time for topKMatches("notarealword", 13) - 7.3813E-8
Time for topKMatches("notarealword", 16) - 7.2942E-8
Time for topKMatches("notarealword", 19) - 7.2557E-8
Time for topKMatches("notarealword", 22) - 8.8448E-8

Time for topKMatches("notarealword", 25) - 7.2708E-8

Theoretical runtimes:

topKMatches: lower bound: k log k ; upper bound: x(logx + log k)
topMatches: w

Analysis:

According to the theoretical runtimes, increasing the size of the source (n) should not affect either the runtimes of topKMatches() or that of topMatches(), since there is their runtime doesn't depend on n. This is not supported by empirical data. For example, the runtime for topKMatches() with prefix "notarealword" and k = 22 increases from 8.8448E-8 to 2.24401E-7 as n doubles. Similarly, the runtime for topMatches() with prefix of length 1 increases from 9.87922E-7 to 1.502256E-6 as n doubles. This is due to the small number of trails performed, rendering the trails unrepresentative.

On the other hand, increasing the size of the prefix decreases m. With fewer matches, it takes longer to do each compare and each add (which is reflected by the increase in the value of r). However, it doesn't affect the runtimes of either topKMatches() or topMatches() because their runtimes again doesn't depend on m. This is again not supported by empirical data. In Data Table 2, with k=1, the runtime of topKMatches() changes from 5.816919E-6 to 7.3178E-8 as the prefix changes from "a" to "notarealword". Similarly, in Data Table 2, as prefix changes from "ae" to "notarealword", the runtime for topMatch() changes from 1.03275E-6 to 1.88303E-7. This is due to the small number of trails performed, rendering the trails unrepresentative.