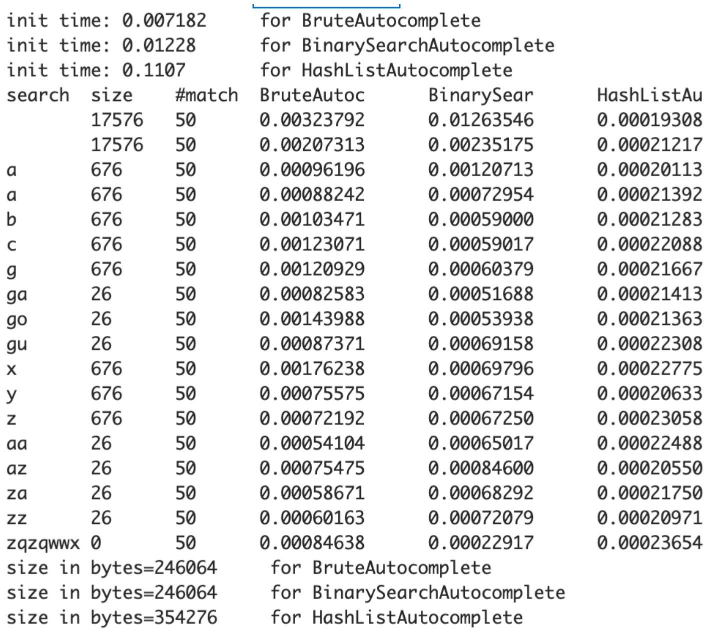
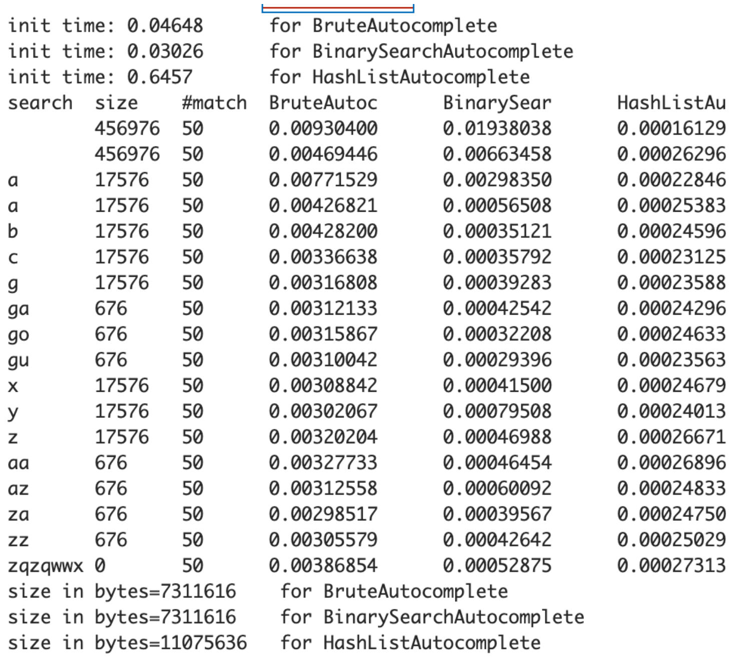
1.
Textfile: txt.threeletterwords



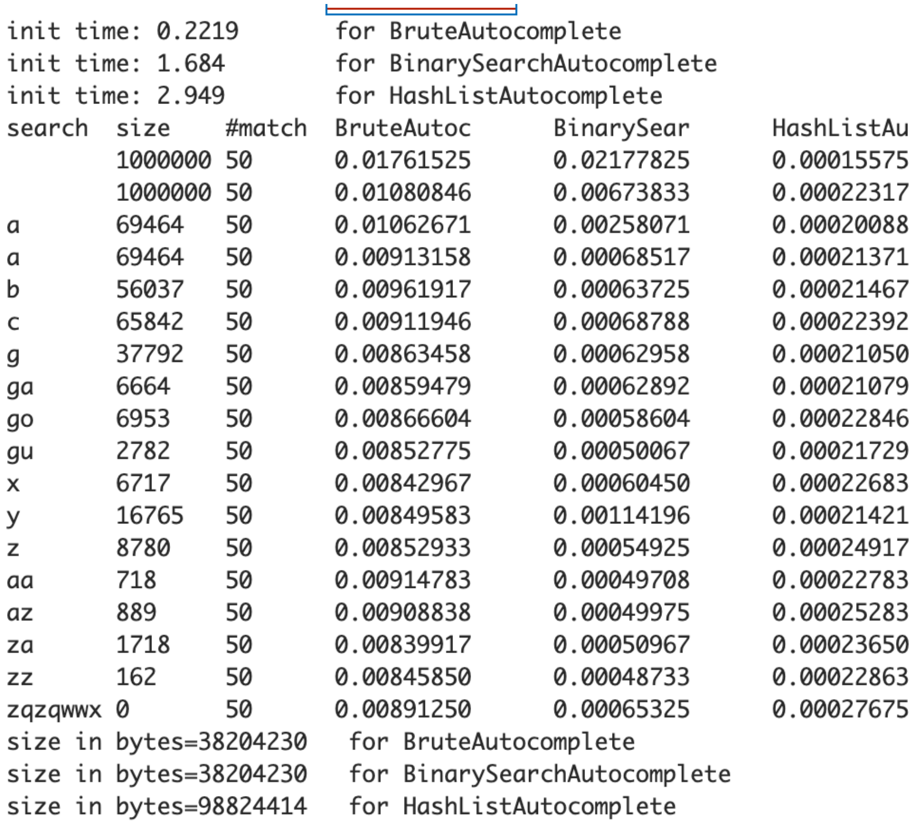
```
init time: 0.007182     for BruteAutocomplete
init time: 0.01228      for BinarySearchAutocomplete
init time: 0.1107       for HashListAutocomplete
search  size    #match  BruteAutoc      BinarySear      HashListAu
        17576   50      0.00323792      0.01263546      0.00019308
        17576   50      0.00207313      0.00235175      0.00021217
a       676     50      0.00096196      0.00120713      0.00020113
a       676     50      0.00088242      0.00072954      0.00021392
b       676     50      0.00103471      0.00059000      0.00021283
c       676     50      0.00123071      0.00059017      0.00022088
g       676     50      0.00120929      0.00060379      0.00021667
ga      26      50      0.00082583      0.00051688      0.00021413
go      26      50      0.00143988      0.00053938      0.00021363
gu      26      50      0.00087371      0.00069158      0.00022308
x       676     50      0.00176238      0.00069796      0.00022775
y       676     50      0.00075575      0.00067154      0.00020633
z       676     50      0.00072192      0.00067250      0.00023058
aa      26      50      0.00054104      0.00065017      0.00022488
az      26      50      0.00075475      0.00084600      0.00020550
za      26      50      0.00058671      0.00068292      0.00021750
zz      26      50      0.00060163      0.00072079      0.00020971
zqzqwwx 0       50      0.00084638      0.00022917      0.00023654
size in bytes=246064     for BruteAutocomplete
size in bytes=246064     for BinarySearchAutocomplete
size in bytes=354276     for HashListAutocomplete
```

Textfile: txt.fourletterwords



```
init time: 0.04648      for BruteAutocomplete
init time: 0.03026      for BinarySearchAutocomplete
init time: 0.6457       for HashListAutocomplete
search  size    #match  BruteAutoc      BinarySear      HashListAu
        456976  50      0.00930400      0.01938038      0.00016129
        456976  50      0.00469446      0.00663458      0.00026296
a       17576   50      0.00771529      0.00298350      0.00022846
a       17576   50      0.00426821      0.00056508      0.00025383
b       17576   50      0.00428200      0.00035121      0.00024596
c       17576   50      0.00336638      0.00035792      0.00023125
g       17576   50      0.00316808      0.00039283      0.00023588
ga      676     50      0.00312133      0.00042542      0.00024296
go      676     50      0.00315867      0.00032208      0.00024633
gu      676     50      0.00310042      0.00029396      0.00023563
x       17576   50      0.00308842      0.00041500      0.00024679
y       17576   50      0.00302067      0.00079508      0.00024013
z       17576   50      0.00320204      0.00046988      0.00026671
aa      676     50      0.00327733      0.00046454      0.00026896
az      676     50      0.00312558      0.00060092      0.00024833
za      676     50      0.00298517      0.00039567      0.00024750
zz      676     50      0.00305579      0.00042642      0.00025029
zqzqwwx 0       50      0.00386854      0.00052875      0.00027313
size in bytes=7311616    for BruteAutocomplete
size in bytes=7311616    for BinarySearchAutocomplete
size in bytes=11075636   for HashListAutocomplete
```

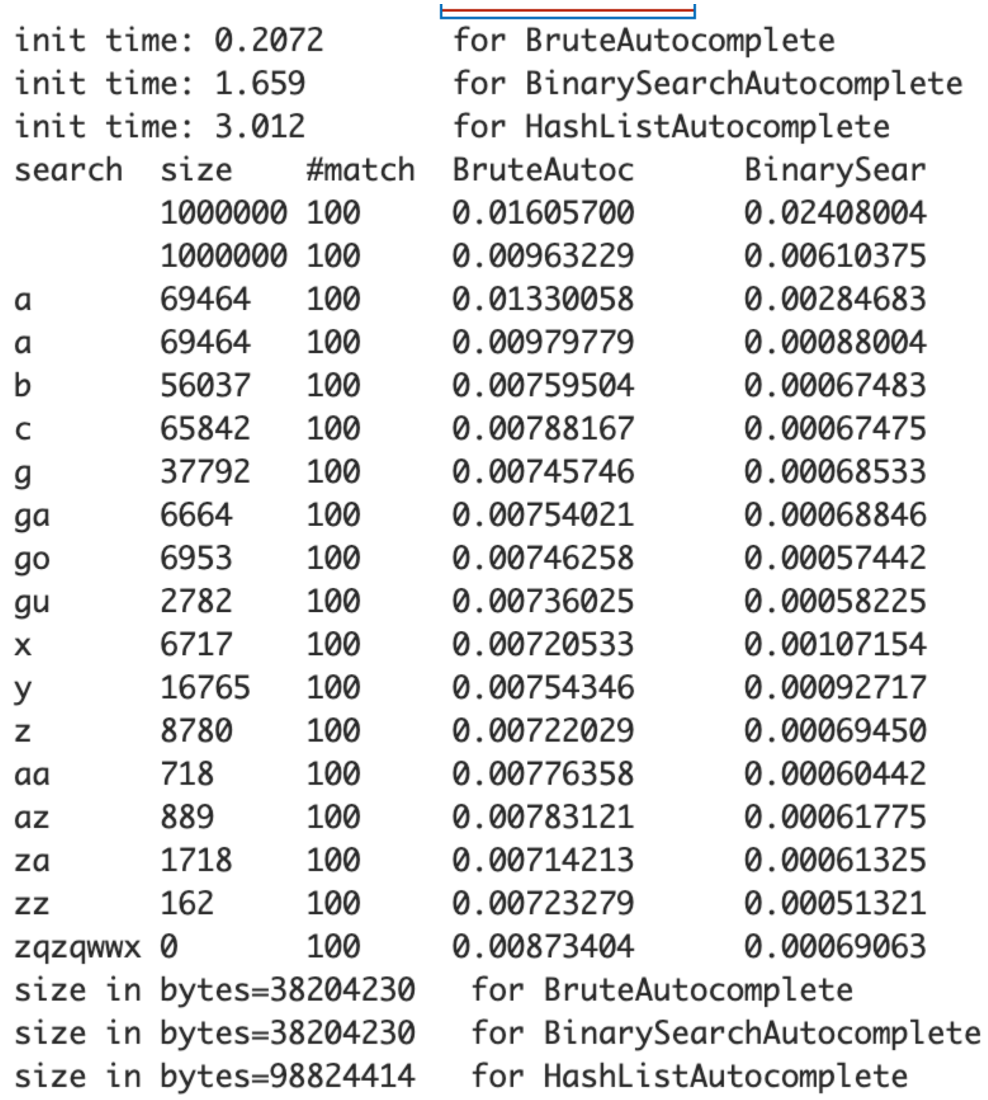
Textfile: txt.alexa



```
init time: 0.2219       for BruteAutocomplete
init time: 1.684        for BinarySearchAutocomplete
init time: 2.949        for HashListAutocomplete
search  size    #match  BruteAutoc      BinarySear      HashListAu
        1000000 50      0.01761525      0.02177825      0.00015575
        1000000 50      0.01080846      0.00673833      0.00022317
a       69464   50      0.01062671      0.00258071      0.00020088
a       69464   50      0.00913158      0.00068517      0.00021371
b       56037   50      0.00961917      0.00063725      0.00021467
c       65842   50      0.00911946      0.00068788      0.00022392
g       37792   50      0.00863458      0.00062958      0.00021050
ga      6664    50      0.00859479      0.00062892      0.00021079
go      6953    50      0.00866604      0.00058604      0.00022846
gu      2782    50      0.00852775      0.00050067      0.00021729
x       6717    50      0.00842967      0.00060450      0.00022683
y       16765   50      0.00849583      0.00114196      0.00021421
z       8780    50      0.00852933      0.00054925      0.00024917
aa      718     50      0.00914783      0.00049708      0.00022783
az      889     50      0.00908838      0.00049975      0.00025283
za      1718    50      0.00839917      0.00050967      0.00023650
zz      162     50      0.00845850      0.00048733      0.00022863
zqzqwwx 0       50      0.00891250      0.00065325      0.00027675
size in bytes=38204230   for BruteAutocomplete
size in bytes=38204230   for BinarySearchAutocomplete
size in bytes=98824414   for HashListAutocomplete
```

2.
The reason BruteAutocomplete and BinarySearchAutocomplete have similar runtimes for the String ”” is because when the prefix String is “”, it corresponds to a prefix-match with every term N. Thus N == M. The reason for this is that BinarySearch as well as BruteAutocomplete will have to search through every term no matter what. The reason BruteAutocomplete may appear slightly more efficient is based the algorithm of firstMatch. BruteAutocomplete won’t have to set int values for first and last since it automatically searches the entire list of Terms. However, BinarySearchAutocomplete’s version of firstMatch sets first and last bounds arbitrarily thus adding an extra step, slightly increasing runtime.

3.
Textfile: alexa.txt



```
init time: 0.2072       for BruteAutocomplete
init time: 1.659        for BinarySearchAutocomplete
init time: 3.012        for HashListAutocomplete
search  size    #match  BruteAutoc      BinarySear      HashListAu
        1000000 100     0.01605700      0.02408004      0.00015371
        1000000 100     0.00963229      0.00610375      0.00018808
a       69464   100     0.01330058      0.00284683      0.00017575
a       69464   100     0.00979779      0.00088004      0.00020175
b       56037   100     0.00759504      0.00067483      0.00019121
c       65842   100     0.00788167      0.00067475      0.00020500
g       37792   100     0.00745746      0.00068533      0.00019871
ga      6664    100     0.00754021      0.00068846      0.00027792
go      6953    100     0.00746258      0.00057442      0.00027917
gu      2782    100     0.00736025      0.00058225      0.00027288
x       6717    100     0.00720533      0.00107154      0.00019321
y       16765   100     0.00754346      0.00092717      0.00019138
z       8780    100     0.00722029      0.00069450      0.00022858
aa      718     100     0.00776358      0.00060442      0.00029050
az      889     100     0.00783121      0.00061775      0.00020658
za      1718    100     0.00714213      0.00061325      0.00029871
zz      162     100     0.00723279      0.00051321      0.00029471
zqzqwwx 0       100     0.00873404      0.00069063      0.00031879
size in bytes=38204230   for BruteAutocomplete
size in bytes=38204230   for BinarySearchAutocomplete
size in bytes=98824414   for HashListAutocomplete
```

Yes. The doubling of k doesn't correspond to a doubling of runtime. Instead it represents to and increase of log(k) which is uniform with our runtime hypothesis.

4.
The biggest difference between the HashAutocomplete firstMatch method and the other autocomplete classes is the implementation of a Priority Queue vs. a HashMap. BinarySearchAutocomplete (BSA) and BruteAutocomplete (BA) use an implementation that requires the firstMatch method to iterate through the list of terms each time the method is called and feeding the terms through a PriorityQueue. While BSA uses a binary search algorithm to speed up this process, the HashAutocomplete firstMatch method has a constant runtime as all the terms are copied into a hash map and accessed in O(1). This means HashAutocomplete only iterates through the list of terms once (not in firstMatch). The one drawback to this implementation is the HashMap must read the entire list of terms before the firstMatch method is called, which uses more memory than BSA and BA as they only read the terms with the matching prefixes.