sortingAlgorithms

January 25, 2017

In [1]:
```
import os
os.chdir('/Users/ken/Desktop/Grad School/Classes/Biocomputing Algorithms/ho
from sortingComplexity import algs, run
import numpy as np
import sys
%matplotlib inline
```

In [2]:
```
#testing quicksort
A = np.round(np.random.rand(100)*1000)
print('Unsorted')
print(A)
print('')
print('Conditionals, Assignments and Time')
print(algs.quicksort(A))
print('')
print('Sorted')
print(A)
```

```
Unsorted
[ 453.  541.  342.  623.  495.  710.  726.  775.  228.  773.   81.  733.
  222.  767.  627.   70.  479.  855.  366.  275.  682.   85.  419.  395.
  462.  670.  528.  991.  776.  512.  512.  354.  440.  148.  628.  119.
  676.  246.  361.  529.  928.  926.  198.  582.  376.  335.  225.  238.
  754.  819.  219.  784.  173.  537.  886.  779.  509.  154.  231.  700.
  945.  123.  295.   89.  208.  603.   67.  531.  459.  972.  296.  311.
  975.  144.  244.  944.  554.   50.  412.  256.  947.  981.  257.  279.
  952.  434.  646.  277.  740.  628.  201.  902.  463.  583.  752.  450.
  626.  688.  990.  414.]

Conditionals, Assignments and Time
(1108, 1894, 0.004269838333129883)

Sorted
[  50.   67.   70.   81.   85.   89.  119.  123.  144.  148.  154.  173.
  198.  201.  208.  219.  222.  225.  228.  231.  238.  244.  246.  256.
  257.  275.  277.  279.  295.  296.  311.  335.  342.  354.  361.  366.
  376.  395.  412.  414.  419.  434.  440.  450.  453.  459.  462.  463.
```

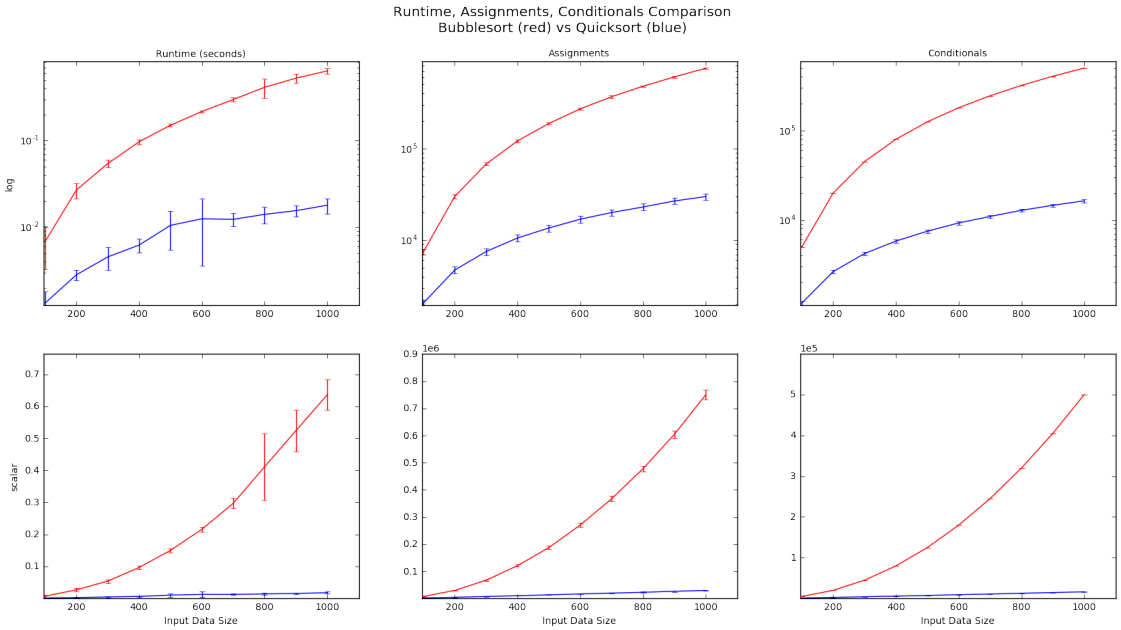
```
  479.  495.  509.  512.  512.  528.  529.  531.  537.  541.  554.  582.
  583.  603.  623.  626.  627.  628.  628.  646.  670.  676.  682.  688.
  700.  710.  726.  733.  740.  752.  754.  767.  773.  775.  776.  779.
  784.  819.  855.  886.  902.  926.  928.  944.  945.  947.  952.  972.
  975.  981.  990.  991.]
```

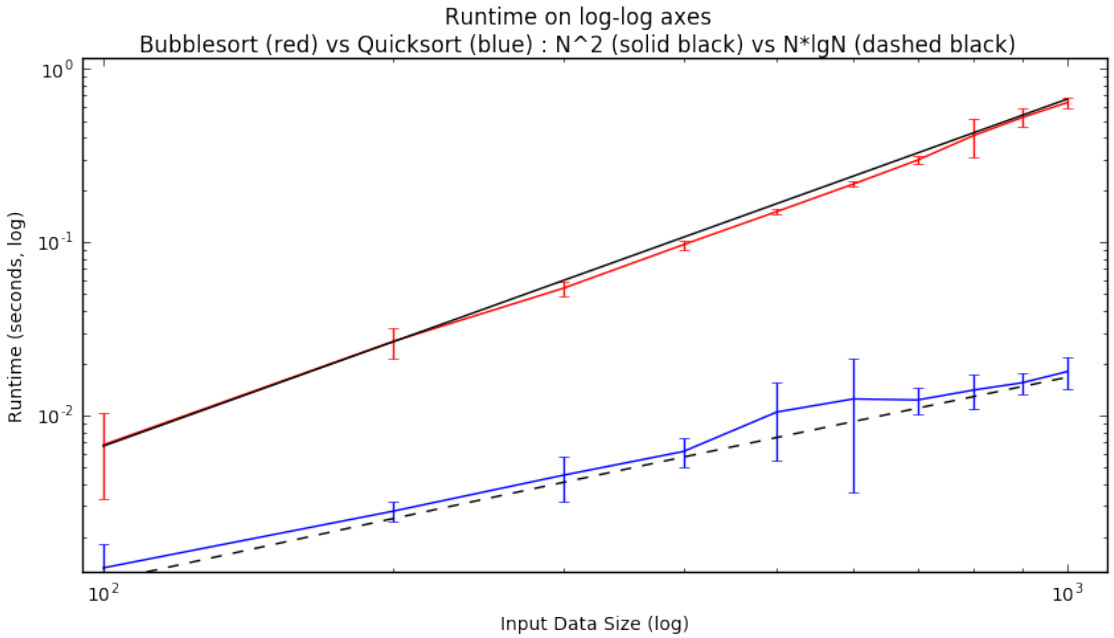
```
In [3]: #testing bubblesort
        A = np.round(np.random.rand(100)*1000)
        print('Unsorted')
        print(A)
        print('')
        print('Conditionals, Assignments, and Time')
        print(algs.bubblesort(A))
        print('')
        print('Sorted')
        print(A)
```

```
Unsorted
[ 763.  504.  757.  714.  886.  464.  114.  780.  652.  248.  228.  956.
  802.  509.  846.  700.  377.   30.  705.  663.   91.  989.  235.  720.
  893.  809.  582.  532.  873.  433.  688.  773.  238.  775.  750.  894.
  667.  886.   95.  529.  121.  648.  355.  868.  586.  216.  676.  208.
  760.  277.  746.  401.  884.  742.   95.  502.  619.  681.  739.  200.
    1.  643.  882.  841.  681.  136.  823.  817.  584.  207.  209.  516.
   54.  964.  926.  638.  326.  111.  555.  610.  163.  383.  417.  301.
  997.  582.  665.  766.  970.  755.  854.  246.  862.  585.    8.  408.
  801.  490.  409.  213.]

Conditionals, Assignments, and Time
(4951, 7950, 0.012)

Sorted
[   1.    8.   30.   54.   91.   95.   95.  111.  114.  121.  136.  163.
  200.  207.  208.  209.  213.  216.  228.  235.  238.  246.  248.  277.
  301.  326.  355.  377.  383.  401.  408.  409.  417.  433.  464.  490.
  502.  504.  509.  516.  529.  532.  555.  582.  582.  584.  585.  586.
  610.  619.  638.  643.  648.  652.  663.  665.  667.  676.  681.  681.
  688.  700.  705.  714.  720.  739.  742.  746.  750.  755.  757.  760.
  763.  766.  773.  775.  780.  801.  802.  809.  817.  823.  841.  846.
  854.  862.  868.  873.  882.  884.  886.  886.  893.  894.  926.  956.
  964.  970.  989.  997.]
```

```
In [5]: #visualize complexity as a function of
        #runtime, assignments and conditionals.
        #also compares to N^2 and N*lgN expectations

        run.complexity_test()
```

```
------------------------------------------------

        Testing Bubblesort Complexity:
('\tinputDataSizes = ', [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000])
('\trepsPerInputSize = ', 100, '\n')
[##################################################] 100%
------------------------------------------------

        Testing Quicksort Complexity:
('\tinputDataSizes = ', [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000])
('\trepsPerInputSize = ', 100, '\n')
[##################################################] 100%
------------------------------------------------
```




Runtime on log-log axes
Bubblesort (red) vs Quicksort (blue) : N^2 (solid black) vs N*lgN (dashed black)
Runtime (seconds, log)
Input Data Size (log)
10^0
10^{-1}
10^{-2}
10^2
10^3