GRPAH:

In the graph above, it is clear to see that the Quicksort algorithm is the fastest of the three. The slowest is the Insertion, which means that Selection is in the middle of the two. At the beginning of the graph, until at least one hundred elements in a list, all three of the algorithms can be completed in zero seconds. Only Quicksort had a difference in time of zero for the one thousand sized element.

While in the long run Quicksort is much faster than Insertion, that is only for small lists. Quicksort is not efficient on small lists because the function call is recursive, and therefore uses more memory. While this trade off may seem worth it in the long run, with short lists Insertion is actually a better function call. This is also true with Selection and Quicksort. With large lists Quicksort is much faster, but with smaller ones it may not be as worth it.

# Python Code

import time

import random

import sys

sys.setrecursionlimit(1000000)

# Insertion

def insertionSort(alist):

for index in range(1,len(alist)):

currentvalue = alist[index]

position = index

while position>0 and alist[position-1]>currentvalue:

alist[position]=alist[position-1]

position = position-1

alist[position]=currentvalue

# Quicksort

def quickSort(alist):

quickSortHelper(alist,0,len(alist)-1)

def quickSortHelper(alist,first,last):

if first<last:

splitpoint = partition(alist,first,last)

quickSortHelper(alist,first,splitpoint-1)

quickSortHelper(alist,splitpoint+1,last)

def partition(alist,first,last):

pivotvalue = alist[first]

leftmark = first+1

rightmark = last

done = False

while not done:

while leftmark <= rightmark and alist[leftmark] <= pivotvalue:

leftmark = leftmark + 1

while alist[rightmark] >= pivotvalue and rightmark >= leftmark:

rightmark = rightmark -1

if rightmark < leftmark:

done = True

else:

temp = alist[leftmark]

alist[leftmark] = alist[rightmark]

alist[rightmark] = temp

temp = alist[first]

alist[first] = alist[rightmark]

alist[rightmark] = temp

return rightmark

# Selection

def selectionSort(alist):

for fillslot in range(len(alist)-1,0,-1):

positionOfMax=0

for location in range(1,fillslot+1):

if alist[location]>alist[positionOfMax]:

positionOfMax = location

temp = alist[fillslot]

alist[fillslot] = alist[positionOfMax]

alist[positionOfMax] = temp

blist = random.sample(xrange(11), 10)

clist = random.sample(xrange(101), 100)

dlist = random.sample(xrange(1001), 1000)

elist = random.sample(xrange(10001), 10000)

flist = random.sample(xrange(100001), 100000)

print "insertionSort(blist)"

print time.time()

insertionSort(blist)

print time.time()

print "quickSort(blist)"

print time.time()

quickSort(blist)

print time.time()

print "selectionSort(blist)"

print time.time()

selectionSort(blist)

print time.time()

print "insertionSort(clist)"

print time.time()

insertionSort(clist)

print time.time()

print "quickSort(clist)"

print time.time()

quickSort(clist)

print time.time()

print "selectionSort(clist)"

print time.time()

selectionSort(clist)

print time.time()

print "insertionSort(dlist)"

print time.time()

insertionSort(dlist)

print time.time()

print "quickSort(dlist)"

print time.time()

quickSort(dlist)

print time.time()

print "selectionSort(dlist)"

print time.time()

selectionSort(dlist)

print time.time()

print "insertionSort(elist)"

print time.time()

insertionSort(elist)

print time.time()

print "quickSort(elist)"

print time.time()

quickSort(elist)

print time.time()

print "selectionSort(elist)"

print time.time()

selectionSort(elist)

print time.time()

print "insertionSort(flist)"

print time.time()

insertionSort(flist)

print time.time()

print "quickSort(flist)"

print time.time()

quickSort(flist)

print time.time()

print "selectionSort(flist)"

print time.time()

selectionSort(flist)

print time.time()

listInOrder = range(0,5000, 1)

listBackwards = range(5000,0,-1)

listInRandom = random.sample(xrange(5000), 5000)

print "in order:"

print time.time()

insertionSort(listInOrder)

print time.time()

quickSort(listInOrder)

print time.time()

selectionSort(listInOrder)

print time.time()

print "backwards"

print time.time()

insertionSort(listBackwards)

print time.time()

quickSort(listBackwards)

print time.time()

selectionSort(listBackwards)

print time.time()

print "random"

print time.time()

insertionSort(listInRandom)

print time.time()

quickSort(listInRandom)

print time.time()

selectionSort(listInRandom)

print time.time()