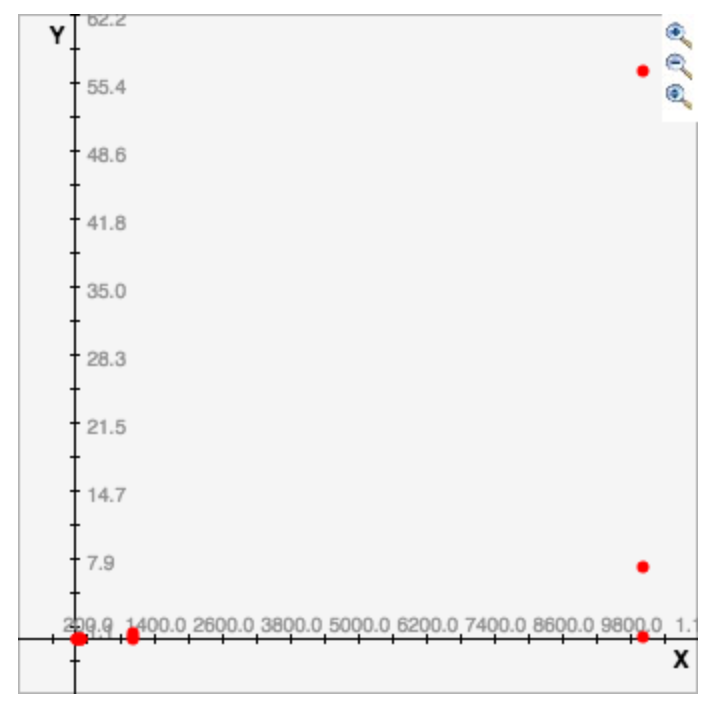
Jennie Walker

CS 1122

First of all, it is important to note that as each of the programs (except for the one with an input of 10 digits) have been set to randomize the integers chosen to be sorted. All randomly chosen integers are picked anywhere from 0 to 1234567 in order to increase the chance there will be at least a six digit number in the list (as requested) without completely overloading the program. The runtimes have been computed before the program printed so as to not overly skew the runtime data for the larger inputs. For each program, I made alternate versions that did not print out the list (after checking to make sure each algorithm worked correctly) so I didn’t end up with a million integers on my screen or have the program crash.



|  |
| --- |
| The runtimes of 10, 100, 1000, and 10000 inputs |

For 10 inputs, the runtime of insertion program was 1.6927719116210938e-05 seconds. The runtime of the quicksort program was 1.9073486328125e-05 seconds. The runtime of the selection program was 2.193450927734375e-05 seconds.

For 100 inputs, the runtime of insertion program was 0.0004868507385253906 seconds. The runtime of the quicksort program was 0.00047087669372558594 seconds. The runtime of the selection program was 0.0055789947509765625 seconds.

For 1000 inputs, the runtime of insertion program was 0.06512117385864258 seconds. The runtime of the quicksort program was 0.009360074996948242 seconds. The runtime of the selection program was 0.5181300640106201 seconds.

For 10000 inputs, the runtime of insertion program was 7.136783123016357 seconds. The runtime of the quicksort program was 0.13089799880981445 seconds. The runtime of the selection program was 56.513365030288696 seconds.

For 100000 inputs, the runtime of insertion program was 932.1603579521179 seconds. The runtime of the quicksort program was 0.5074360370635986 seconds. The runtime of the selection program was 682.9022371768951 seconds.

I chose not to test the sort time for 1,000,000 inputs because as the multiplier for the Insertion program alone from each program to the next averaged at about 130, I learned it would take approximately 38 hours to calculate the next runtime (provided the trend continued) and I simply do not have the time. I tried to test it but after an hour, there was still no result.

The insertion program tended to be quite reliable. It was never the quickest program, that honor went to the quicksort program, but it did increase its run time pretty consistently. While the first jump increased the run time by only seven times the original, the tests after revealed an increase in runtime by a factor from around 110 to 140. It was very quick, like the other programs, when sorting 10 inputs, but once it reached 10,000 inputs, it went from running under a second to running over seven seconds.

The quicksort program was much more reliable for a shorter sorting time, even though the program itself had more text in it. It tended to decrease the multiplier for the run time with each interval. When going from 10 inputs to 100 inputs, it increased by a factor of about 24.7. From 100 inputs to 1000 inputs, it increased by a factor of 19.9. From 1000 inputs to 10000 inputs, it increased by an even lesser factor of 14, and from 10000 inputs to 100000 inputs, the time was only multiplied by a factor of 3.9, allowing the final run time to still be under one second.

The selection program was easily the least efficient, at least in the beginning. The biggest surprise was that it had a lesser run time than the insertion program for 100000 inputs. The increase of it’s run time was also inconsistent, with each run time being multiplied by 254, 93, 109 and 12, respectively, as more and more inputs were added.

In general, the program with the best run time, the quicksort program, also had the most while loops with a total of three. The program that most often had the second best run time, the insertion program, also had a while loop, though there was only one. The selection program had none. I am not sure why the selection program was faster than the insertion program when sorting through 100000 inputs.

My Code:

import time

#insertion w set 10 inputs

print("Insertion Program:")

t0 = time.time()

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

alist = [54,26,93,17,77,31,44,55,20,100000]

insertionSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print(runTime, "seconds.")

#quicksort w set 10 inputs

print("Quicksort Program:")

t0 = time.time()

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

alist = [54,26,93,17,77,31,44,55,20,100000]

quickSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print(runTime, "seconds.")

#selection with 10 set inputs

print("Selection Program:")

t0 = time.time()

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

alist = [54,26,93,17,77,31,44,55,20,100000]

selectionSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print(runTime, "seconds.")

import time

import random

#insertion w 100 rand inputs

print("Insertion Program:")

#sets up list

alist = []

for listAdd in range(0, 100):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

insertionSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#quicksort w 100 rand inputs

print("\n\nQuicksort Program:")

for listAdd in range(0, 100):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

quickSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#selection with 100 rand inputs

print("\n\nSelection Program:")

for listAdd in range(0, 100):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

selectionSort(alist)

print(alist)

t1 = time.time()

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds")

import time

import random

#insertion w 1000 rand inputs

print("Insertion Program:")

#sets up list

alist = []

for listAdd in range(0, 1000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

insertionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#quicksort w 1000 rand inputs

print("\n\nQuicksort Program:")

for listAdd in range(0, 1000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

quickSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#selection with 1000 rand inputs

print("\n\nSelection Program:")

for listAdd in range(0, 1000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

selectionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

import time

import random

#insertion w 10000 rand inputs

print("Insertion Program:")

#sets up list

alist = []

for listAdd in range(0, 10000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

insertionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#quicksort w 10000 rand inputs

print("\n\nQuicksort Program:")

for listAdd in range(0, 10000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

quickSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#selection with 10000 rand inputs

print("\n\nSelection Program:")

for listAdd in range(0, 10000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

selectionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

import time

import random

#insertion w 100000 rand inputs

print("Insertion Program:")

#sets up list

alist = []

for listAdd in range(0, 100000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

insertionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#quicksort w 100000 rand inputs

print("\n\nQuicksort Program:")

#sets up list

alist = []

for listAdd in range(0, 100000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

quickSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")

#selection with 100000 rand inputs

print("\n\nSelection Program:")

#sets up list

alist = []

for listAdd in range(0, 100000):

alist.append(random.randint(0,1234567))

t0 = time.time()

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

selectionSort(alist)

t1 = time.time()

print(alist)

runTime = t1 - t0

print("\nRun Time:")

print(runTime, "seconds.")