Imagen que contiene computer, tabla, computadora, agua

Descripción generada automáticamenteLogotipo

Descripción generada automáticamente UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Algoritmia

Actividad 03: Analisis de tiempos y operaciones de diferentes algoritmos de ordenación

Sección: D12

Clave: I5884

Profesor: Luis Fernando Rodriguez Acosta

Alumno: José Eduardo Martínez Ulloa

Código: 217552546

A continuación el código utilizado para obtener los resultados:

#Selection Sort Reading From InputFiles Folder

#Tracks Execution Time, Assignations and Comparisons

import datetime

import time

arr=[]

inc=[5,10,100,1000,10000,100000]

start\_time = time.time()

restantes = ["Orden","Reverso"]

InpFileR = "Ordenamientos/InputFiles/Random{incid}\_{id}.txt"

InpFileOtros = "Ordenamientos/InputFiles/{nombre}{incid}.txt"

OutFile = "Ordenamientos/Stats/InsertionSortStats.txt"

def insertionSort(arr, nombreArchivo):

comp=0

asign=0

n = len(arr) # Get the length of the array

if n <= 1: return # If the array has 0 or 1 element, it is already sorted, so return

for i in range(1, n): # Iterate over the array starting from the second element

key = arr[i] # Store the current element as the key to be inserted in the right position

j = i-1

comp+=1

while j >= 0 and key < arr[j]: # Move elements greater than key one position ahead

asign+=1

arr[j+1] = arr[j] # Shift elements to the right

j -= 1

asign+=1

arr[j+1] = key

with open(OutFile, "a") as sFile:

print(nombreArchivo, file=sFile)

print("Tiempo de Ejecucion\t\tComparaciones\tAsignaciones", file=sFile)

print("%s\t\t\t%s\t%s\n" % (time.time()-start\_time,comp,asign), file=sFile)

def release\_list(a):

del a[:]

del a

def salida(nombreArchivo):

inp = open (nombreArchivo,"r")

for line in inp.readlines(): #Read line into array

for i in line.split(): #loop over the elemets, split by whitespace

arr.append(int(i)) #convert to integer and append to the list

inp.close()

print(arr)

start\_time = time.time()

insertionSort(arr, nombreArchivo)

print(arr)

release\_list(arr)

for c in range(0,6):

for a in range(1,11):

nombreArchivo = InpFileR.format(incid=inc[c],id = a)

salida(nombreArchivo)

for m, n in enumerate(restantes):

nombreArchivo = InpFileOtros.format(nombre=n,incid=inc[c])

salida(nombreArchivo)

# Quick Sort Reading form InputFiles folder

# Tracks Execution Time, Assignations and Comparisons

import datetime

import time

start\_time = time.time()

arr = []

inc = [5, 10, 100, 1000, 10000, 100000]

comp = 0

asign = 0

restantes = ["Orden","Reverso"]

InpFileR = "Ordenamientos/InputFiles/Random{incid}\_{id}.txt"

InpFileOtros = "Ordenamientos/InputFiles/{nombre}{incid}.txt"

OutFile = "Ordenamientos/Stats/QuickSortStats.txt"

def release\_list(a):

del a[:]

del a

def partition(array, low, high):

pivot = array[high]

i = low - 1

for j in range(low, high):

comp+=1

if array[j] <= pivot:

i = i + 1

asign+=2

(array[i], array[j]) = (array[j], array[i])

asign+=2

(array[i + 1], array[high]) = (array[high], array[i + 1])

return i + 1

def quickSort(array, low, high, nombreArchivo):

if low < high:

pi = partition(array, low, high)

quickSort(array, low, pi - 1, nombreArchivo)

quickSort(array, pi + 1, high, nombreArchivo)

def salida(nombreArchivo):

inp = open (nombreArchivo,"r")

for line in inp.readlines(): #Read line into array

for i in line.split(): #loop over the elemets, split by whitespace

arr.append(int(i)) #convert to integer and append to the list

inp.close()

start\_time = time.time()

print(arr)

quickSort(arr, 0, len(arr)-1, nombreArchivo)

with open(OutFile, "a") as sFile:

print(nombreArchivo, file=sFile)

print("Tiempo de Ejecucion\t\tComparaciones\tAsignaciones", file=sFile)

print("%s\t\t\t%s\t%s\n" % (time.time()-start\_time,comp,asign), file=sFile)

print(arr)

for c in range(0,6):

for a in range(1,11):

nombreArchivo = InpFileR.format(incid=inc[c],id = a)

salida(nombreArchivo)

for m, n in enumerate(restantes):

nombreArchivo = InpFileOtros.format(nombre=n,incid=inc[c])

salida(nombreArchivo)

# Bubble Sort Reading from InputFiles folder

# Tracks Execution Time, Asignations and Comparisons

import datetime

import time

start\_time = time.time()

arr = []

inc = [5, 10, 100, 1000, 10000, 100000]

restantes = ["Orden","Reverso"]

InpFileR = "Ordenamientos/InputFiles/Random{incid}\_{id}.txt"

InpFileOtros = "Ordenamientos/InputFiles/{nombre}{incid}.txt"

OutFile = "Ordenamientos/Stats/BubbleSortStats.txt"

def release\_list(a):

del a[:]

del a

def bubbleSort(arr, nombreArchivo):

n = len(arr)

comp = 0

asign = 0

swapped = False

for i in range(n-1):

for j in range(0, n-i-1):

if arr[j] > arr[j + 1]:

swapped = True

asign+=2

arr[j], arr[j + 1] = arr[j + 1], arr[j]

comp+=1

if not swapped: break

with open(OutFile, "a") as sFile:

print(nombreArchivo, file=sFile)

print("Tiempo de Ejecucion\t\tComparaciones\tAsignaciones", file=sFile)

print("%s\t\t\t%s\t%s\n" % (time.time()-start\_time,comp,asign), file=sFile)

def salida(nombreArchivo):

inp = open (nombreArchivo,"r")

for line in inp.readlines(): #Read line into array

for i in line.split(): #loop over the elemets, split by whitespace

arr.append(int(i)) #convert to integer and append to the list

inp.close()

print(arr)

start\_time = time.time()

bubbleSort(arr, nombreArchivo)

print(arr)

release\_list(arr)

for c in range(0,6):

for a in range(1,11):

nombreArchivo = InpFileR.format(incid=inc[c],id = a)

salida(nombreArchivo)

for m, n in enumerate(restantes):

nombreArchivo = InpFileOtros.format(nombre=n,incid=inc[c])

salida(nombreArchivo)

#Selection Sort Reading From InputFiles Folder

# Tracks Execution Time, Asignations and Comparisons

import datetime

import time

start\_time = time.time()

arr = []

inc = [5, 10, 100, 1000, 10000, 100000]

restantes = ["Orden","Reverso"]

InpFileR = "Ordenamientos/InputFiles/Random{incid}\_{id}.txt"

InpFileOtros = "Ordenamientos/InputFiles/{nombre}{incid}.txt"

OutFile = "Ordenamientos/Stats/SelectionSortStats.txt"

def release\_list(a):

del a[:]

del a

def selectionSort(array, nombreArchivo):

comp = 0

asign = 0

size = len(array)

for ind in range(size):

min\_index = ind

for j in range(ind + 1, size):

comp+=1

if array[j] < array[min\_index]: # select the minimum element in every iteration

min\_index = j

asign+=2

(array[ind], array[min\_index]) = (array[min\_index], array[ind]) # swapping the elements to sort the array

with open(OutFile, "a") as sFile:

print(nombreArchivo, file=sFile)

print("Tiempo de Ejecucion\t\tComparaciones\tAsignaciones", file=sFile)

print("%s\t\t\t%s\t%s\n" % (time.time()-start\_time,comp,asign), file=sFile)

def salida(nombreArchivo):

inp = open (nombreArchivo,"r")

for line in inp.readlines(): #Read line into array

for i in line.split(): #loop over the elemets, split by whitespace

arr.append(int(i)) #convert to integer and append to the list

inp.close()

print(arr)

start\_time = time.time()

selectionSort(arr, nombreArchivo)

print(arr)

release\_list(arr)

for c in range(0,6):

for a in range(1,11):

nombreArchivo = InpFileR.format(incid=inc[c],id = a)

salida(nombreArchivo)

for m, n in enumerate(restantes):

nombreArchivo = InpFileOtros.format(nombre=n,incid=inc[c])

salida(nombreArchivo)

# Shell Sort Reading from InputFiles folder

# Tracks Execution Time, Assignations and Comparisons

import datetime

import time

import math

start\_time = time.time()

arr = []

inc = [5, 10, 100, 1000, 10000, 100000]

restantes = ["Orden","Reverso"]

InpFileR = "Ordenamientos/InputFiles/Random{incid}\_{id}.txt"

InpFileOtros = "Ordenamientos/InputFiles/{nombre}{incid}.txt"

OutFile = "Ordenamientos/Stats/ShellSortStats.txt"

def release\_list(a):

del a[:]

del a

def shellSort(arr, nombreArchivo):

asign = 0

comp =0

n = len(arr)

interval = n//2

while interval > 0:

for i in range(int(interval),n):

temp = arr[i]

j = i

comp+=1

while j >= interval and arr[j-int(interval)] > temp:

asign+=1

arr[j] = arr[j-interval]

j -= interval

asign+=1

arr[j] = temp

interval //= 2

with open(OutFile, "a") as sFile:

print(nombreArchivo, file=sFile)

print("Tiempo de Ejecucion\t\tComparaciones\tAsignaciones", file=sFile)

print("%s\t\t\t%s\t%s\n" % (time.time()-start\_time,comp,asign), file=sFile)

def salida(nombreArchivo):

inp = open (nombreArchivo,"r")

for line in inp.readlines(): #Read line into array

for i in line.split(): #loop over the elemets, split by whitespace

arr.append(int(i)) #convert to integer and append to the list

inp.close()

print(arr)

start\_time = time.time()

shellSort(arr, nombreArchivo)

print(arr)

release\_list(arr)

for c in range(0,6):

for a in range(1,11):

nombreArchivo = InpFileR.format(incid=inc[c],id = a)

salida(nombreArchivo)

for m, n in enumerate(restantes):

nombreArchivo = InpFileOtros.format(nombre=n,incid=inc[c])

salida(nombreArchivo)

Estos fueron los resultados obtenidos en tiempos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Elementos | InsertionSort | SelectionSort | ShellSort | BubbleSort | QuickSort |
| Ordenados | 5 | 0.0024750233 | 0.0021641254 | 0.0025827885 | 0.0020422935 | 0.0009987354 |
|  | 10 | 0.0059008598 | 0.0033528805 | 0.0038747787 | 0.0032792091 | 0.0009982586 |
|  | 100 | 0.0189969540 | 0.0112919807 | 0.0063178539 | 0.0082061291 | 0.0105125904 |
|  | 1000 | 0.1848278046 | 0.2519450188 | 0.0348238945 | 0.4037072659 | N/A |
|  | 10000 | 16.8493330479 | 22.4619400501 | 0.4847469330 | 42.9657993317 | N/A |
|  | 100000 | 1801.1705029011 | 2412.1171951294 | 7.4664309025 | 6135.5116090775 | N/A |
| Reversos | 5 | 0.0025680065 | 0.0022511482 | 0.0026679039 | 0.0021722317 | 0.0009984970 |
|  | 10 | 0.0060019493 | 0.0036318302 | 0.0039689541 | 0.0033810139 | 0.0010082722 |
|  | 100 | 0.0194277763 | 0.0117809772 | 0.0065417290 | 0.0087783337 | 0.0105130672 |
|  | 1000 | 0.2148649693 | 0.2730488777 | 0.0366268158 | 0.4528744221 | N/A |
|  | 10000 | 20.1398389339 | 24.5327901840 | 0.5133917332 | 48.6859872341 | N/A |
|  | 100000 | 2140.3744688034 | 2633.9879031181 | 7.8872547150 | 6734.0701181889 | N/A |
| Aleatorios - 1 | 5 | 0.0005519390 | 0.0004041195 | 0.0005538464 | 0.0004270077 | 0.0000000000 |
|  | 10 | 0.0027410984 | 0.0024111271 | 0.0028507710 | 0.0023512840 | 0.0010008812 |
|  | 100 | 0.0062489510 | 0.0036318302 | 0.0041649342 | 0.0038242340 | 0.0019972324 |
|  | 1000 | 0.0370790958 | 0.0330359936 | 0.0088047981 | 0.0485551357 | 0.0103690624 |
|  | 10000 | 1.8459458351 | 2.2679388523 | 0.0710468292 | 4.6841080189 | N/A |
|  | 100000 | 190.4950070381 | 239.9030458927 | 1.1014957428 | 538.0857143402 | N/A |
| Aleatorios - 2 | 5 | 0.0008111000 | 0.0007188320 | 0.0008080006 | 0.0006902218 | 0.0000000000 |
|  | 10 | 0.0028469563 | 0.0025029182 | 0.0029349327 | 0.0024383068 | 0.0009989738 |
|  | 100 | 0.0079259872 | 0.0039129257 | 0.0043618679 | 0.0042891502 | 0.0020027161 |
|  | 1000 | 0.0541930199 | 0.0544321537 | 0.0113248825 | 0.0881421566 | 0.0185763836 |
|  | 10000 | 3.4835667610 | 4.2833960056 | 0.1158089638 | 8.8779871464 | N/A |
|  | 100000 | 367.6241228580 | 456.4596838951 | 1.7576439381 | 1807.3981029987 | N/A |
| Aleatorios - 3 | 5 | 0.0010409355 | 0.0013079643 | 0.0010769367 | 0.0012919903 | 0.0010051727 |
|  | 10 | 0.0029449463 | 0.0025999546 | 0.0030469894 | 0.0025362968 | 0.0009994507 |
|  | 100 | 0.0120871067 | 0.0042021275 | 0.0045478344 | 0.0047512054 | 0.0019972324 |
|  | 1000 | 0.0706608295 | 0.0844500065 | 0.0144257545 | 0.1275250912 | 0.0205464363 |
|  | 10000 | 5.1533398628 | 6.2986781597 | 0.1552939415 | 13.0801970959 | N/A |
|  | 100000 | 548.3641219139 | 673.5910449028 | 2.4096410275 | 2255.5225751400 | N/A |
| Aleatorios - 4 | 5 | 0.0016467571 | 0.0014228821 | 0.0018439293 | 0.0014002323 | 0.0009989738 |
|  | 10 | 0.0030338764 | 0.0026881695 | 0.0031549931 | 0.0026280880 | 0.0011034012 |
|  | 100 | 0.0143837929 | 0.0044810772 | 0.0047459602 | 0.0052161217 | 0.0029990673 |
|  | 1000 | 0.0867960453 | 0.1087088585 | 0.0168447495 | 0.1662521362 | 0.0245516300 |
|  | 10000 | 6.8544080257 | 8.3212518692 | 0.1994879246 | 17.3483912945 | N/A |
|  | 100000 | 727.6802558899 | 890.1885910034 | 3.0682067871 | 2711.2126991749 | N/A |
| Aleatorios - 5 | 5 | 0.0017638206 | 0.0015220642 | 0.0019509792 | 0.0014882088 | 0.0009999275 |
|  | 10 | 0.0031239986 | 0.0027740002 | 0.0032429695 | 0.0027141571 | 0.0010008812 |
|  | 100 | 0.0153810978 | 0.0048949718 | 0.0049369335 | 0.0057032108 | 0.0030009747 |
|  | 1000 | 0.1030418873 | 0.1291849613 | 0.0196690559 | 0.2052271366 | 0.0318048000 |
|  | 10000 | 8.5052537918 | 10.3335838318 | 0.2399387360 | 21.6037352085 | N/A |
|  | 100000 | 905.7978377342 | 1109.5786199570 | 3.7523558140 | 3835.8251750469 | N/A |
| Aleatorios - 6 | 5 | 0.0018739700 | 0.0016138554 | 0.0020408630 | 0.0015702248 | 0.0009977818 |
|  | 10 | 0.0032169819 | 0.0028629303 | 0.0033359528 | 0.0028181076 | 0.0010011196 |
|  | 100 | 0.0160839558 | 0.0052020550 | 0.0053029060 | 0.0061862469 | 0.0045068264 |
|  | 1000 | 0.1195280552 | 0.1496219635 | 0.0218999386 | 0.2446162701 | 0.0465850830 |
|  | 10000 | 10.1657478809 | 12.3531458378 | 0.2972810268 | 25.8958091736 | N/A |
|  | 100000 | 1088.4382619858 | 1326.8958559036 | 4.4084680080 | 4289.7284932137 | N/A |
| Aleatorios - 7 | 5 | 0.0020189285 | 0.0017008781 | 0.0021247864 | 0.0016512871 | 0.0010013580 |
|  | 10 | 0.0050518513 | 0.0029451847 | 0.0034229755 | 0.0029222965 | 0.0010015965 |
|  | 100 | 0.0165879726 | 0.0056710243 | 0.0055029392 | 0.0066671371 | 0.0045104027 |
|  | 1000 | 0.1357109547 | 0.1700859070 | 0.0254631042 | 0.2854321003 | 0.0415613651 |
|  | 10000 | 11.8420600891 | 14.3774621487 | 0.3425018787 | 30.1236152649 | N/A |
|  | 100000 | 1269.0760760307 | 1545.7855329514 | 5.0842916965 | 4742.7583212852 | N/A |
| Aleatorios - 8 | 5 | 0.0021269321 | 0.0017869473 | 0.0022089481 | 0.0017251968 | 0.0000000000 |
|  | 10 | 0.0054428577 | 0.0030260086 | 0.0035109520 | 0.0030112267 | 0.0012919903 |
|  | 100 | 0.0171508789 | 0.0063838959 | 0.0056998730 | 0.0071301460 | 0.0075151920 |
|  | 1000 | 0.1515431404 | 0.1905491352 | 0.0278327465 | 0.3241882324 | 0.0579791069 |
|  | 10000 | 13.5040011406 | 16.3990578651 | 0.3831439018 | 34.3395142555 | N/A |
|  | 100000 | 1450.5114350319 | 1763.2731080055 | 5.7354266644 | 5201.7384982109 | N/A |
| Aleatorios - 9 | 5 | 0.0022170544 | 0.0018918514 | 0.0023078918 | 0.0018002987 | 0.0010001659 |
|  | 10 | 0.0056838989 | 0.0031077862 | 0.0035939217 | 0.0031013489 | 0.0010035038 |
|  | 100 | 0.0181529522 | 0.0069608688 | 0.0059008598 | 0.0076181889 | 0.0065038204 |
|  | 1000 | 0.1674129963 | 0.2111260891 | 0.0305879116 | 0.3633742332 | 0.0590603352 |
|  | 10000 | 15.1519918442 | 18.4383440018 | 0.4220738411 | 38.6300811768 | N/A |
|  | 100000 | 1626.5666518211 | 1981.8458099365 | 6.3819210529 | 5660.0758502483 | N/A |
| Aleatorios - 10 | 5 | 0.0023009777 | 0.0019967556 | 0.0024089813 | 0.0018732548 | 0.0010001659 |
|  | 10 | 0.0059008598 | 0.0031881332 | 0.0036809444 | 0.0031940937 | 0.0010001659 |
|  | 100 | 0.0184457302 | 0.0109119415 | 0.0061199665 | 0.0080811977 | 0.0075204372 |
|  | 1000 | 0.1840341091 | 0.2316329479 | 0.0328898430 | 0.4029111862 | 0.0721039772 |
|  | 10000 | 16.8382239342 | 20.4551491737 | 0.4631597996 | 42.9545190334 | N/A |
|  | 100000 | 1800.8754630089 | 2200.5149488449 | 7.1000628471 | 6135.2939350605 | N/A |
| Promedios Aleatorios | 5 | 0.0016352415 | 0.0014366150 | 0.0017325163 | 0.0013917923 | 0.0007003546 |
|  | 10 | 0.0039987326 | 0.0028106213 | 0.0032775402 | 0.0027715206 | 0.0010401964 |
|  | 100 | 0.0142448425 | 0.0056252718 | 0.0051284075 | 0.0059466839 | 0.0042553902 |
|  | 1000 | 0.1110000134 | 0.1362828016 | 0.0209742785 | 0.2256223679 | 0.0383138180 |
|  | 10000 | 9.3344539165 | 11.3528007746 | 0.2689736843 | 23.7537957668 | N/A |
|  | 100000 | 997.5429233313 | 1218.8036241293 | 4.0799513578 | 3717.7639364719 | N/A |

Estos fueron los resultados obtenidos en comparaciones:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Version |  | Elementos | InsertionSort | SelectionSort | ShellSort | BubbleSort | QuickSort |
| Ordenados |  | 5 | 4 | 10 | 7 | 4 | 1972 |
|  |  | 10 | 9 | 45 | 22 | 9 | 17600 |
|  |  | 100 | 99 | 4950 | 503 | 99 | 228983 |
|  |  | 1000 | 999 | 499500 | 8006 | 999 | N/A |
|  |  | 10000 | 9999 | 49995000 | 120005 | 9999 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 99999 | N/A |
| Reversos |  | 5 | 4 | 10 | 7 | 10 | 2491 |
|  |  | 10 | 9 | 45 | 22 | 45 | 20326 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 305885 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | N/A |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 1 |  | 5 | 4 | 10 | 7 | 10 | 6 |
|  |  | 10 | 9 | 45 | 22 | 45 | 2949 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 23580 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 362507 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 2 |  | 5 | 4 | 10 | 7 | 10 | 37 |
|  |  | 10 | 9 | 45 | 22 | 45 | 3852 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 28028 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 452244 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 3 |  | 5 | 4 | 10 | 7 | 10 | 98 |
|  |  | 10 | 9 | 45 | 22 | 45 | 4669 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 34441 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 563427 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 4 |  | 5 | 4 | 10 | 7 | 10 | 178 |
|  |  | 10 | 9 | 45 | 22 | 45 | 5923 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 43124 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 711181 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 5 |  | 5 | 4 | 10 | 7 | 10 | 275 |
|  |  | 10 | 9 | 45 | 22 | 45 | 7074 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 53405 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 903470 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 6 |  | 5 | 4 | 10 | 7 | 10 | 384 |
|  |  | 10 | 9 | 45 | 22 | 45 | 8122 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 67828 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 1125625 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 7 |  | 5 | 4 | 10 | 7 | 10 | 623 |
|  |  | 10 | 9 | 45 | 22 | 45 | 9459 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 81519 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 1385569 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 8 |  | 5 | 4 | 10 | 7 | 10 | 925 |
|  |  | 10 | 9 | 45 | 22 | 45 | 10877 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 101720 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 1694491 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 9 |  | 5 | 4 | 10 | 7 | 10 | 1272 |
|  |  | 10 | 9 | 45 | 22 | 45 | 14164 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 134258 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 2108143 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Aleatorios - 10 |  | 5 | 4 | 10 | 7 | 10 | 1587 |
|  |  | 10 | 9 | 45 | 22 | 45 | 15587 |
|  |  | 100 | 99 | 4950 | 503 | 4950 | 158207 |
|  |  | 1000 | 999 | 499500 | 8006 | 499500 | 2589655 |
|  |  | 10000 | 9999 | 49995000 | 120005 | 49995000 | N/A |
|  |  | 100000 | 99999 | 4999950000 | 1500006 | 4999950000 | N/A |
| Promedios Aleatorios |  | 5 | 4.0000000000 | 10.0000000000 | 7.0000000000 | 10.0000000000 | 538.5000000000 |
|  |  | 10 | 9.0000000000 | 45.0000000000 | 22.0000000000 | 45.0000000000 | 8267.6000000000 |
|  |  | 100 | 99.0000000000 | 4950.0000000000 | 503.0000000000 | 4950.0000000000 | 72611.0000000000 |
|  |  | 1000 | 999.0000000000 | 499500.0000000000 | 8006.0000000000 | 499500.0000000000 | 1189631.2000000000 |
|  |  | 10000 | 9999.0000000000 | 49995000.0000000000 | 120005.0000000000 | 49995000.0000000000 | N/A |
|  |  | 100000 | 99999.0000000000 | 4999950000.0000000000 | 1500006.0000000000 | 4999950000.0000000000 | N/A |

Estos fueron los resultados obtenidos en asignaciones:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Elementos | InsertionSort | SelectionSort | ShellSort | BubbleSort | QuickSort |
| Ordenados | 5 | 4 | 10 | 7 | 0 | 1542 |
|  | 10 | 9 | 20 | 22 | 0 | 14590 |
|  | 100 | 99 | 200 | 503 | 0 | 303788 |
|  | 1000 | 999 | 2000 | 8006 | 0 | N/A |
|  | 10000 | 9999 | 20000 | 120005 | 0 | N/A |
|  | 100000 | 99999 | 200000 | 1500006 | 0 | N/A |
| Reversos | 5 | 14 | 10 | 11 | 20 | 1844 |
|  | 10 | 54 | 20 | 35 | 90 | 16298 |
|  | 100 | 5049 | 200 | 763 | 9900 | 419726 |
|  | 1000 | 500499 | 2000 | 12706 | 999000 | N/A |
|  | 10000 | 50004999 | 20000 | 182565 | 99990000 | N/A |
|  | 100000 | 5000049999 | 200000 | 2344566 | 9999900000 | N/A |
| Aleatorios - 1 | 5 | 8 | 10 | 11 | 8 | 8 |
|  | 10 | 37 | 20 | 32 | 56 | 2418 |
|  | 100 | 2325 | 200 | 901 | 4452 | 19818 |
|  | 1000 | 249898 | 2000 | 15552 | 497798 | 503852 |
|  | 10000 | 24854314 | 20000 | 270061 | 49688630 | N/A |
|  | 100000 | 2500477966 | 200000 | 4326224 | 5000755934 | N/A |
| Aleatorios - 2 | 5 | 6 | 10 | 9 | 4 | 52 |
|  | 10 | 39 | 20 | 38 | 60 | 3316 |
|  | 100 | 2600 | 200 | 892 | 5002 | 25790 |
|  | 1000 | 254260 | 2000 | 16520 | 506522 | 628792 |
|  | 10000 | 24599342 | 20000 | 265440 | 49178686 | N/A |
|  | 100000 | 2495606211 | 200000 | 4393974 | 4991012424 | N/A |
| Aleatorios - 3 | 5 | 13 | 10 | 12 | 18 | 88 |
|  | 10 | 39 | 20 | 33 | 54 | 3912 |
|  | 100 | 2553 | 200 | 821 | 4908 | 34344 |
|  | 1000 | 251646 | 2000 | 15900 | 501294 | 779782 |
|  | 10000 | 24890077 | 20000 | 272949 | 49760156 | N/A |
|  | 100000 | 2495949809 | 200000 | 4317894 | 4991699620 | N/A |
| Aleatorios - 4 | 5 | 10 | 10 | 11 | 12 | 140 |
|  | 10 | 27 | 20 | 30 | 36 | 5134 |
|  | 100 | 2587 | 200 | 937 | 4976 | 44996 |
|  | 1000 | 245047 | 2000 | 17032 | 488096 | 946622 |
|  | 10000 | 25392171 | 20000 | 264494 | 50764344 | N/A |
|  | 100000 | 2500158363 | 200000 | 4371890 | 5000116728 | N/A |
| Aleatorios - 5 | 5 | 10 | 10 | 11 | 12 | 228 |
|  | 10 | 40 | 20 | 39 | 62 | 6584 |
|  | 100 | 2717 | 200 | 853 | 5236 | 56392 |
|  | 1000 | 248498 | 2000 | 15251 | 494998 | 1199648 |
|  | 10000 | 24604773 | 20000 | 277310 | 49189548 | N/A |
|  | 100000 | 2499458756 | 200000 | 4384771 | 4998717514 | N/A |
| Aleatorios - 6 | 5 | 10 | 10 | 11 | 12 | 392 |
|  | 10 | 40 | 20 | 33 | 38 | 7546 |
|  | 100 | 2668 | 200 | 1018 | 5138 | 75036 |
|  | 1000 | 253690 | 2000 | 15496 | 505382 | 1364602 |
|  | 10000 | 24966526 | 20000 | 275728 | 49913054 | N/A |
|  | 100000 | 2494708551 | 200000 | 4411250 | 4989217104 | N/A |
| Aleatorios - 7 | 5 | 7 | 10 | 10 | 6 | 542 |
|  | 10 | 23 | 20 | 30 | 28 | 8520 |
|  | 100 | 2373 | 200 | 857 | 4548 | 91776 |
|  | 1000 | 250094 | 2000 | 15517 | 498190 | 1645090 |
|  | 10000 | 25075798 | 20000 | 278447 | 50131598 | N/A |
|  | 100000 | 2505168479 | 200000 | 4455811 | 5010136960 | N/A |
| Aleatorios - 8 | 5 | 8 | 10 | 9 | 8 | 794 |
|  | 10 | 41 | 20 | 40 | 64 | 10132 |
|  | 100 | 2385 | 200 | 827 | 4572 | 110570 |
|  | 1000 | 244086 | 2000 | 15347 | 486174 | 1965240 |
|  | 10000 | 25057444 | 20000 | 277421 | 50094890 | N/A |
|  | 100000 | 2498329015 | 200000 | 4407370 | 4996458032 | N/A |
| Aleatorios - 9 | 5 | 9 | 10 | 10 | 10 | 976 |
|  | 10 | 31 | 20 | 34 | 44 | 11272 |
|  | 100 | 2839 | 200 | 907 | 5480 | 143158 |
|  | 1000 | 246322 | 2000 | 15225 | 490646 | 2418338 |
|  | 10000 | 24993012 | 20000 | 262478 | 49966026 | N/A |
|  | 100000 | 2493869952 | 200000 | 4292185 | 4987539906 | N/A |
| Aleatorios - 10 | 5 | 9 | 10 | 12 | 10 | 1234 |
|  | 10 | 32 | 20 | 33 | 46 | 12680 |
|  | 100 | 2516 | 200 | 960 | 4834 | 174172 |
|  | 1000 | 258142 | 2000 | 15893 | 514286 | 2985692 |
|  | 10000 | 25076146 | 20000 | 264502 | 50132294 | N/A |
|  | 100000 | 2509778408 | 200000 | 4313954 | 5019356818 | N/A |
| Promedios Aleatorios | 5 | 9.0000000000 | 10.0000000000 | 10.6000000000 | 10.0000000000 | 445.4000000000 |
|  | 10 | 34.9000000000 | 20.0000000000 | 34.2000000000 | 48.8000000000 | 7151.4000000000 |
|  | 100 | 2556.3000000000 | 200.0000000000 | 897.3000000000 | 4914.6000000000 | 77605.2000000000 |
|  | 1000 | 250168.3000000000 | 2000.0000000000 | 15773.3000000000 | 498338.6000000000 | 1443765.8000000000 |
|  | 10000 | 24950960.3000000000 | 20000.0000000000 | 270883.0000000000 | 49881922.6000000000 | N/A |
|  | 100000 | 2499350551.0000000000 | 200000.0000000000 | 4367532.3000000000 | 4998501104.0000000000 | N/A |

Graficas de tiempo:

Graficas de asignaciones:

Graficas de comparaciones:

Observaciones:

Al realizar las primeras pruebas con un solo algoritmo, el ejercicio parecía algo redundante, ya que me parecía que Quicksort ganaría frente a los demás algoritmos siempre debido a su gran ventaja, el uso de llamadas recursivas. Esto como se puede ver en todas las graficas con ordenes de magnitud de 100000 elementos, resultó ser érroneo, pues el lenguaje de programación utilizado (Python, por su facilidad de uso y mayor lentitud, haciéndolo mejor como prueba estándar frente a lenguajes mas optimizados como C) limita el número de llamadas recursivas por protección de uso de memoria. Esto en mi opinión es una prueba clara de que hacer suposiciones respecto al comportamiento de nuestros algoritmos es erróneo y siempre vale la pena cuestionarlos incluso aquello que ya creemos dominado.

Podemos observar por otro lado como el algoritmo de Shell Sort muestra gran ventaja respecto a los demás algoritmos de forma general, recordando claro que la eficiencia de operaciones y de tiempo pueden ser intercambiables dependiendo de nuestra aplicación deseada. En este caso, se observa que mientras mayor es el número de elementos, crece mucho menos que los demás ordenamientos.

Algo que identificar, es que uno de los más inefectivos a gran escala definitivamente es el ordenamiento de burbuja, con un terrible desempeño en los ordenamientos desde 1000 en adelante, esto por la forma en la que utiliza las asignaciones y comparaciones claro está.

Si esto se tratara de una competencia, los resultados serían los siguientes (Considerando las tres métricas como parte de un todo):

1. Shell Sort
2. Selection Sort
3. Insertion Sort
4. Bubble Sort
5. Quick Sort

Esto, debido a la limitación presentada en el entorno de trabajo con el lenguaje de programación (Python).

Conclusiones:

Este ejercicio fue divertido y a la vez muy estresante, ocupé mucho tiempo resolviendo errores pequeños en la forma de medir el trabajo, haciendo que tuviera que repetir el mismo trabajo varias veces, esto me limitaba el uso del ordenador mientras trabajaba para mantener un equilibrio de recursos y no terminar dando ventaja a algún algoritmo al que de algún modo se le presentara mas memoria o menor uso del CPU en el equipo en el momento de su ejecución.

Me pareció muy interesante encontrar como la escala resulta ser un gran factor, ya que en los trabajos escolares comunes el alcance no suele llegar más allá de los 100 elementos, en muchos casos las diferencias realmente importantes se mostraban más allá de los 1000 elementos así que fue un gran avance respecto a ese aspecto.

Finalmente, espero que mi trabajo me apoye a comprender más a fondo el como comparar varias formas de resolver un problema ayuda a encontrar las debilidades entre ellas y mostrarme cuando aprovechar cada tipo de solución que hay.