Richard Hayes Crowley

06/26/2021

CSC\_242\_Lab\_05

**Algorithms.py**

*from* random *import* shuffle, sample

*from* tabulate *import* tabulate

*import* time

*from* english\_words *import* english\_words\_lower\_alpha\_set

*# a bit expensive to generate a list from such a large set, but using random sampler on sets throws a deprecation warning, and this won't make much difference to a user anyway*

english\_word\_list = list(english\_words\_lower\_alpha\_set)

class Algorithims(object):

def \_\_init\_\_(*self*, *lyst*=[]) -> None:

super().\_\_init\_\_()

*self*.lyst = *lyst*

*self*.iterationTable = []

*self*.startTime = ""

*self*.timeElapsed = ""

def printList(*self*):

print(*self*.lyst)

def generateNumericList(*self*, *size*=10):

*self*.lyst = list(range(1, *size*+1))

shuffle(*self*.lyst)

print(f"List generated...\n")

def generateWordList(*self*, *size*=10):

print(*size*)

wordList = sample(english\_word\_list, *size*)

*self*.lyst = wordList

print(f"List generated... \n")

def \_\_generateTable(*self*, *headers*, *table*):

*self*.iterationTable = tabulate(

*table*, *headers*, *tablefmt*='github', *numalign*="left")

def \_\_startClock(*self*):

*self*.startTime = time.time()

def \_\_stopClock(*self*):

*self*.timeElapsed = time.time() - *self*.startTime

*self*.startTime = None

def \_\_swap(*self*, *i*, *j*):

''' exchanges items at positions i and j '''

*# you could say lyst[i], lyst[j] = lyst[j], lyst[i] but this code shows what going*

temp = *self*.lyst[*i*]

*self*.lyst[*i*] = *self*.lyst[*j*]

*self*.lyst[*j*] = temp

def linearSearch(*self*, *target*):

''' mimic python's 'in' operator return position of target item if found, else return -1'''

*self*.\_\_startClock()

position = 0

*while* position < len(*self*.lyst):

*if* *target* == *self*.lyst[position]:

*self*.\_\_stopClock()

print(

f"Linear Search complete, {*target*} found in position {position}. Time elapsed: {*self*.timeElapsed:0.6f}")

*return* 1

position += 1

*self*.\_\_stopClock()

print(

f"Linear Search complete, {*target*} not found.\nTime elapsed: {*self*.timeElapsed}")

*return* -1

def binarySearch(*self*, *target*):

'''note! This assumes that the list is already sorted!!! '''

print("Running binary search, sorting list using quickSort and running search...")

*self*.quicksort()

sortedLyst = *self*.lyst

*self*.\_\_startClock()

iterationTable = []

left = 0

right = len(sortedLyst) - 1

*while* left <= right:

midpoint = (left + right) // 2

iterationTable.append([left, right, midpoint])

*if* *target* == sortedLyst[midpoint]:

*self*.\_\_stopClock()

print("Finished! \nBinary search iteration table\n")

*self*.\_\_generateTable(*headers*=[

"Left", "Right", "Midpoint"], *table*=iterationTable)

print(*self*.iterationTable)

print(

f"Index of {*target*} is {midpoint}. Number of iterations: {len(iterationTable)}. Time elapsed: {*self*.timeElapsed:0.6f}")

*return* 1

*elif* *target* < sortedLyst[midpoint]:

right = midpoint - 1

*else*:

left = midpoint + 1

*self*.\_\_stopClock()

*self*.\_\_generateTable(*headers*=[

"Left", "Right", "Midpoint"], *table*=iterationTable)

print(*self*.iterationTable)

print(

f"Finished!\n{*target*} not found. Time elapsed: {*self*.timeElapsed:0.6f}")

*return* -1

def bubbleSort(*self*):

*self*.\_\_startClock()

n = len(*self*.lyst)

*while* n > 1:

i = 1

*while* i < n:

*if* *self*.lyst[i] < *self*.lyst[i-1]:

*self*.\_\_swap(i, i-1)

i += 1

n -= 1

*self*.\_\_stopClock()

print(f"Bubble sort complete, time elapsed: {*self*.timeElapsed:0.6f}")

def selectionSort(*self*):

*self*.\_\_startClock()

lyst = *self*.lyst

i = 0

*# do n - 1 searches*

*while* i < len(lyst)-1:

*# for the smallest item*

minIndex = i

j = i + 1

*while* j < len(lyst):

*if* lyst[j] < lyst[minIndex]:

minIndex = j

j += 1

*if* minIndex != i:

*self*.\_\_swap(minIndex, i)

i += 1

*self*.\_\_stopClock()

print(

f"Selection sort complete, time elapsed: {*self*.timeElapsed:0.6f}")

def insertionSort(*self*):

*self*.\_\_startClock()

lyst = *self*.lyst

i = 1

*while* i < len(lyst):

itemToInsert = lyst[i]

j = i - 1

*while* j >= 0:

*if* itemToInsert < lyst[j]:

lyst[j+1] = lyst[j]

j -= 1

*else*:

*break*

lyst[j+1] = itemToInsert

i += 1

*self*.\_\_stopClock()

print(

f"Insertion sort complete, time elapsed: {*self*.timeElapsed:0.6f}")

def quicksort(*self*):

lyst = *self*.lyst

*self*.\_\_startClock()

*self*.quicksortHelper(lyst, 0, len(lyst)-1)

*self*.\_\_stopClock()

print(f"Quick sort complete, time elapsed: {*self*.timeElapsed:0.6f}")

def quicksortHelper(*self*, *lyst*, *left*, *right*):

*if* *left* < *right*:

pivotLocation = *self*.partition(*lyst*, *left*, *right*)

*self*.quicksortHelper(*lyst*, *left*, pivotLocation - 1)

*self*.quicksortHelper(*lyst*, pivotLocation + 1, *right*)

def partition(*self*, *lyst*, *left*, *right*):

*# find the pivot and exchange it with the last item*

middle = (*left* + *right*) // 2

pivot = *lyst*[middle]

*lyst*[middle] = *lyst*[*right*]

*lyst*[*right*] = pivot

*# set boundary point to first position*

boundary = *left*

*# move items less than pivot to the left*

*for* index *in* range(*left*, *right*):

*if* *lyst*[index] < pivot:

*self*.\_\_swap(index, boundary)

boundary += 1

*# exchange pivot item and boundary item*

*self*.\_\_swap(*right*, boundary)

*return* boundary

**searchAndSort.py**

*from* simple\_term\_menu *import* TerminalMenu

*from* Algorithms *import* \*

listType = ""

def main():

global listType

algos = ""

print("Hello! Welcome to 'Search and Sort', a profiling search and sort tool. Would you like to use your own list or generate an unsorted list?")

list\_choice = TerminalMenu(["Random list", "My own list", "Exit"]).show()

*if* list\_choice == 0:

algos = Algorithims()

list\_type = TerminalMenu(["Numeric list", "Word list"],

*title*="What kind of list would you like to generate?").show()

listType = "numeric" *if* list\_type == 0 *else* "word"

size = input(

"What size list? Please enter a positive integer: ")

*try*:

size = int(size)

*except* ValueError:

print("That's not an integer! Play nice or leave!")

*return* exit()

*if* list\_type == 0:

algos.generateNumericList(size)

*else*:

algos.generateWordList(size)

viewList = input(

"Would you like to view your unsorted list? Enter 'y' if you would: ")

*if* viewList == "y":

algos.printList()

*elif* list\_choice == 1:

listInput = input(

"Please enter a comma separated list OF THE SAME TYPE OF VALUES (not numbers and words, please): ").split(",")

algos = Algorithims(listInput)

*else*:

print("goodbye!")

exit()

*while* True:

sort\_or\_search = TerminalMenu(

["Sort list", "Search list", "Print current list", "Generate new list", "Exit"], *title*="\nWhat would you like to do?").show()

*if* sort\_or\_search == 0:

sort\_choice = TerminalMenu(

["Bubble Sort", "Selection Sort", "Insertion Sort", "Quick Sort"], *title*="\nWhat sorting algorithm would you like to use?").show()

*if* sort\_choice == 0:

algos.bubbleSort()

*elif* sort\_choice == 1:

algos.selectionSort()

*elif* sort\_choice == 2:

algos.insertionSort()

*else*:

algos.quicksort()

*elif* sort\_or\_search == 1:

target = input("Please input a target value for searching: ")

target = int(target) *if* listType == "numeric" *else* target

search\_choice = TerminalMenu(

["Linear search", "Binary search"], *title*=f"Thanks! What searching algorithm would you like to use to find {target}?").show()

*if* search\_choice == 0:

algos.linearSearch(target)

*else*:

algos.binarySearch(target)

*elif* sort\_or\_search == 2:

algos.printList()

*elif* sort\_or\_search == 3:

list\_type = TerminalMenu(

["Numeric list", "Word list"], *title*="What kind of list would you like to generate?").show()

listType = "numeric" *if* list\_type == 0 *else* "word"

size = input(

"What size list? Please enter a positive integer: ")

*try*:

size = int(size)

*except* ValueError:

print("That's not an integer! Play nice or leave!")

*return* exit()

*if* list\_type == 0:

algos.generateNumericList(size)

*else*:

algos.generateWordList(size)

*else*:

print("goodbye!")

exit()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT**

Please see attached video in assignment submission!! Only 3 minutes long ☺