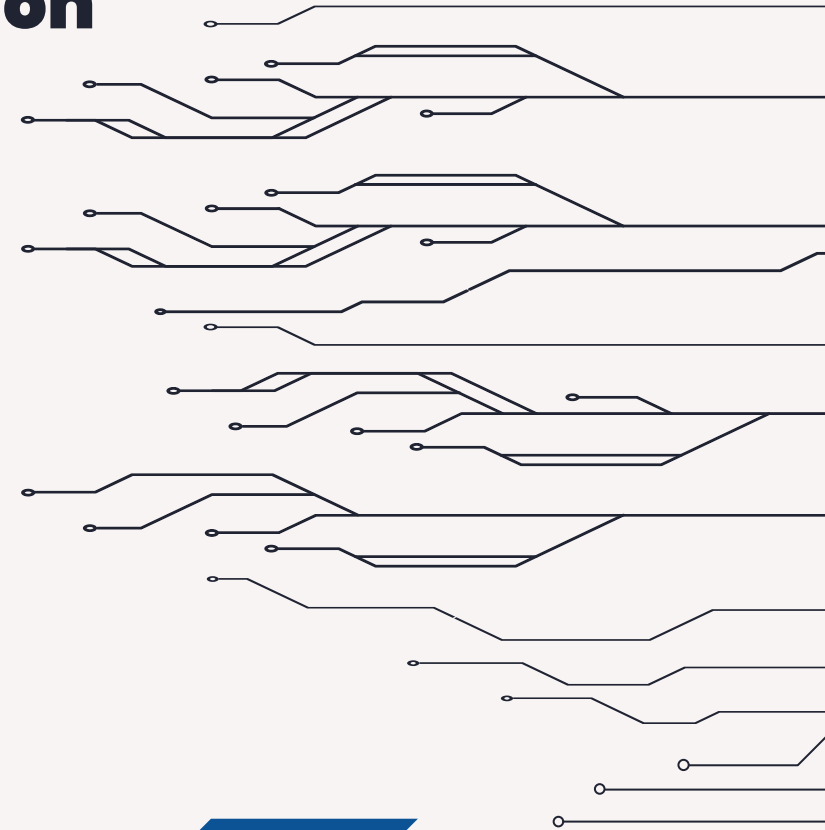
DATA STRUCTURES & ALGORITHMS

CEIS295
Faith Burnett
04/15/2024

Introduction

- Array-Based Implementation
- Linked Lists
- Stacks and Queues
- Sorting Algorithms
- Search Techniques and Hashing
- Binary Trees



ArrayList Real-World Speeds

- Create sample code for calculating speeds
- Create a Client Class
- Create method for displaying name and date for authenticity
- Read data in from file
- Test speeds adding, displaying and removing from ArrayList
- Use Quicksort to test speeds of sorted data
- Create an Excel table to track speeds

Sample Code For Calculating Speeds

E: > CEIS295 > Week 1 Project > TimeProcess.py > ...

```python
#Name: Faith Burnett
#Date:  04/11/2024
import time    # use time library to time the code executions
from datetime import date # use datetime library to display current date in output

# display name and date in output
def nameAndDate():
    baseDate = date.today()
    today = baseDate.strftime('%B %d, %Y')
    print(f"Name: Faith Burnett\nDate: {today}\n")

# output phrase 1000 times
def phraseOutputSpeed():
    # get current time before the process
    start_time = time.time()

    # run the process
    for i in range(1000):
        print( "Hello Everyone!" )

    # get current time after the process
    end_time = time.time()

    # subtract start time from end time to get time used by process
    elapsed_time = end_time - start_time

    # Show the result.  Note: .6f means "show six decimal places"
    print(f"Seconds to run phrase 1000 times: {elapsed_time}")

if __name__ == "__main__":
    nameAndDate()
    phraseOutputSpeed()
```

Client Class

Client.py > Client

```python
#Name: Faith Burnett
#Date:  04/17/2024

class Client:

    #initialize variables
    def __init__(self, client_id=0, first_name="Unknown", last_name="Unknown", phone="Unknown", email="Unknown"):
        self.__client_id = client_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__phone = phone
        self.__email = email

    #Less than method for comparing instances of Client
    def __lt__(self, other):
        return self.__client_id < other.__client_id

    #equals method for comparing instances of Client
    def __eq__(self, other):
        return self.__client_id == other.__client_id

    #string method displays the preferred data from Client
    def __str__(self):
        return str(self.__client_id) + ", " + self.__last_name + ", " + self.__first_name
```

Client.py > ...

```python

    #define getters and setters
    def get_client_id(self):
        return self.__client_id

    def set_client_id(self, client_id):
        self.__client_id = client_id

    def get_first_name(self):
        return self.__first_name

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def get_last_name(self):
        return self.__last_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def get_phone(self):
        return self.__phone

    def set_phone(self, phone):
        self.__phone = phone

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email
```

Name and Date Method

ArrayListActualSpeed.py > ...

```python
#Name: Faith Burnett
#Date:  04/17/2024

from ArrayList import ArrayList
from Client import Client
from datetime import date
import time, random

def nameAndDate():
    baseDate = date.today()
    today = baseDate.strftime('%B %d, %Y')
    print(f"Name: Faith Burnett\nDate: {today}\n")

```

```python
if __name__ == "__main__":

    nameAndDate()
```

The **nameAndDate()** method displays the authors name and the current date using the datetime library.

In the main method we call the method to be ran

Read Clients from File

The **read_clients_from_csv()** method reads the client data from a csv file, assigns each element to an index of a Client instance and then that instance is added to an empty list. It takes **file_path** as a parameter.

```
14  def read_clients_from_csv(file_path):
15      clients = []
16      with open(file_path) as infile:
17          for line in infile:
18              s = line.split(',')
19              client_id = int(s[0])
20              first_name = s[1]
21              last_name = s[2]
22              phone = s[3]
23              email = s[4]
24              clt = Client(client_id, first_name, last_name, phone, email)
25              clients.append(clt)
26      return clients
27
```

In the main method we declare the **file_path** which is set to the location of the csv file.
Next, we declare the **client_list** variable which we set to **read_clients_from_csv** method that takes **file_path** as it's parameter.

```
107 if __name__ == "__main__":
108
109     nameAndDate()
110
111     file_path = 'ClientData.csv'
112     client_list = read_clients_from_csv(file_path)
113
```

Add Clients to ArrayList

```python
def addClientsSpeed(clientList, clientArray):

    start_time = time.time()

    for i in range(len(clientList)):
        clientArray.append(clientList[i])

    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"Time taken to add {len(clientList)} Client objects to ArrayList: {elapsed_time} seconds\n")
```

```python
if __name__ == "__main__":

    nameAndDate()

    file_path = 'ClientData.csv'
    client_list = read_clients_from_csv(file_path)

    client_array = ArrayList()

    addClientsSpeed(client_list, client_array)
```

The first action that we need to test the speed for is adding the unsorted list of Client objects(clientList) to the ArrayList(clientArray). I created the addClientsSpeed() method which will loop through the Client objects in clientList and append them to clientArray. In the main method we have the clientList(client_list) and we create an instance of ArrayList(client_array). The last thing to do is call addClientsSpeed() and insert the parameters which are client_list and client_array.

Add Clients Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.ex
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 19, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0019989013671875 seconds

PS E:\CEIS295\Week 1 Project>
```

Remove Clients from ArrayList

Next, we want to remove all Client objects from the front of the clientArray. We do this by creating the removeClientSpeed() method. This method loops through the Client objects and removes them one by one from index 0 of the clientArray.

In the main method we add removeClientSpeed() with it's parameters client_list and client_array.

```
41  def removeClientsSpeed(clientList, clientArray):
42
43      start_time = time.time()
44
45      for i in range(len(clientList)):
46          clientArray.remove_at(0)
47
48      end_time = time.time()
49
50      elapsed_time = end_time - start_time
51
52
53      print(f"Time taken to remove {len(clientList)} Client objects from ArrayList: {elapsed_time} seconds\n")
54
```

```
107 if __name__ == "__main__":
108
109     nameAndDate()
110
111     file_path = 'ClientData.csv'
112     client_list = read_clients_from_csv(file_path)
113
114     client_array = ArrayList()
115
116     addClientsSpeed(client_list, client_array)
117     removeClientsSpeed(client_list, client_array)
118
```

Remove Clients Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 19, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0019998550415039062 seconds

Time taken to remove 10000 Client objects from ArrayList: 2.1972689628601074 seconds

PS E:\CEIS295\Week 1 Project>
```

Display Random Clients from ArrayList

```python
def getRandomRecords(clientList, clientArray):

    start_time = time.time()

    for i in range(1000):
        min_id = 100001
        max_id = min_id + len(clientList)
        r_num = random.randint(min_id, max_id)
        print(clientArray.search(Client(r_num)))

    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"\nTime taken to get a random number of Client objects from ArrayList:\n{elapsed_time} seconds")
```

```python
if __name__ == "__main__":

    nameAndDate()

    file_path = 'ClientData.csv'
    client_list = read_clients_from_csv(file_path)

    client_array = ArrayList()

    addClientsSpeed(client_list, client_array)
    getRandomRecords(client_list, client_array)

```

The getRandomRecords() method loops through the Client objects, sets a random variable and searches the clientArray at indices that match the random number. It will return the Client object at the random indices.

In the main method we call the addClientsSpeed() method again since we deleted them all previously. After we call getRandomRecords() and insert the parameters.

Display Random Clients Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS295/Week 1 Project
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 19, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0019996166229248047 seconds

104710, Flowers, Christian
100371, Jacobs, Teegan
109902, Church, Isadora
103064, Lowery, Plato
102740, Oneil, Chiquita
105876, Roberts, Freya
100911, Hart, Zephania
109248, Myers, Brock
108770, Joyner, Virginia
102749, Finley, Jaquelyn
109372, Moss, Victor
108405, Aguirre, Eleanor
100298, Sandoval, Blaze
103591, Parks, Miriam
109551, Harvey, Kibo
105191, James, Savannah
105135, Burris, Moses
100129, Porter, Brett
103966, Pierce, Martina
105205, Weeks, Abdul
105047, Stout, Ashton
108579, Dickerson, Hiram
105442, Delacruz, Irene
101704, Reilly, Lillith
108319, Zamora, Lillian

Time taken to get a random number of Client objects from ArrayList:
0.7263095378875732 seconds
PS E:\CEIS295\Week 1 Project>
```

Add, Display Random, Remove Random Clients

In this final unsorted test we create the addDisplayRemoveSpeed() method. This method adds some Client objects to the ArrayList, displays random Client objects and removes random Client objects.

In the main method we remove previous calls and call the addDisplayRemoveSpeed() method and add the parameters.

```
def addDisplayRemoveSpeed(clientList, clientArray):

    r_num = random.randint(0, 100001)
    start_time = time.time()

    print("Adding clients...\n")

    #add client
    for i in range(5000):
        clientArray.append(clientList[i])

    print("Displaying random clients...\n")

    #display random clients
    for r_num in range(1000):
        min_id = 100001
        max_id = min_id + len(clientList)
        r_num = random.randint(min_id, max_id)
        print(clientArray.search(Client(r_num)))

    print("\n")

    #remove random clients
    for r_num in range(1000):
        clientArray.remove_at(r_num)

    print("Removing random clients...\n")

    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"Time taken to add, display and delete client objects in ArrayList:\n{elapsed_time} seconds\n")

if __name__ == "__main__":

    nameAndDate()

    file_path = 'ClientData.csv'
    client_list = read_clients_from_csv(file_path)

    client_array = ArrayList()

    addDisplayRemoveSpeed(client_list, client_array)
```

Add, Display Random, Remove Random Clients Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS295/
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 19, 2024

Adding clients...

Displaying random clients...

105840, Tate, Teegan
102563, Cline, John
102482, Holman, Lila
108092, Schmidt, Elton
100369, Clements, Logan
102457, Lamb, Jack
104063, Sweeney, James
104473, Haynes, Magee
100583, Vinson, Sawyer
101139, Fry, Daphne
105948, Fuentes, Piper
103832, Merrill, Mira
109384, Wynn, Iris
104444, Lang, Hyacinth
104505, Porter, Tyler
108843, Randall, Orla
104031, Pruitt, Jeremy
108420, Lott, Alma
106129, Bender, Martena
102224, Turner, Naomi
102835, Herrera, Octavius
108498, Jarvis, Lesley
100980, Castro, Isaiah
107116, Preston, Deanna
101678, Delacruz, Hedy
104799, Mayo, Talon
109704, Jacobson, Alden
108349, Weaver, Hayes


Removing random clients...

Time taken to add, display and delete client objects in ArrayList:
1.1127536296844482 seconds
```

Read Clients from File Quicksort

```
15  def read_clients_from_csv(file_path):
16      clients = []
17      with open(file_path) as infile:
18          for line in infile:
19              s = line.split(',')
20              client_id = int(s[0])
21              first_name = s[1]
22              last_name = s[2]
23              phone = s[3]
24              email = s[4]
25              clt = Client(client_id, first_name, last_name, phone, email)
26              clients.append(clt)
27      Quicksort.sort(clients)
28      return clients
```

Our next action is to use Quicksort to sort the Client objects. We'll add this after we've appended them all to clients.

Updated Sorted Search Method

```
73  def addDisplayRemoveSpeed(clientList, clientArray):
74
75      r_num = random.randint(0, 100001)
76      start_time = time.time()
77
78      print("Adding clients...\n")
79
80      #add client
81      for i in range(5000):
82          clientArray.append(clientList[i])
83
84      print("Displaying random clients...\n")
85
86      #display random clients
87      for r_num in range(1000):
88          min_id = 100001
89          max_id = min_id + len(clientList)
90          r_num = random.randint(min_id, max_id)
91          print(clientArray.search_sorted(Client(r_num)))
92
93      print("\n")
```

```
57  def getRandomRecords(clientList, clientArray):
58
59      start_time = time.time()
60
61      for i in range(1000):
62          min_id = 100001
63          max_id = min_id + len(clientList)
64          r_num = random.randint(min_id, max_id)
65          print(clientArray.search_sorted(Client(r_num)))
66
67      end_time = time.time()
68
69      elapsed_time = end_time - start_time
70
71      print(f"\nTime taken to get a random number of Client objects from ArrayList:\n{elapsed_time} seconds")
72
```

Add Clients Quicksort Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0019998550415039062 seconds

PS E:\CEIS295\Week 1 Project>
```

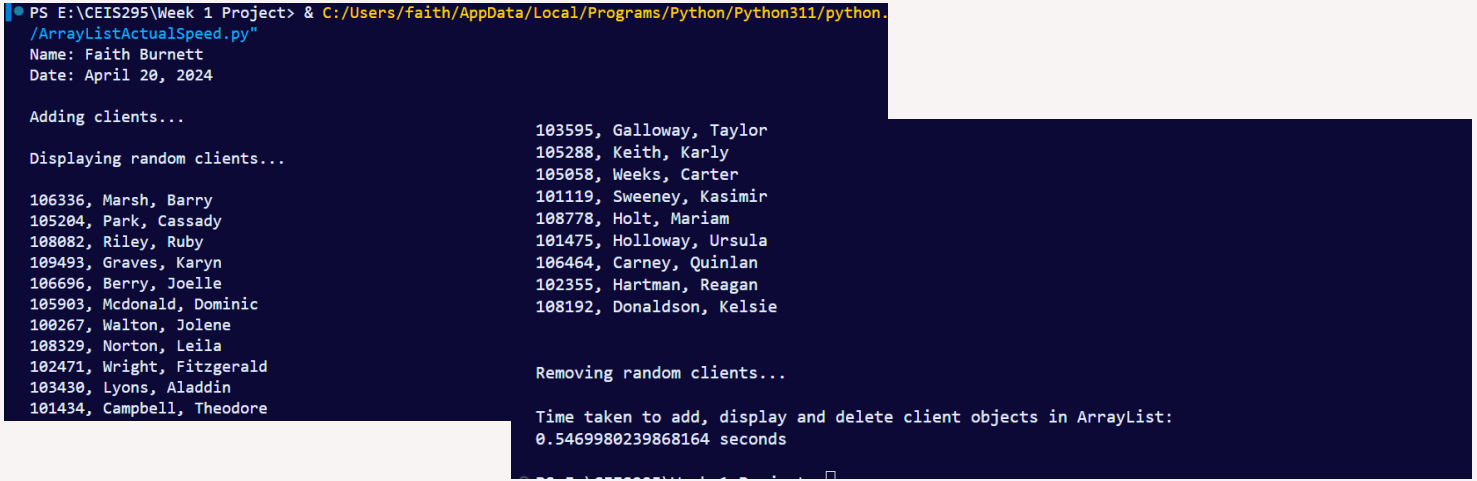
Remove Clients Quicksort Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0020008087158203125 seconds

Time taken to remove 10000 Client objects from ArrayList: 2.4200711250305176 seconds

PS E:\CEIS295\Week 1 Project>
```

Display Random Clients Quicksort Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CE
/ArrayListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to ArrayList: 0.0019998550415039062 seconds

101911, Small, Quemby
108422, Hess, Damon
108541, Jackson, Lucas
102357, Huber, Jeanette
108272, May, Donna
109382, Gutierrez, Pearl
101138, Navarro, Francesca
103392, Hinton, Driscoll
105774, Wade, Evangeline
103168, Rodriguez, Erasmus
101247, Webb, Ezra
107134, Mullins, Ross
109949, Blake, Carolyn
100838, Poole, Kennedy
103702, Fuller, Tyrone
103678, Head, Tashya
108888, Mitchell, Eve
101931, Mckenzie, Cairo
106948, Wise, Eden
107950, Chang, Sebastian
108794, Garrison, Adam
104008, Hobbs, Stephen

Time taken to get a random number of Client objects from ArrayList:
0.08999800682067871 seconds
PS E:\CEIS295\Week 1 Project>
```

Add, Display Random, Remove Random Clients Quicksort Output



ArrayList Real World Speeds

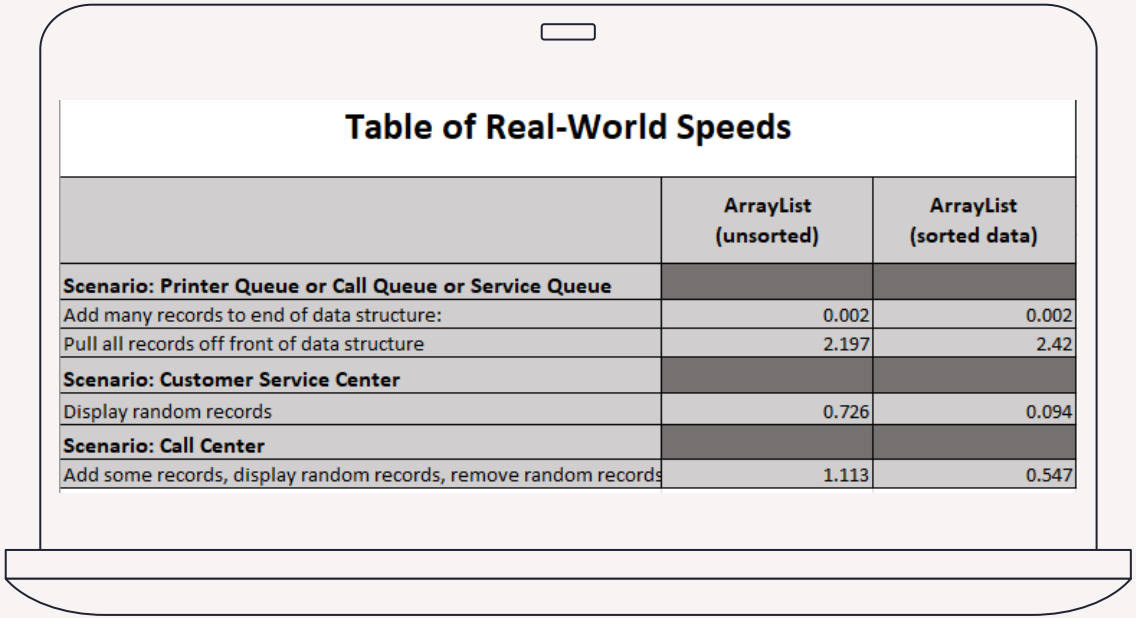


Table of Real-World Speeds

	ArrayList (unsorted)	ArrayList (sorted data)
Scenario: Printer Queue or Call Queue or Service Queue		
Add many records to end of data structure:	0.002	0.002
Pull all records off front of data structure	2.197	2.42
Scenario: Customer Service Center		
Display random records	0.726	0.094
Scenario: Call Center		
Add some records, display random records, remove random records	1.113	0.547

Linked List Real-World Speeds

- Test speeds for adding to the Linked List
- Test speeds for displaying and removing from LinkedList
- Create an Excel table to track speeds

Add Clients to Linked List

```
28  def addClients(clientList, linkedList):
29      start_time = time.time()
30
31      for i in range(len(clientList)):
32          linkedList.add_last(clientList[i])
33
34      end_time = time.time()
35
36      elapsed_time = end_time - start_time
37
38      print(f"Time taken to add {len(clientList)} Client objects to LinkedList: {elapsed_time} seconds\n")
```

```
94  if __name__ == "__main__":
95
96      nameAndDate()
97
98      client_list = read_clients_to_file('ClientData.csv')
99      linked_list = LinkedList()
100
101     addClients(client_list, linked_list)
102
```

The first action that we need to test the speed for is adding the list of Client objects(clientList) to the LinkedList(linkedList). I created the addClients() method which will loop through the Client objects in clientList and add them to linkedList using add_last(). In the main method we have the clientList(client_list) and we create an instance of LinkedList(linked_list). The last thing to do is call addClients() and insert the parameters which are client_list and linked_list.

Add Clients Output

```
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to LinkedList: 0.0030009746551513 67 seconds

PS E:\CEIS295\Week 2 Project>
```

Remove Clients from Linked List

Next, we want to remove all Client objects from the front of the linkedList. We do this by creating the removeClients() method. This method loops through the Client objects and removes them using the remove_first() method.

In the main method we add removeClients() with it's parameters client_list and linked_list.

```
40  def removeClients(clientList, linkedList):
41      start_time = time.time()
42
43      for i in range(len(clientList)):
44          linkedList.remove_first()
45
46      end_time = time.time()
47
48      elapsed_time = end_time - start_time
49
50      print(f"Time taken to remove {len(clientList)} Client objects from LinkedList: {elapsed_time} seconds\n")
```

```
92  if __name__ == "__main__":
93
94      nameAndDate()
95
96      client_list = read_clients_to_file('ClientData.csv')
97      linked_list = LinkedList()
98
99      addClients(client_list, linked_list)
100     removeClients(client_list, linked_list)
```

Remove Clients Output

```
PS E:\CEIS295\Week 2 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS
/LinkedListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to LinkedList: 0.0035164356231689453 seconds

Time taken to remove 10000 Client objects from LinkedList: 0.0019974708557128906 seconds

PS E:\CEIS295\Week 2 Project>
```

Display Random Clients from Linked List

```
52  def displayRandom(clientList, linkedList):
53      start_time = time.time()
54
55      for i in range(1000):
56          small_id = 100001
57          large_id = small_id + len(clientList)
58          random_num = random.randint(small_id, large_id)
59          print(linkedList.search(Client(random_num)))
60
61      end_time = time.time()
62
63      elapsed_time = end_time - start_time
64
65      print(f"Time taken to get 1000 random Client objects from LinkedList: {elapsed_time} seconds\n")
66
```

```
92  if __name__ == "__main__":
93
94      nameAndDate()
95
96      client_list = read_clients_to_file('ClientData.csv')
97      linked_list = LinkedList()
98
99      addClients(client_list, linked_list)
100     displayRandom(client_list, linked_list)
101
```

The displayRandom() method loops through the Client objects, sets a random variable and searches the linkedList at indices that match the random number. It will return the Client object at the random indices.

In the main method we call the addClients() method again since we deleted them all previously. After we call displayRandom() and insert the parameters.

Display Random Clients Output

```
PS E:\CEIS295\Week 2 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS295/
/LinkedListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Time taken to add 10000 Client objects to LinkedList: 0.005025148391723633 seconds

105274, Travis, Aileen
105262, Robbins, Shelley
100976, Hickman, Len
105865, Sutton, Lynn
103150, Hopper, Kyle
107786, Ortega, Blake
105709, Fitzgerald, Eric
104415, Swanson, Ramona
106336, Boyer, Odessa
106278, Hamilton, Debra
103883, Patel, Richard
101903, Walker, Jonah
108694, Gilbert, Chava
103608, Mathews, Lacy
109530, Tillman, Abigail
105434, Montgomery, Brandon
102914, Hopkins, Rowan
100035, Pate, Brent
105754, Whitley, Odysseus
102047, Chaney, Garrett
108341, Lambert, Sean
100138, Hurley, Gloria
Time taken to get 1000 random Client objects from LinkedList: 0.6337177753448486 seconds

PS E:\CEIS295\Week 2 Project>
```

Add, Display Random, Remove Random Clients

In this final test we create the addDisplayRemove() method. This method adds some Client objects to the linkedList, displays random Client objects and removes random Client objects.

In the main method we remove previous calls and call the addDisplayRemove() method and add the parameters.

```
def addDisplayRemove(clientList, linkedList):

    start_time = time.time()

    for i in range(len(clientList)):
        linkedList.add_last(clientList[i])

    print("Adding clients...\n")

    print("Displaying clients...\n")
    for i in range(1000):
        small_id = 100001
        large_id = small_id + len(clientList)
        random_num = random.randint(small_id, large_id)
        print(linkedList.search(Client(random_num)))

    print("\nRemoving clients...")
    for i in range(1000):
        small_id = 100001
        large_id = small_id + len(clientList)
        random_num = random.randint(small_id, large_id)
        linkedList.remove(Client(random_num))


    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"Time taken to get clients added, grab random clients and remove random clients: {elapsed_time} seconds\n")
```

```
if __name__ == "__main__":

    nameAndDate()

    client_list = read_clients_to_file('ClientData.csv')
    linked_list = LinkedList()

    addDisplayRemove(client_list, linked_list)
```

Add, Display Random, Remove Random Clients Output

```
PS E:\CEIS295\Week 2 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS
/LinkedListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Adding clients...

Displaying clients...

107026, Jacobson, Levi
101145, Tillman, Eve
106015, Boyd, Fredericka
105596, Winters, Bethany
105706, Scott, Breanna
105194, Swanson, Sawyer
107315, Roth, Amela
106001, Lopez, Dustin
103908, Ball, Madonna
107745, Gross, Christopher
102434, Stafford, Rana
108003, Emerson, Hanae
103255, Frank, Jacqueline
103094, Howe, Marvin
105174, Baker, Warren
102149, Dotson, Meredith
101055, Witt, Erasmus
108694, Gilbert, Chava
109191, Guerrero, Ciaran
109332, Fox, Reed
109736, Fox, Basil
100627, Ratliff, Odette
106246, Nelson, Vincent
100887, Shepard, Ursa
108048, Stark, Deborah

Removing clients...
Time taken to get clients added, grab random clients and remove random clients: 1.1212549209594727 seconds

PS E:\CEIS295\Week 2 Project>
```

Linked List Real World Speeds

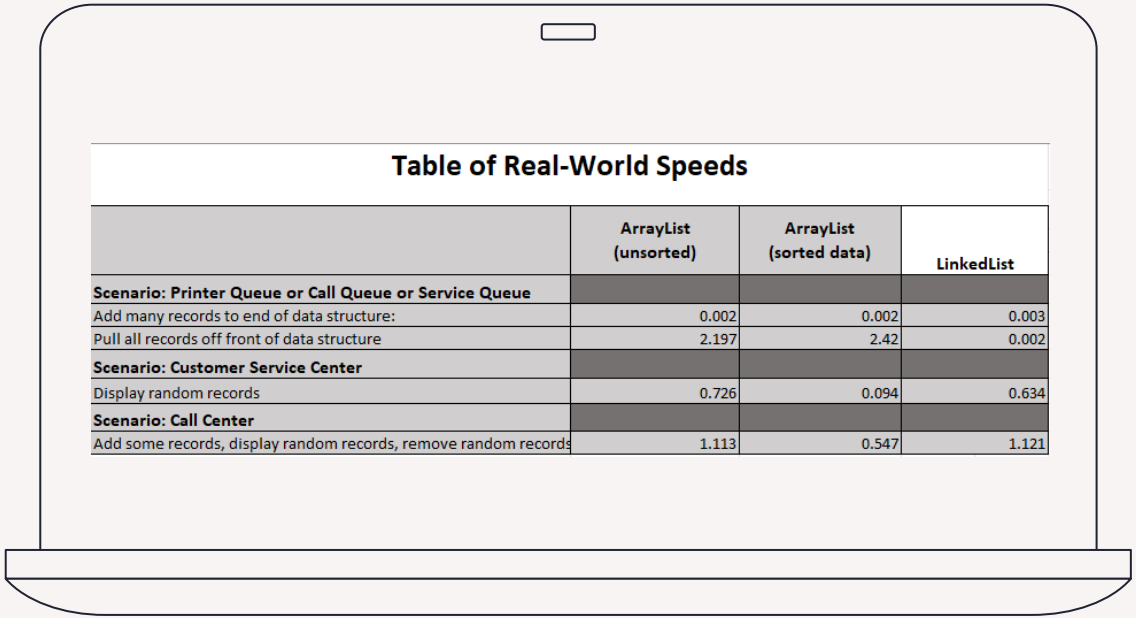


Table of Real-World Speeds

	ArrayList (unsorted)	ArrayList (sorted data)	LinkedList
Scenario: Printer Queue or Call Queue or Service Queue			
Add many records to end of data structure:	0.002	0.002	0.003
Pull all records off front of data structure	2.197	2.42	0.002
Scenario: Customer Service Center			
Display random records	0.726	0.094	0.634
Scenario: Call Center			
Add some records, display random records, remove random records	1.113	0.547	1.121

Automatic Call Distributor Queues

- Create a Call class
- Read data in from file
- Test adding to queue and removing from queue

Read Clients from File

```
15  def read_clients_from_file(filepath):
16      calls = []
17      with open(filepath) as infile:
18          for line in infile:
19              s = line.split(',')
20              client_id = int(s[0])
21              client_name = s[1]
22              client_phone = s[2]
23              call = Call(client_id, client_name, client_phone)
24              calls.append(call)
25          return calls
```

The **read_clients_from_file()** method reads the client data from a csv file, assigns each element to an index of a Call instance and then that instance is added to an empty list. It takes **file_path** as a parameter.

In the main method we declare the **file_path** which is set to the location of the csv file. Next, we declare the **call_list** variable which we set to **read_clients_from_file()** method that takes **file_path** as it's parameter.

Add and Remove Calls Using Queue

```
def event1(callList, callsWaiting, callNumber):
    print("Call received. Caller added to queue.")
    rand = random.randint(0, len(callList))
    callsWaiting.enqueue(callList[rand])
    callNumber += 1
    print("\tNumber of calls waiting in queue:", callsWaiting.get_length())
```

```
def event2(callsWaiting):
    print("Call sent to representative for service.")
    if callsWaiting.get_length() > 0:
        print("Caller information:")
        print(callsWaiting.dequeue())
    else:
        print("The call waiting queue is empty.")
        print("\tNumber of calls waiting in queue:", callsWaiting.get_length())
```

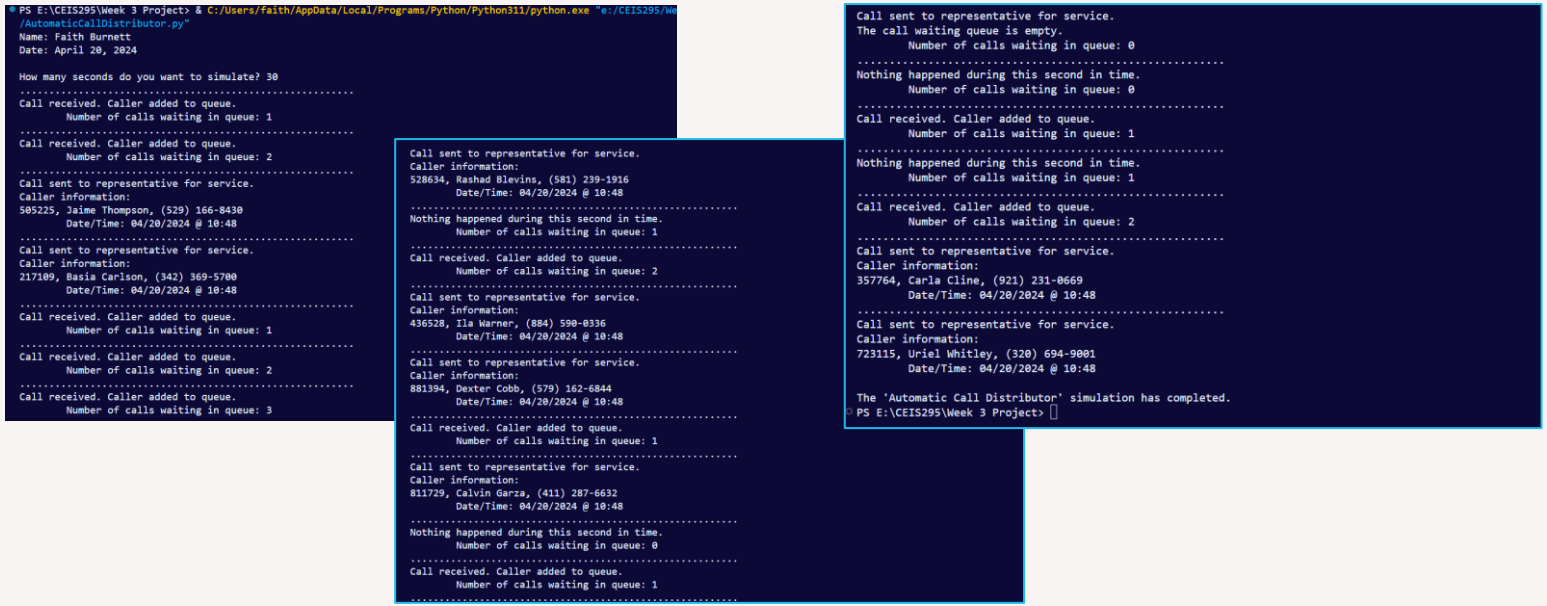
```
if __name__ == "__main__":
    nameAndDate()
    call_list = read_clients_from_file('CallsData.csv')
    calls_waiting = Queue()
    call_number = 0

    seconds = int(input("How many seconds do you want to simulate? "))

    for i in range(seconds):
        print("..........")
        time.sleep(2)
        rand_num = random.randint(1, 3)

        if rand_num == 1:
            event1(call_list, calls_waiting, call_number)

        elif rand_num == 2:
            event2(calls_waiting)

        else:
            print("Nothing happened during this second in time.")
            print("\tNumber of calls waiting in queue:", calls_waiting.get_length())

    print("\nThe 'Automatic Call Distributor' simulation has completed.")
```

In event1() we will create a random number variable. That will grab a random Client from callList and put it in the callsWaiting Queue using enqueue().

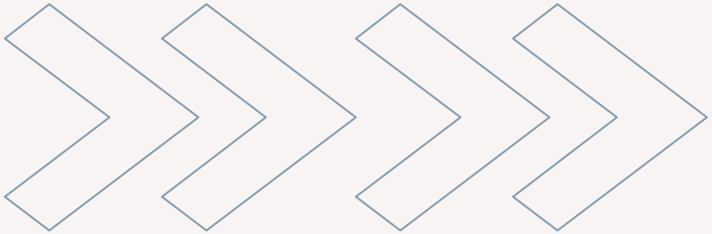
The next method is event2(), here is where the Client will be removed from callsWaiting using dequeue().

In the main method we declare our call_list, calls_waiting and call_number variables. Then we ask the user the seconds they'd like the simulation to last.
Once input we will loop through the Queue either calling event1() or event(2) depending on the contents of the Queue.

Call Distributor Output



Sorting Algorithm Speeds



- Create a Client class
- Test the speed of each sorting algorithm
- Create an Excel table to track speeds

Bubble Sort

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      #client_list = read_clients_to_file("ClientData1000.csv")
39      #client_list  = read_clients_to_file("ClientData10000.csv")
40      #client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      BubbleSort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
```

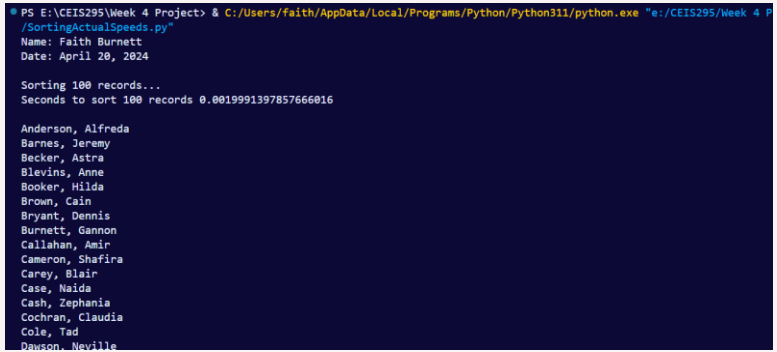
Here we use the same read_clients_to_file() method as we have in previous modules.

For each dataset we only need to uncomment the one we'd like to use.

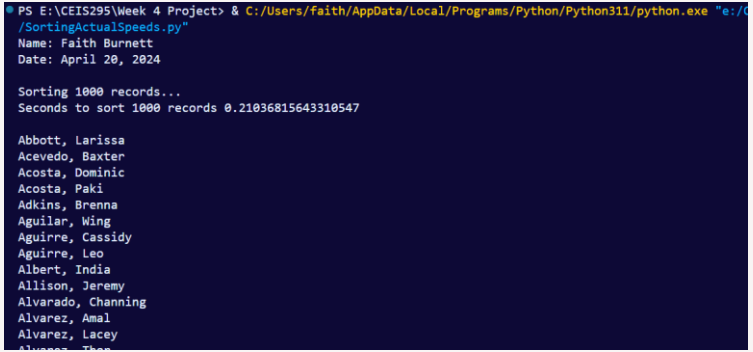
Next, we test the speed of BubbleSort using the sort() method.

Bubble Sort Output

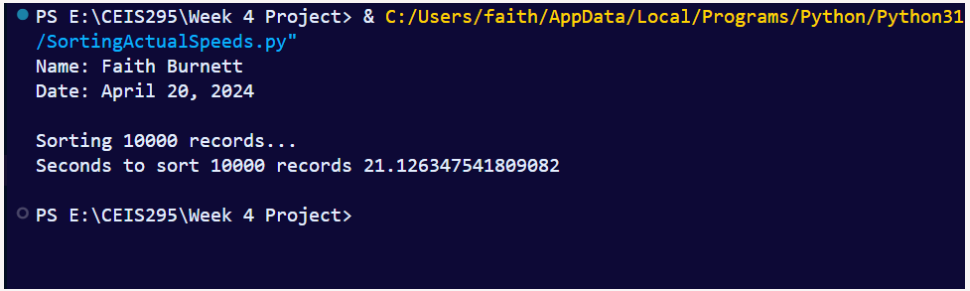
Bubble Sort 100



Bubble Sort 1000



Bubble Sort 10000



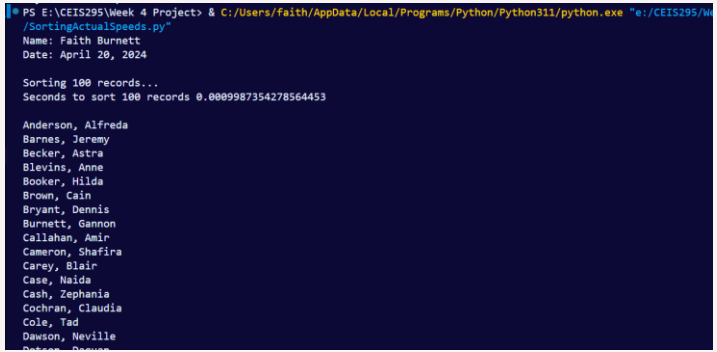
Selection Sort

Testing the speed of SelectionSort using the sort() method.

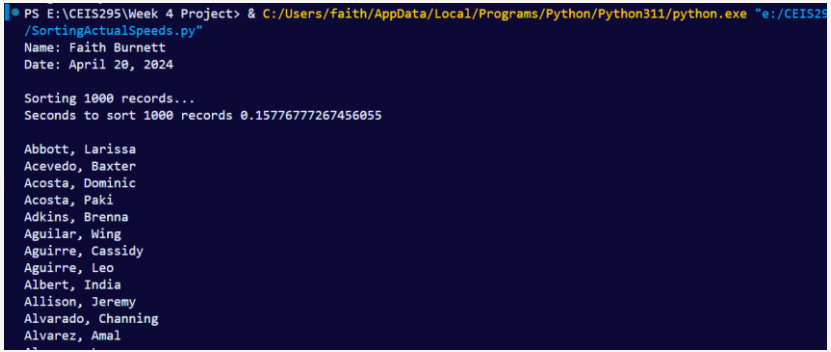
```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      SelectionSort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
```

Selection Sort Output

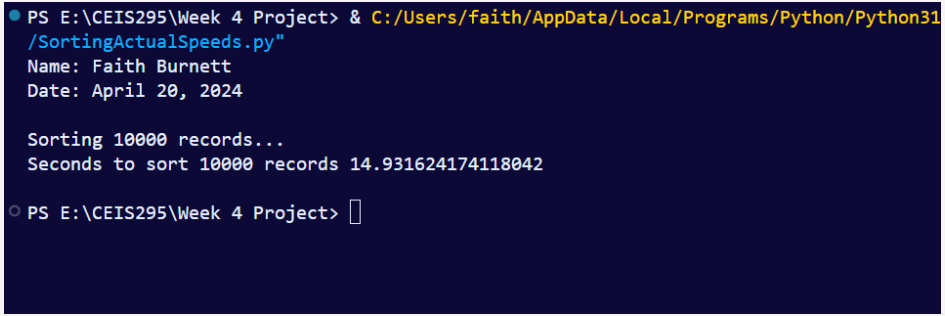
Selection Sort 100



Selection Sort 1000



Selection Sort 10000



Insertion Sort

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      InsertionSort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
```

Testing speed of InsertionSort using the sort() method.

Insertion Sort Output

Insertion Sort 100

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100 records...
Seconds to sort 100 records 0.0009999275207519531

Anderson, Alfreda
Barnes, Jeremy
Becker, Astra
Blevins, Anne
Booker, Hilda
Brown, Cain
Bryant, Dennis
Burnett, Gannon
Callahan, Amir
Cameron, Shafira
Carey, Blair
Case, Naida
Cash, Zephania
Cochran, Claudia
Cole, Tad
Dawson, Neville
Dotson, Daquan
```

Insertion Sort 1000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 1000 records...
Seconds to sort 1000 records 0.10294055938720703

Abbott, Larissa
Acevedo, Baxter
Acosta, Dominic
Acosta, Paki
Adkins, Brenna
Aguilar, Wing
Aguirre, Cassidy
Aguirre, Leo
Albert, India
Allison, Jeremy
Alvarado, Channing
Alvarez, Amal
Alvarez, Lacey
Alvarez, Thor
Anderson, Ila
Anthony, Donovan
```

Insertion Sort 10000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 10000 records...
Seconds to sort 10000 records 9.21253752708435

PS E:\CEIS295\Week 4 Project>
```

Shell Sort

Testing the speed of ShellSort using the sort() method.

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      ShellSort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
58
```

Shell Sort Output

Shell Sort 100

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100 records...
Seconds to sort 100 records 0.0

Anderson, Alfreda
Barnes, Jeremy
Becker, Astra
Blevins, Anne
Booker, Hilda
Brown, Cain
Bryant, Dennis
Burnett, Gannon
Callahan, Amir
Cameron, Shafira
Carey, Blair
Case, Naida
Cash, Zephania
Cochran, Claudia
Cole, Tad
Dawson, Neville
```

Shell Sort 1000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 1000 records...
Seconds to sort 1000 records 0.005997657775878906

Abbott, Larissa
Acevedo, Baxter
Acosta, Dominic
Acosta, Paki
Adkins, Brenna
Aguilar, Wing
Aguirre, Cassidy
Aguirre, Leo
Albert, India
Allison, Jeremy
Alvarado, Channing
Alvarez, Amal
Alvarez, Lacey
Alvarez, Thor
```

Shell Sort 10000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 10000 records...
Seconds to sort 10000 records 0.1096031665802002

Abbott, Brittany
Abbott, Catherine
Abbott, Darius
Abbott, Farrah
Abbott, Garrison
Abbott, Harrison
Abbott, Kellie
Abbott, Kylie
Abbott, Olympia
Abbott, Sean
Abbott, Susan
Acevedo, Abbot
Acevedo, Amir
```

Shell Sort 100000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100000 records...
Seconds to sort 100000 records 2.004929780960083

PS E:\CEIS295\Week 4 Project>
```

Quicksort

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      Quicksort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
```

Testing speed of Quicksort using the sort() method.

Quicksort Output

Quicksort 100

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100 records...
Seconds to sort 100 records 0.0

Anderson, Alfreda
Barnes, Jeremy
Becker, Astra
Blevins, Anne
Booker, Hilda
Brown, Cain
Bryant, Dennis
Burnett, Gannon
Callahan, Amir
Cameron, Shafira
Carey, Blair
Case, Naida
Cash, Zephania
Cochran, Claudia
Cole, Tad
```

Quicksort 1000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe "e:/C
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 1000 records...
Seconds to sort 1000 records 0.005135774612426758

Abbott, Larissa
Acevedo, Baxter
Acosta, Dominic
Acosta, Paki
Adkins, Brenna
Aguilar, Wing
Aguirre, Cassidy
Aguirre, Leo
Albert, India
Allison, Jeremy
Alvarado, Channing
Alvarez, Amal
Alvarez, Lacey
Alvarez, Thor
Anderson, Ila
```

Quicksort 10000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/pyth
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 10000 records...
Seconds to sort 10000 records 0.06612110137939453

PS E:\CEIS295\Week 4 Project>
```

Quicksort 100000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.e
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100000 records...
Seconds to sort 100000 records 0.9308252334594727

PS E:\CEIS295\Week 4 Project>
```

Merge Sort

Testing the speed of MergeSort using the sort() method.

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Sorting " +  str(num_records) + " records...")
45
46      start_time = time.time()
47
48      MergeSort.sort(client_list)
49
50      end_time = time.time()
51
52      elapsed_time = end_time - start_time
53
54      print("Seconds to sort " + str(num_records) + " records " + str(elapsed_time) + "\n")
55
56      for client in client_list:
57          print(client)
58
```

Merge Sort Output

Merge Sort 100

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/pytho
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100 records...
Seconds to sort 100 records 0.0

Anderson, Alfreda
Barnes, Jeremy
Becker, Astra
Blevins, Anne
Booker, Hilda
Brown, Cain
Bryant, Dennis
Burnett, Gannon
Callahan, Amir
Cameron, Shafira
Carey, Blair
Case, Naida
Cash, Zephania
Cochran, Claudia
```

Merge Sort 1000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 1000 records...
Seconds to sort 1000 records 0.0049974918365478516

Abbott, Larissa
Acevedo, Baxter
Acosta, Dominic
Acosta, Paki
Adkins, Brenna
Aguilar, Wing
Aguirre, Cassidy
Aguirre, Leo
Albert, India
Allison, Jeremy
Alvarado, Channing
Alvarez, Amal
Alvarez, Lacey
Alvarez, Thor
```

Merge Sort 10000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/py
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 10000 records...
Seconds to sort 10000 records 0.06191849708557129

PS E:\CEIS295\Week 4 Project>
```

Merge Sort 100000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/pyt
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 100000 records...
Seconds to sort 100000 records 0.8521134853363037

PS E:\CEIS295\Week 4 Project>
```

Sorting Real World Speeds

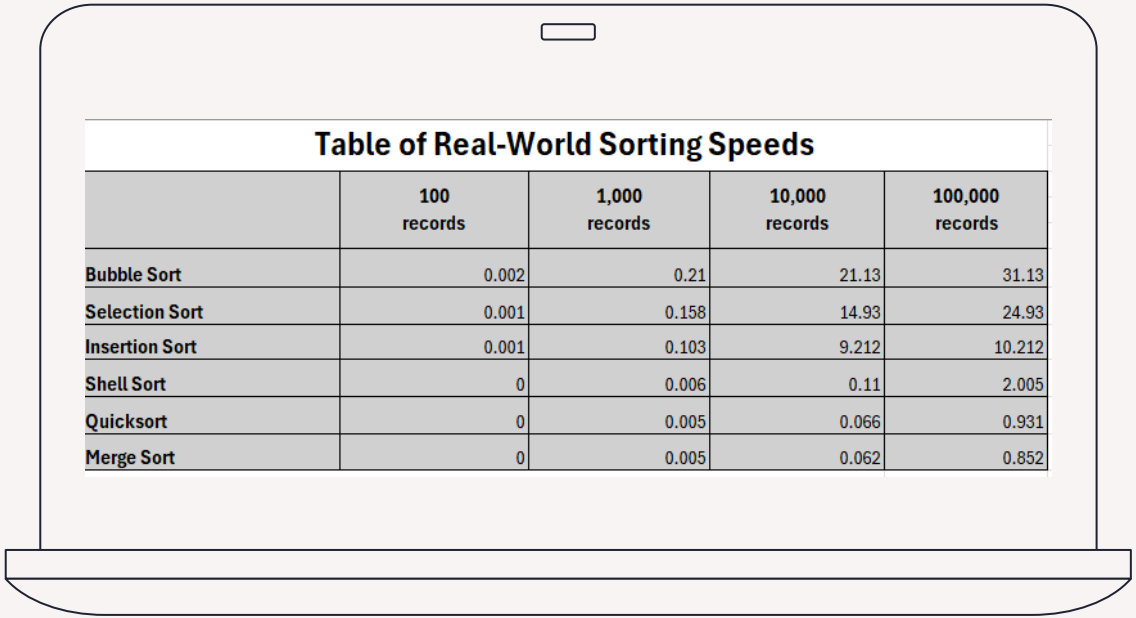
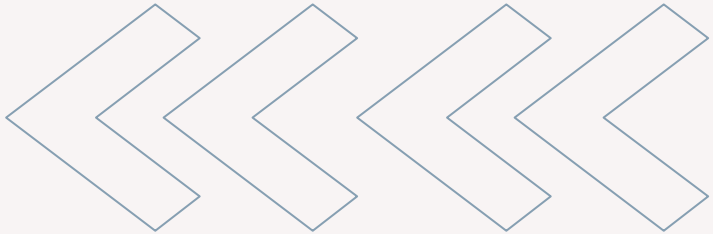


Table of Real-World Sorting Speeds				
	100 records	1,000 records	10,000 records	100,000 records
Bubble Sort	0.002	0.21	21.13	31.13
Selection Sort	0.001	0.158	14.93	24.93
Insertion Sort	0.001	0.103	9.212	10.212
Shell Sort	0	0.006	0.11	2.005
Quicksort	0	0.005	0.066	0.931
Merge Sort	0	0.005	0.062	0.852

Searching Algorithm Speeds

- Create a Client class
- Test the speed of each searching algorithm
- Create an Excel table to track speeds



Linear Search

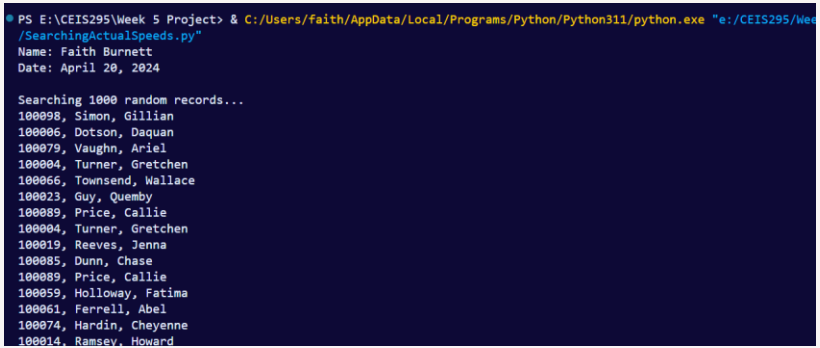
```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      #client_list = read_clients_to_file("ClientData1000.csv")
39      #client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Searching 1000 random records...")
45
46      start_time = time.time()
47
48      for i in range(1000):
49          rand_idx = random.randrange(0, num_records)
50          random_num = client_list[rand_idx]
51          LinearSearch.search(client_list, random_num)
52          print(random_num)
53
54      end_time = time.time()
55
56      elapsed_time = end_time - start_time
57
58      print("Seconds to search 1000 records " + str(elapsed_time) + "\n")
```

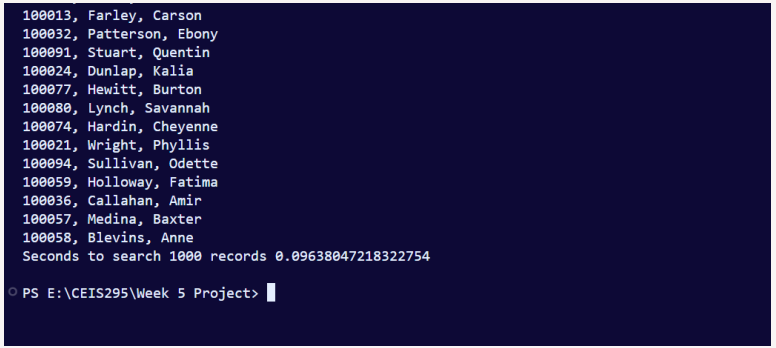
Here we use the same read_clients_to_file() method as we have in previous modules.

For each dataset we only need to uncomment the one we'd like to use.

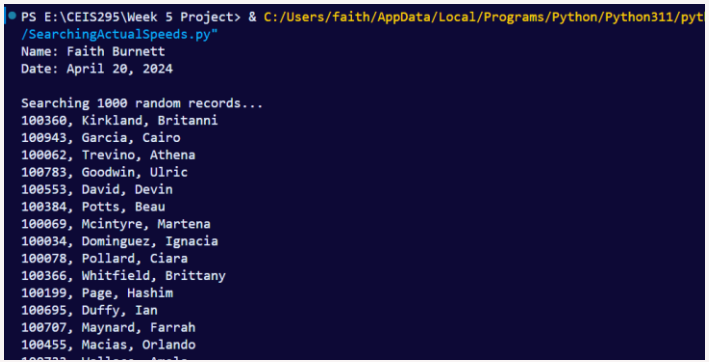
Next, we test the speed of LinearSearch using the search() method.

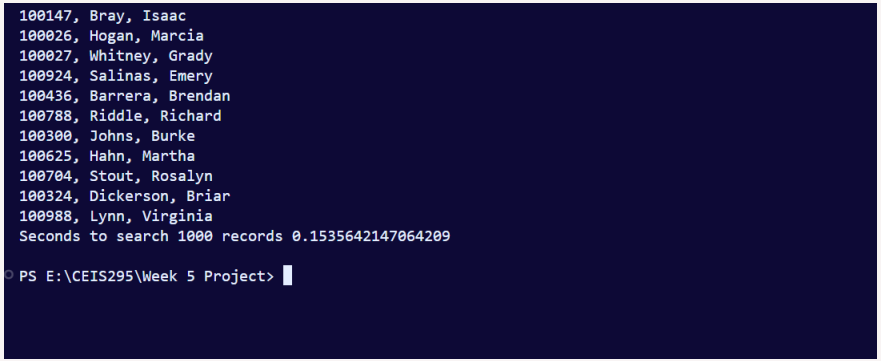
Linear Search 100 Output





Linear Search 1000 Output



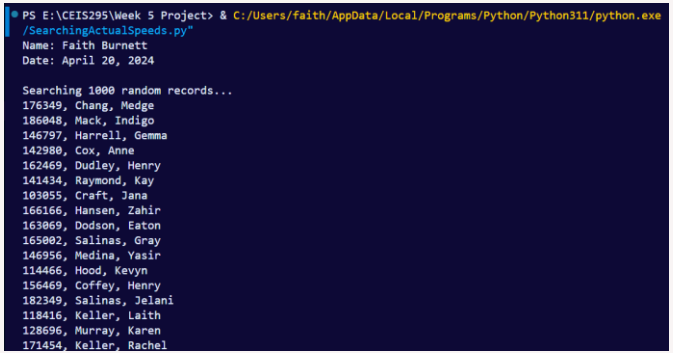


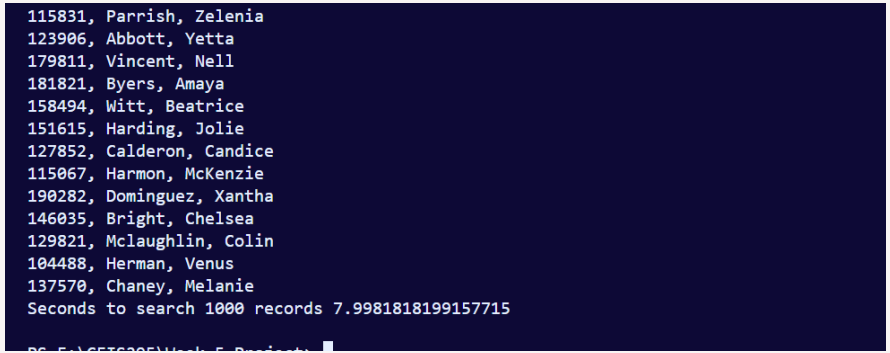
Linear Search 10000 Output

```
PS E:\CEIS295\Week 5 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.
/SearchingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Searching 1000 random records...
109362, Conway, Chaim
109488, Jefferson, Amir
109979, Holmes, Connor
104873, James, Cara
102085, Frye, Shelby
105511, Horne, Barclay
103321, Clemons, Carla
109090, Park, Rashad
100183, Charles, Rashad
106311, Mercer, Abdul
109608, Fletcher, Yeo
105284, Marks, Driscoll
104182, Cash, Hashim
109612, Witt, Aubrey
```

```
103701, Jarvis, Yuli
109830, Quinn, Lunea
106199, Yates, Carl
106060, Parker, Kaseem
108709, Ray, Jin
106916, Conway, Suki
108931, Peters, Nolan
101405, Merritt, Michael
107490, Cervantes, Jena
102867, Henson, Maisie
102643, Jacobs, Price
100589, Ramsey, Anne
Seconds to search 1000 records 0.6988885402679443

PS E:\CEIS295\Week 5 Project>
```

Linear Search 100000 Output



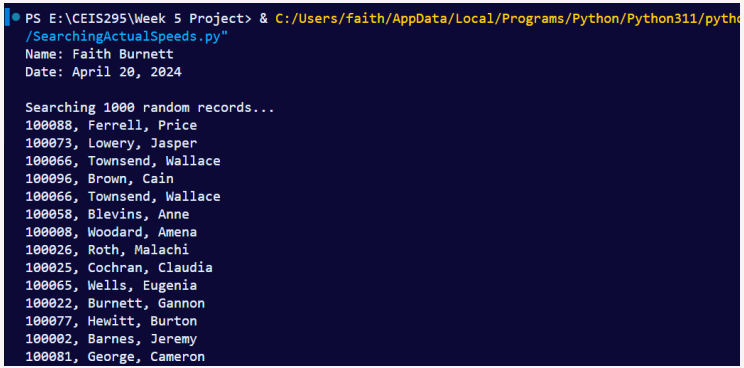


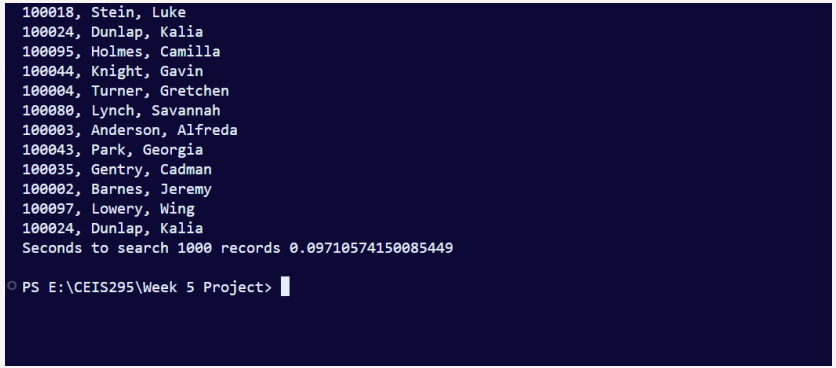
Binary Search

```
34  if __name__ == "__main__":
35      nameAndDate()
36
37      client_list = read_clients_to_file("ClientData100.csv")
38      # client_list = read_clients_to_file("ClientData1000.csv")
39      # client_list  = read_clients_to_file("ClientData10000.csv")
40      # client_list = read_clients_to_file("ClientData100000.csv")
41
42      num_records = len(client_list)
43
44      print("Searching 1000 random records...")
45
46      start_time = time.time()
47
48      for i in range(1000):
49          rand_idx = random.randrange(0, num_records)
50          random_num = client_list[rand_idx]
51          BinarySearch.search(client_list, random_num)
52          print(random_num)
53
54      end_time = time.time()
55
56      elapsed_time = end_time - start_time
57
58      print("Seconds to search 1000 records " + str(elapsed_time) + "\n")
```

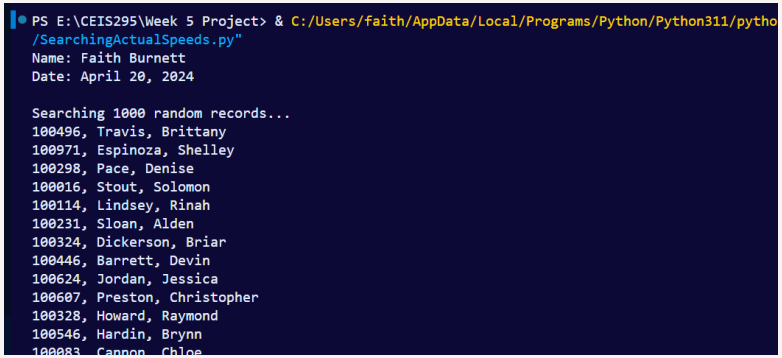
Testing the speed of BinarySearch using the search() method.

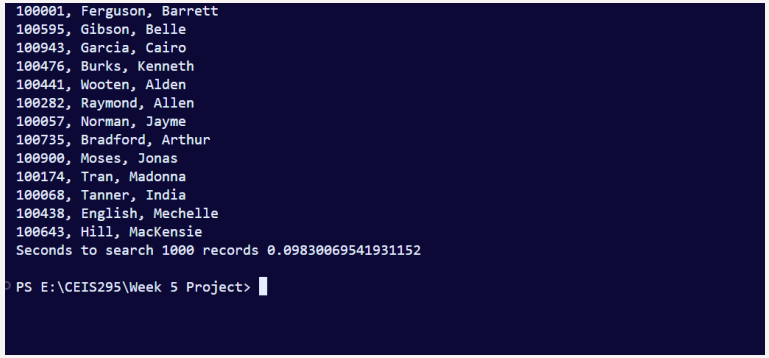
Binary Search 100 Output



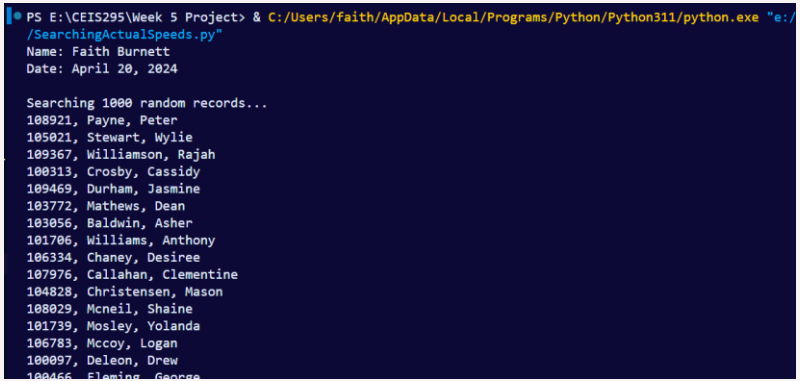


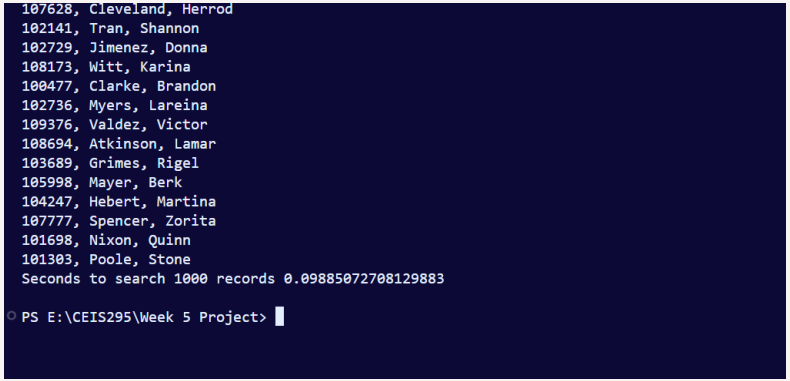
Binary Search 1000 Output



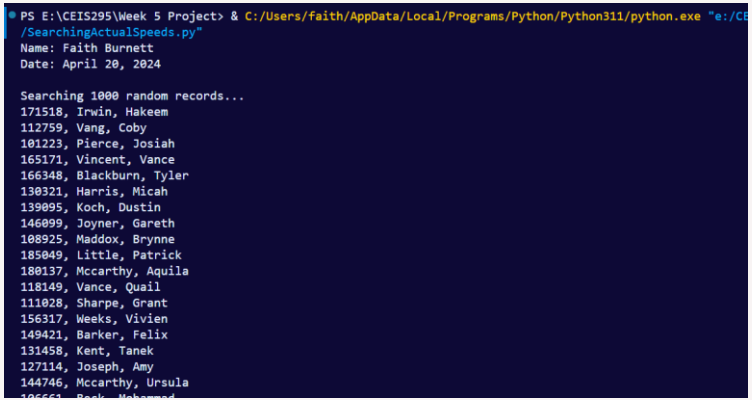


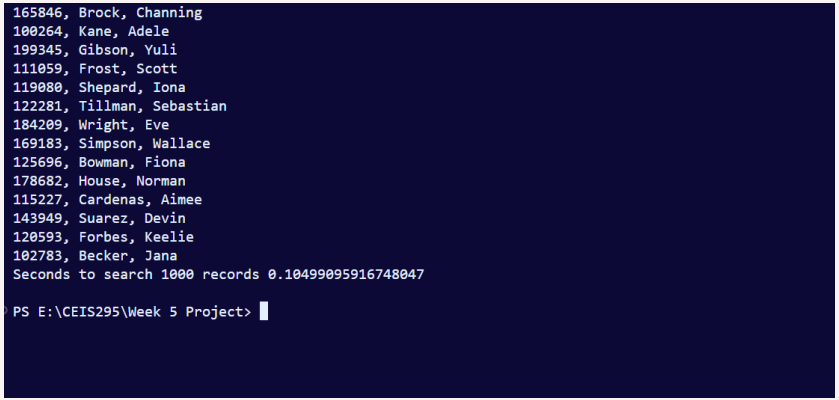
Binary Search 10000 Output





Binary Search 100000 Output





Searching Real World Speeds

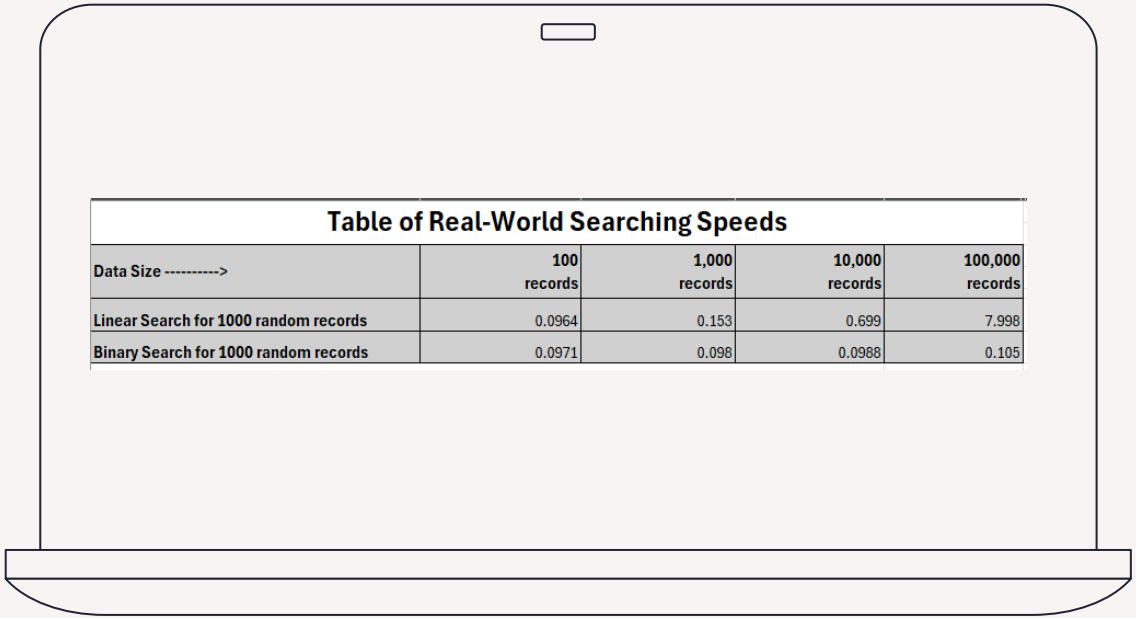


Table of Real-World Searching Speeds				
Data Size ---------->	100 records	1,000 records	10,000 records	100,000 records
Linear Search for 1000 random records	0.0964	0.153	0.699	7.998
Binary Search for 1000 random records	0.0971	0.098	0.0988	0.105

Binary Search Tree Real World Speeds

- Test speeds for adding to the Binary Search Tree
- Test speeds for displaying and removing from the Binary Search Tree
- Create an Excel table to track speeds

Add Clients to Binary Search Tree

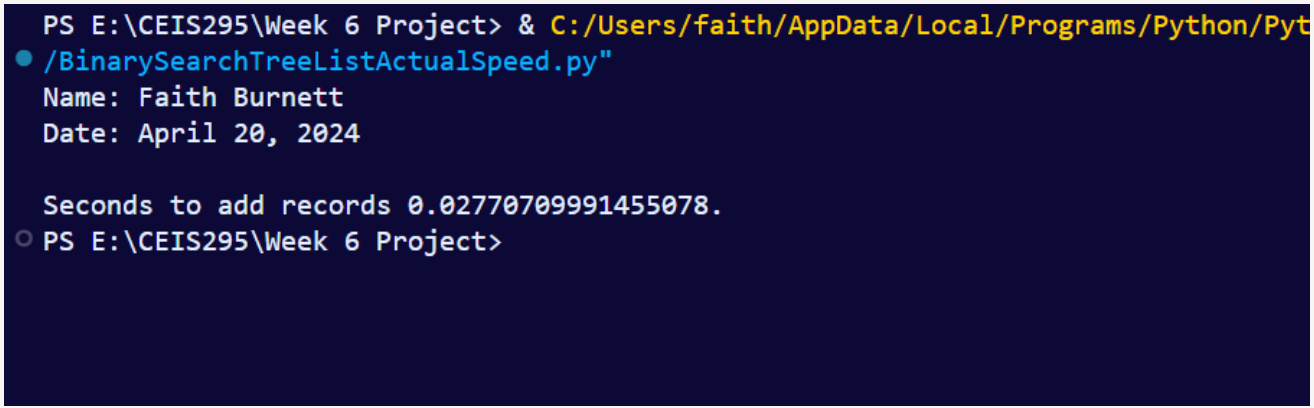
```python
def add_clients(clientList, binarySearch):
    start_time = time.time()

    for client in range(len(clientList)):
        binarySearch.insert(clientList[client])

    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"Seconds to add records {elapsed_time}.")
```

```python
if __name__ == "__main__":
    nameAndDate()
    file_path = 'ClientData.csv'
    client_list = read_clients_from_csv(file_path)

    bst = BinarySearchTree()

    add_clients(client_list, bst)
```

The add_clients() method loops through the Client objects and inserts them into binarySearch.

In the main method we call read_clients_from_csv() method which we have used previously. Next, we create an instance of BinarySearchTree called bst. Then we call add_clients() with parameters client_list and bst.

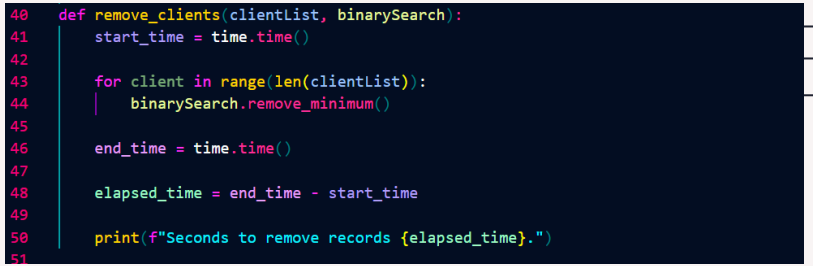
Add Clients Output



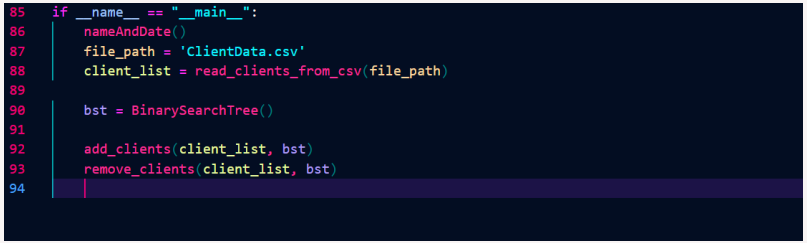
Remove Clients from Binary Search Tree

The remove_clients() method loops through the Client objects and removes them from the binarySearch using remove_minimum().

In the main method we call remove_clients() with parameters client_list and bst.



```
40  def remove_clients(clientList, binarySearch):
41      start_time = time.time()
42
43      for client in range(len(clientList)):
44          binarySearch.remove_minimum()
45
46      end_time = time.time()
47
48      elapsed_time = end_time - start_time
49
50      print(f"Seconds to remove records {elapsed_time}.")
51
```



```
85  if __name__ == "__main__":
86      nameAndDate()
87      file_path = 'ClientData.csv'
88      client_list = read_clients_from_csv(file_path)
89
90      bst = BinarySearchTree()
91
92      add_clients(client_list, bst)
93      remove_clients(client_list, bst)
94
```

Remove Clients Output

```
PS E:\CEIS295\Week 6 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/pyth
/BinarySearchTreeListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Seconds to add records 0.02912616729736328.
Seconds to remove records 0.007517099380493164.
PS E:\CEIS295\Week 6 Project>
```

Display Random Clients from Binary Search Tree

```python
def display_random_clients(clientList, binarySearch):
    start_time = time.time()

    for client in range(1000):
        small_id = 100001
        large_id = small_id + len(clientList)
        rand = random.randint(small_id, large_id)
        print(binarySearch.search(Client(rand)))

    end_time = time.time()

    elapsed_time = end_time - start_time

    print(f"Seconds to display 1000 random records {elapsed_time}.")
```

```python
def add_remove_display_clients(clientList, binarySearch):

if __name__ == "__main__":
    nameAndDate()
    file_path = 'ClientData.csv'
    client_list = read_clients_from_csv(file_path)

    bst = BinarySearchTree()

    add_clients(client_list, bst)
    display_random_clients(client_list, bst)
```

The display_random_clients() method loops through a range of Client objects, declares a random range and then searches for the random variable in binarySearch using search().

In the main method we call display_random_clients() with parameters client_list and bst.

Display Random Clients Output

```
PS E:\CEIS295\Week 6 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311
/BinarySearchTreeListActualSpeed.py"
Name: Faith Burnett
Date: April 20, 2024

Seconds to add records 0.026024341583251953.
[109199]
[100636]
[104199]
[102031]
[100984]
[103830]
[107573]
[103323]
[100746]
[101833]
[104864]
[105047]
[109019]
[108090]
[105241]
[102545]
[106783]
[106252]
[108906]
[104017]
[105042]
[108568]
[101500]
Seconds to display 1000 random records 0.1153721809387207.
PS E:\CEIS295\Week 6 Project>
```

Add, Display Random, Remove Random Clients

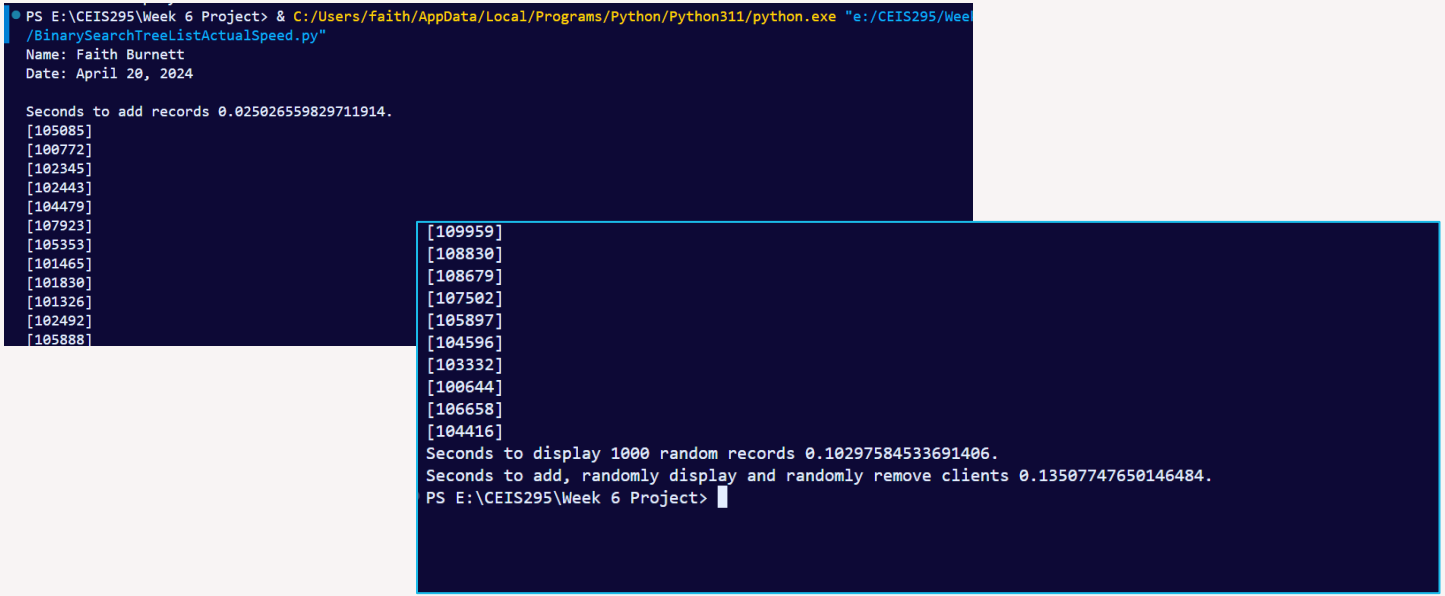
The add_remove_display_clients() method calls add_clients() and display_random_clients(). It also loops through a range of Client objects, declares a random variable and then uses the binarySearch method remove().

In the main method we call display_random_clients() with parameters client_list and bst.

```
67  def add_remove_display_clients(clientList, binarySearch):
68      start_time = time.time()
69
70      add_clients(clientList, binarySearch)
71      display_random_clients(clientList, binarySearch)
72
73      for client in range(1000):
74          small_id = 100001
75          large_id = small_id + len(clientList)
76          rand = random.randint(small_id, large_id)
77          binarySearch.remove(Client(rand))
78
79      end_time = time.time()
80
81      elapsed_time = end_time - start_time
82
83      print(f"Seconds to add, randomly display and randomly remove clients {elapsed_time}.")
84
```

```
85  if __name__ == "__main__":
86      nameAndDate()
87      file_path = 'ClientData.csv'
88      client_list = read_clients_from_csv(file_path)
89
90      bst = BinarySearchTree()
91
92      add_remove_display_clients(client_list, bst)
93
```

Add, Display Random, Remove Random Clients Output



Binary Search Tree Real World Speeds

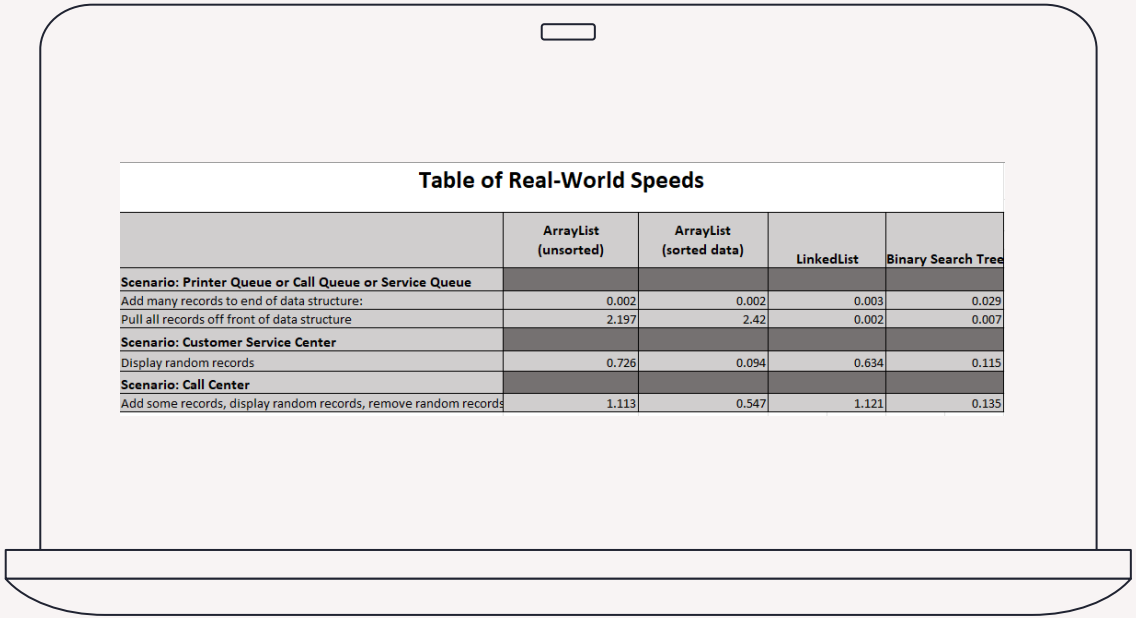


Table of Real-World Speeds

	ArrayList (unsorted)	ArrayList (sorted data)	LinkedList	Binary Search Tree
Scenario: Printer Queue or Call Queue or Service Queue				
Add many records to end of data structure:	0.002	0.002	0.003	0.029
Pull all records off front of data structure	2.197	2.42	0.002	0.007
Scenario: Customer Service Center				
Display random records	0.726	0.094	0.634	0.115
Scenario: Call Center				
Add some records, display random records, remove random records	1.113	0.547	1.121	0.135

Final Analysis

- Out of all the data structures the fastest processing times were using the Binary Search Tree.
- When testing searching algorithms the Binary Search was considerably more efficient than a Linear Search.
- Using the sorting algorithms I found that the Merge Sort provide the fastest processing time.

Challenges

- My biggest challenge was trying to perform Merge Sort and Quicksort on paper so that I could understand how the code would work.

- Another challenge was the organization of my code and trying not to have repeated code.

Career Skills

- Developing algorithms made to perform operations on arrays, linked lists, stacks and queues
- Using sorting algorithms that find the min or max value, sort and display running time
- Using searching algorithms to search different data structures
- Developing algorithms to use with a tree data structures and graph data structures

Conclusion

In conclusion, this course has provided a comprehensive understanding of fundamental data structures and algorithms. The topics learned in this course have given me a better grasp on how to efficiently organize, store, retrieve and manipulate data.

THANKS!

Do you have any questions?

faithburnett@outlook.com
https://faithbrnttt.github.io/Portfolio/Portfolio/