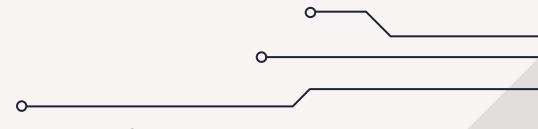


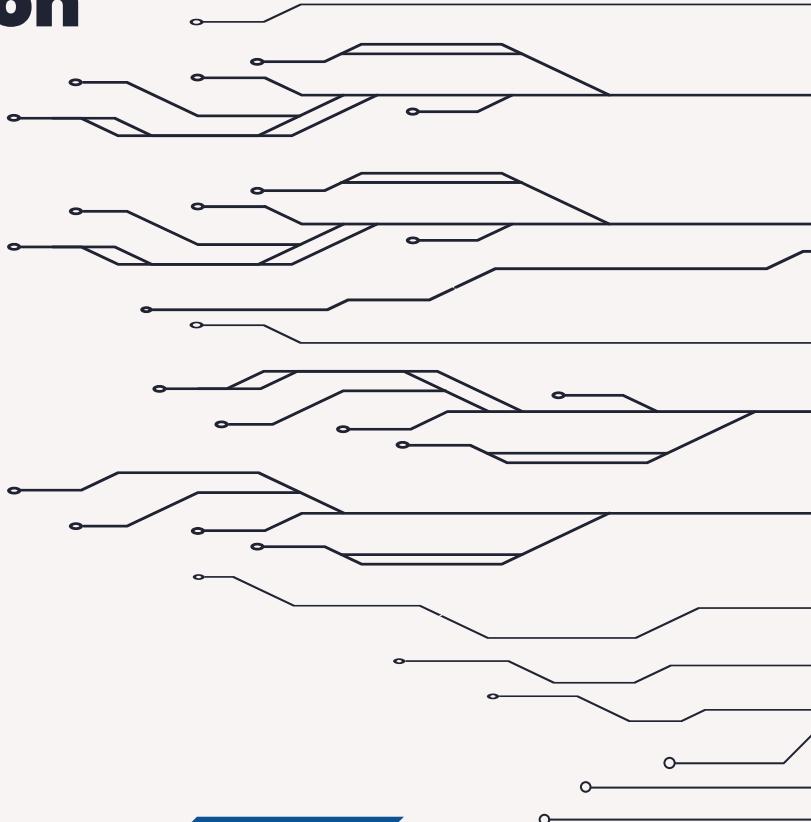
DATA STRUCTURES & ALGORITHMS

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 > ...
```

```
1  #Name: Faith Burnett
2  #Date: 04/11/2024
3  import time    # use time library to time the code executions
4  from datetime import date # use datetime library to display current date in output
5
6  # display name and date in output
7  def nameAndDate():
8      baseDate = date.today()
9      today = baseDate.strftime('%B %d, %Y')
10     print(f"Name: {name} \nDate: {today}\n")
11
```

```
12    # output phrase 1000 times
13    def phraseOutputSpeed():
14        # get current time before the process
15        start_time = time.time()
16
17        # run the process
18        for i in range(1000):
19            print("Hello Everyone!")
20
21        # get current time after the process
22        end_time = time.time()
23
24        # subtract start time from end time to get time used by process
25        elapsed_time = end_time - start_time
26
27        # Show the result. Note: .6f means "show six decimal places"
28        print(f"Seconds to run phrase 1000 times: {elapsed_time}")
29
30    if __name__ == "__main__":
31        nameAndDate()
32        phraseOutputSpeed()
```

Client Class

```
Client.py > Client
  1 #Name: Faith Burnett
  2 #Date: 04/17/2024
  3
  4 class Client:
  5
  6     #Initialize variables
  7     def __init__(self, client_id=0, first_name="Unknown", last_name="Unknown", phone="Unknown", email="Unknown"):
  8         self.__client_id = client_id
  9         self.__first_name = first_name
 10        self.__last_name = last_name
 11        self.__phone = phone
 12        self.__email = email
 13
 14     #Less than method for comparing instances of Client
 15     def __lt__(self, other):
 16         return self.__client_id < other.__client_id
 17
 18     #Equals method for comparing instances of Client
 19     def __eq__(self, other):
 20         return self.__client_id == other.__client_id
 21
 22     #String method displays the preferred data from Client
 23     def __str__(self):
 24         return str(self.__client_id) + ", " + self.__last_name + ", " + self.__first_name
 25
 26     #Define getters and setters
 27     def get_client_id(self):
 28         return self.__client_id
 29
 30     def set_client_id(self, client_id):
 31         self.__client_id = client_id
 32
 33     def get_first_name(self):
 34         return self.__first_name
 35
 36     def set_first_name(self, first_name):
 37         self.__first_name = first_name
 38
 39     def get_last_name(self):
 40         return self.__last_name
 41
 42     def set_last_name(self, last_name):
 43         self.__last_name = last_name
 44
 45     def get_phone(self):
 46         return self.__phone
 47
 48     def set_phone(self, phone):
 49         self.__phone = phone
 50
 51     def get_email(self):
 52         return self.__email
 53
 54     def set_email(self, email):
 55         self.__email = email
```

Name and Date Method

```
ArrayListActualSpeed.py > ...
1 #Name: Faith Burnett
2 #Date: 04/17/2024
3
4 from ArrayList import ArrayList
5 from Client import Client
6 from datetime import date
7 import time, random
8
9 def nameAndDate():
10     baseDate = date.today()
11     today = baseDate.strftime('%B %d, %Y')
12     print(f"Name: Faith Burnett\nDate: {today}\n")
13
107 if __name__ == "__main__":
108     nameAndDate()
109
```

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.

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.

```
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
```

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

Add Clients to ArrayList

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

```
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
```

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.exe  
/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")
```

```
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)
```

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

```
55 def getRandomRecords(clientList, clientArray):
56
57     start_time = time.time()
58
59     for i in range(1000):
60         min_id = 100001
61         max_id = min_id + len(clientList)
62         r_num = random.randint(min_id, max_id)
63         print(clientArray.search(Client(r_num)))
64
65     end_time = time.time()
66
67     elapsed_time = end_time - start_time
68
69     print(f"\nTime taken to get a random number of Client objects from ArrayList:{\n{elapsed_time} seconds")
```

```
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     getRandomRecords(client_list, client_array)
118
```

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  
189902, Church, Isadora  
183064, Lowery, Plato  
182740, O'neil, Chiquita  
105876, Roberts, Freya  
100911, Hart, Zephania  
109248, Myers, Brock  
108770, Joyner, Virginia  
182749, Finley, Jaquelyn  
109372, Moss, Victor  
108405, Aguirre, Eleanor  
100298, Sandoval, Blaze  
103591, Parks, Miriam  
108319, Zamora, Lillian
```

```
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.

```
71 | def addDisplayRemoveSpeed(clientList, clientArray):
72 |
73 |     r_num = random.randint(0, 100001
74 |     start_time = time.time()
75 |
76 |     print("Adding clients...\n")
77 |
78 |     #add client
79 |     for i in range(5000:
80 |         clientArray.append(clientList[i])
81 |
82 |     print("Displaying random clients...\n")
83 |
84 |     #display random clients
85 |     for r_num in range(1000:
86 |         min_id = 100001
87 |         max_id = min_id + len(clientList)
88 |         r_num = random.randint(min_id, max_id)
89 |         print(clientArray.search(Client(r_num)))
90 |
91 |     print("\n")
92 |
93 |     #remove random clients
94 |     for r_num in range(1000:
95 |         clientArray.remove_at(r_num)
96 |
97 |         print("Removing random clients...\n")
98 |
99 |         end_time = time.time()
100 |
101 |         elapsed_time = end_time - start_time
102 |
103 |         print(f"Time taken to add, display and delete client objects in ArrayList:{elapsed_time} seconds\n")
104 |
105 |
106 |
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 |     addDisplayRemoveSpeed(client_list, client_array)
117 |
```

Add, Display Random, Remove Random Clients Output

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faithe/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
68     end_time = time.time()
69
70     elapsed_time = end_time - start_time
71
72     print(f"\nTime taken to get a random number of Client objects from ArrayList:{elapsed_time} seconds")
```

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> []
```

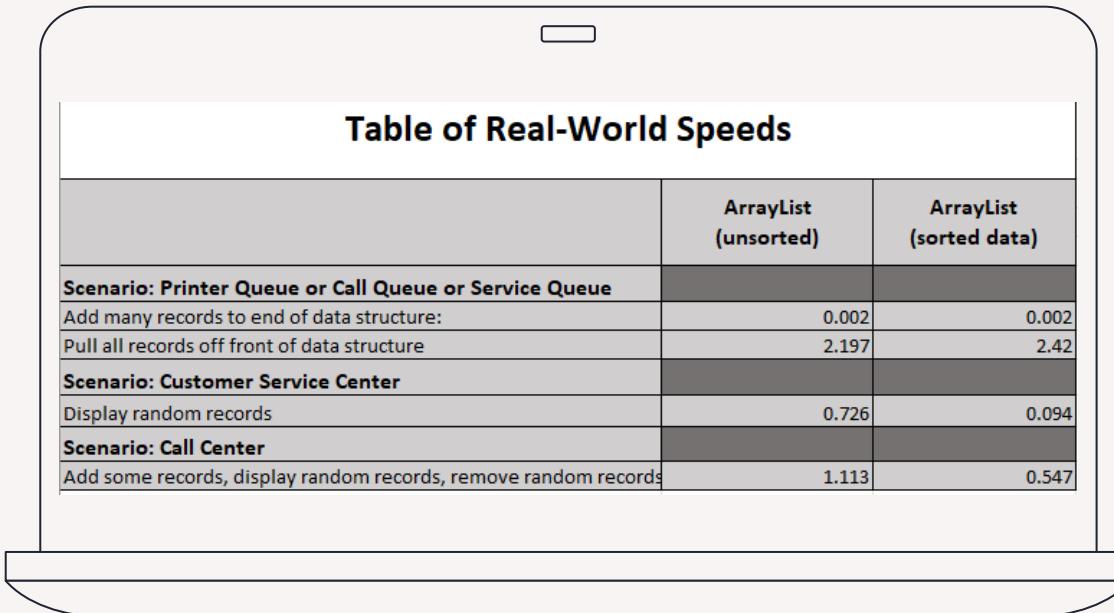
Display Random Clients Quicksort Output

```
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

```
PS E:\CEIS295\Week 1 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.  
/ArrayListActualSpeed.py  
Name: Faith Burnett  
Date: April 20, 2024  
  
Adding clients...  
Displaying random clients...  
106336, Marsh, Barry  
105204, Park, Cassidy  
108082, Riley, Ruby  
109493, Graves, Karyn  
106696, Berry, Joelle  
105903, Mcdonald, Dominic  
100267, Walton, Jolene  
108329, Norton, Leila  
102471, Wright, Fitzgerald  
103430, Lyons, Aladdin  
101434, Campbell, Theodore  
103595, Galloway, Taylor  
105288, Keith, Karly  
105858, Weeks, Carter  
101119, Sweeney, Kasimir  
108778, Holt, Mariam  
101475, Holloway, Ursula  
106464, Carney, Quinlan  
102355, Hartman, Reagan  
108192, Donaldson, Kelsie  
  
Removing random clients...  
Time taken to add, display and delete client objects in ArrayList:  
0.5469980239868164 seconds
```

ArrayList Real World Speeds

A white smartphone icon with a black outline is centered on the slide. It has a small rectangular notch at the top center and a curved bottom edge. The screen displays a table titled "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     nameAndDate()
96
97     client_list = read_clients_to_file('ClientData.csv')
98     linked_list = LinkedList()
99
100    addClients(client_list, linked_list)
101
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: 0.003000974655151367 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 its 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 28, 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.

```
67 def addDisplayRemove(clientList, linkedlist):
68
69     start_time = time.time()
70
71     for i in range(len(clientList)):
72         linkedList.add_last(clientList[i])
73
74     print "Adding clients...\n"
75
76     print "Displaying clients...\n"
77     for i in range(1000:
78         small_id = 100001
79         large_id = small_id + len(clientList)
80         random_num = random.randint(small_id, large_id)
81         print(linkedList.search(Client(random_num)))
82
83     print "\nRemoving clients..."
84     for i in range(1000:
85         small_id = 100001
86         large_id = small_id + len(clientList)
87         random_num = random.randint(small_id, large_id)
88         linkedList.remove(Client(random_num))
89
90
91     end_time = time.time()
92
93     elapsed_time = end_time - start_time
94
95     print f"Time taken to get clients added, grab random clients and remove random clients: {elapsed_time} seconds\n"
96
97
98 if __name__ == "__main__":
99
100     nameAndDate()
101
102     client_list = read_clients_to_file('ClientData.csv')
103     linked_list = LinkedList()
104
105     addDisplayRemove(client_list, linked_list)
```

Add, Display Random, Remove Random Clients Output

```
PS E:\CEIS295\Week 2 Project> & C:/Users/faithe/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

```
27 def event1(callList, callsWaiting, callNumber):
28     print("Call received. Caller added to queue.")
29     rand = random.randint(0, len(callList))
30     callsWaiting.enqueue(callList[rand])
31     callNumber += 1
32     print("\tNumber of calls waiting in queue:", callsWaiting.get_length())
33
34
35 def event2(callsWaiting):
36     print("Call sent to representative for service.")
37     if callsWaiting.get_length() > 0:
38         print("Caller information:")
39         print(callsWaiting.dequeue())
40     else:
41         print("The call waiting queue is empty.")
42         print("\tNumber of calls waiting in queue:", callsWaiting.get_length())
43
44 if __name__ == "__main__":
45     nameAndDate()
46     call_list = read_clients_from_file('CallsData.csv')
47     calls_waiting = Queue()
48     call_number = 0
49
50     seconds = int(input("How many seconds do you want to simulate? "))
51
52     for i in range(seconds):
53         print(".....")
54         time.sleep(2)
55         rand_num = random.randint(1, 3)
56
57         if rand_num == 1:
58             event1(call_list, calls_waiting, call_number)
59
60         elif rand_num == 2:
61             event2(calls_waiting)
62
63         else:
64             print("Nothing happened during this second in time.")
65             print("\tNumber of calls waiting in queue:", calls_waiting.get_length())
66
67     print("\n\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 `event2()` depending on the contents of the Queue.

Call Distributor Output

```
PS E:\CEIS295\Week 3 Project> & C:/Users/Faith/AppData/Local/Programs/Python/Python311/python.exe "e:/CEIS295/Week 3 Project/AutomaticCallDistributor.py"
Name: Faith Burnett
Date: April 20, 2024

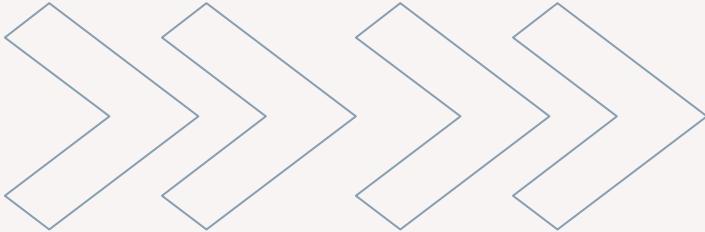
How many seconds do you want to simulate? 30
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 1
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 2
.....
Call sent to representative for service.
Caller information:
528634, Rashad Blevins, (581) 239-1916
Date/Time: 04/20/2024 @ 10:48
.....
Nothing happened during this second in time.
    Number of calls waiting in queue: 1
.....
Call sent to representative for service.
Caller information:
505225, Jaime Thompson, (529) 166-8430
Date/Time: 04/20/2024 @ 10:48
.....
Call sent to representative for service.
Caller information:
217109, Basia Carlson, (342) 369-5700
Date/Time: 04/20/2024 @ 10:48
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 1
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 2
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 3
```

```
Call sent to representative for service.
Caller information:
528634, Rashad Blevins, (581) 239-1916
Date/Time: 04/20/2024 @ 10:48
.....
Nothing happened during this second in time.
    Number of calls waiting in queue: 1
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 2
.....
Nothing happened during this second in time.
    Number of calls waiting in queue: 1
.....
Call received. Caller added to queue.
    Number of calls waiting in queue: 2
.....
Call sent to representative for service.
Caller information:
357764, Carla Cline, (921) 231-0669
Date/Time: 04/20/2024 @ 10:48
.....
Call sent to representative for service.
Caller information:
723115, Uriel Whitley, (320) 694-9001
Date/Time: 04/20/2024 @ 10:48
.....
The 'Automatic Call Distributor' simulation has completed.
```

```
PS E:\CEIS295\Week 3 Project> [
```



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

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

Sorting 100 records...
Seconds to sort 100 records 0.0019991397857666016

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
Davison, Neville
```

Bubble Sort 1000

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

Sorting 1000 records...
Seconds to sort 1000 records 0.21036815643310547

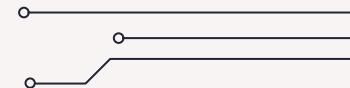
Abbott, Larissa
Acevedo, Baxter
Acosta, Dominic
Acosta, Paki
Adkins, Brenna
Aguilar, Wing
Aguirre, Cassidy
Aguirre, Leo
Albert, India
Allison, Jeremy
Alvaredo, Channing
Alvarez, Amal
Alvarez, Lacey
Alvarez, Then
```

Bubble Sort 10000

```
PS E:\CEIS295\Week 4 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python31
/SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

Sorting 10000 records...
Seconds to sort 10000 records 21.126347541809082

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



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

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

Sorting 100 records...
Seconds to sort 100 records 0.0009987354278564453

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
Deacon, Dequan
```

Selection Sort 1000

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

Sorting 1000 records...
Seconds to sort 1000 records 0.15776777267456055

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

Selection Sort 10000

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

Sorting 10000 records...
Seconds to sort 10000 records 14.931624174118042

○ PS E:\CEIS295\Week 4 Project> □
```

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)
58
```

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
Acededo, 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)
```

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, Alvara
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

Shell 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 0.005997657775878906

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 "E:/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
Acosta, Catherine
```

Shell Sort 100000

```
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 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

Quicksort 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 0.005135774612426758

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 100000

```
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 0.06612110137939453

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

```
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 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/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
```

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/python.exe
./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/python.exe
./SortingActualSpeeds.py"
Name: Faith Burnett
Date: April 20, 2024

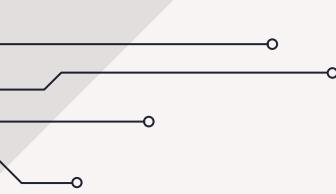
Sorting 100000 records...
Seconds to sort 100000 records 0.8521134853363037

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

Sorting Real World Speeds

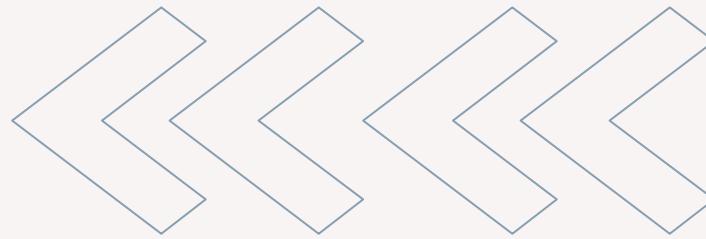
Table of Real-World Sorting Speeds

	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

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

Searching 1000 random records...
100098, Simon, Gillian
100006, Dotson, Daquan
100079, Vaughn, Ariel
100004, Turner, Gretchen
100066, Townsend, Wallace
100023, Guy, Quemby
100089, Price, Callie
100004, Turner, Gretchen
100019, Reeves, Jenna
100085, Dunn, Chase
100089, Price, Callie
100059, Holloway, Fatima
100061, Ferrell, Abel
100074, Hardin, Cheyenne
100014, Ramsey, Howard
```

```
100013, Farley, Carson
100032, Patterson, Ebony
100091, Stuart, Quentin
100024, Dunlap, Kalia
100077, Hewitt, Burton
100088, Lynch, Savannah
100074, Hardin, Cheyenne
100021, Wright, Phyllis
100094, Sullivan, Odette
100059, Holloway, Fatima
100036, Callahan, Amir
100057, Medina, Baxter
100058, Blevins, Anne
Seconds to search 1000 records 0.09638047218322754
```

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

Linear Search 1000 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...
100360, Kirkland, Britanni
100943, Garcia, Cairo
100062, Trevino, Athena
100783, Goodwin, Ulric
100553, David, Devin
100384, Potts, Beau
100069, Mcintyre, Martena
100034, Dominguez, Ignacia
100078, Pollard, Ciara
100366, Whitfield, Brittany
100199, Page, Hashim
100695, Duffy, Ian
100707, Maynard, Farrah
100455, Macias, Orlando
100773, Wallace, Andi
```

```
100147, Bray, Isaac
100026, Hogan, Marcia
100027, Whitney, Grady
100924, Salinas, Emery
100436, Barrera, Brendan
100788, Riddle, Richard
100300, Johns, Burke
100625, Hahn, Martha
100704, Stout, Rosalyn
100324, Dickerson, Briar
100988, Lynn, Virginia
Seconds to search 1000 records 0.1535642147064209
```

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

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
109668, Fletcher, Yeo
105284, Marks, Driscoll
104182, Cash, Hashim
109612, Witt, Aubrey
100356, Goad, Pamela
```

```
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

```
PS E:\CEIS295\Week 5 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python.exe  
/SearchingActualSpeeds.py  
Name: Faith Burnett  
Date: April 20, 2024  
  
Searching 1000 random records...  
176349, Chang, Medge  
186648, Mack, Indigo  
146797, Harrell, Gemma  
142980, Cox, Anne  
162469, Dudley, Henry  
141434, Raymond, Kay  
183855, Craft, Jana  
166166, Hansen, Zahir  
163869, Dodson, Eaton  
165802, Salinas, Gray  
146956, Medina, Yasir  
114466, Hood, Kevyn  
156469, Coffey, Henry  
182349, Salinas, Jelani  
118416, Keller, Laith  
128696, Murray, Karen  
171454, Keller, Rachel
```

```
115831, Parrish, Zelenia  
123906, Abbott, Yetta  
179811, Vincent, Nell  
181821, Byers, Amaya  
158494, Witt, Beatrice  
151615, Harding, Jolie  
127852, Calderon, Candice  
115867, Harmon, McKenzie  
190282, Dominguez, Xantha  
146035, Bright, Chelsea  
129821, McLaughlin, Colin  
104488, Herman, Venus  
137570, Chaney, Melanie  
Seconds to search 1000 records 7.9981818199157715
```



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

```
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...
100088, Ferrell, Price
100073, Lowery, Jasper
100066, Townsend, Wallace
100056, Brown, Cain
100066, Townsend, Wallace
100058, Blevins, Anne
100008, Woodard, Amena
100026, Roth, Malachi
100025, Cochran, Claudia
100065, Wells, Eugenia
100022, Burnett, Gannon
100077, Hewitt, Burton
100002, Barnes, Jeremy
100081, George, Cameron
```

```
100018, Stein, Luke
100024, Dunlap, Kalia
100095, Holmes, Camilla
100044, Knight, Gavin
100004, Turner, Gretchen
100080, Lynch, Savannah
100003, Anderson, Alfreda
100043, Park, Georgia
100035, Gentry, Cadman
100002, Barnes, Jeremy
100097, Lowery, Wing
100024, Dunlap, Kalia
Seconds to search 1000 records 0.09710574150085449
```

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

Binary Search 1000 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...
100496, Travis, Brittany
100971, Espinoza, Shelley
100298, Pace, Denise
100016, Stout, Solomon
100114, Lindsey, Rinah
100231, Sloan, Aiden
100324, Dickerson, Briar
100446, Barrett, Devin
100624, Jordan, Jessica
100667, Preston, Christopher
100328, Howard, Raymond
100546, Hardin, Brynn
100083, Cannon, Chloe
```

```
100001, Ferguson, Barrett
100595, Gibson, Belle
100943, Garcia, Cairo
100476, Burks, Kenneth
100441, Wooten, Aiden
100282, Raymond, Allen
100057, Norman, Jayme
100735, Bradford, Arthur
100900, Moses, Jonas
100174, Tran, Madonna
100068, Tanner, India
100438, English, Mechelle
100643, Hill, MacKensie
Seconds to search 1000 records 0.09830069541931152
```

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

Binary Search 10000 Output

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

Searching 1000 random records...
108921, Payne, Peter
105021, Stewart, Wylie
109367, Williamson, Rajah
100313, Crosby, Cassidy
109469, Durham, Jasmine
103772, Mathews, Dean
103056, Baldwin, Asher
101786, Williams, Anthony
106334, Chaney, Desiree
107976, Callahan, Clementine
104828, Christensen, Mason
108029, Mcneil, Shaine
101739, Mosley, Yolanda
106783, Mccoy, Logan
100097, Deleon, Drew
100466, Fleming, George
```

```
107628, Cleveland, Herrod
102141, Tran, Shannon
102729, Jimenez, Donna
108173, Witt, Karina
100477, Clarke, Brandon
102736, Myers, Lareina
109376, Valdez, Victor
108694, Atkinson, Lamar
103689, Grimes, Rigel
105998, Mayer, Berk
104247, Hebert, Martina
107777, Spencer, Zorita
101698, Nixon, Quinn
101303, Poole, Stone
Seconds to search 1000 records 0.09885072708129883
```

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

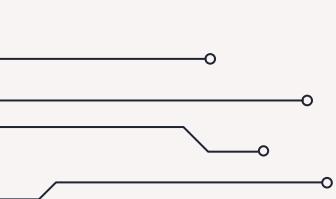
Binary Search 100000 Output

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

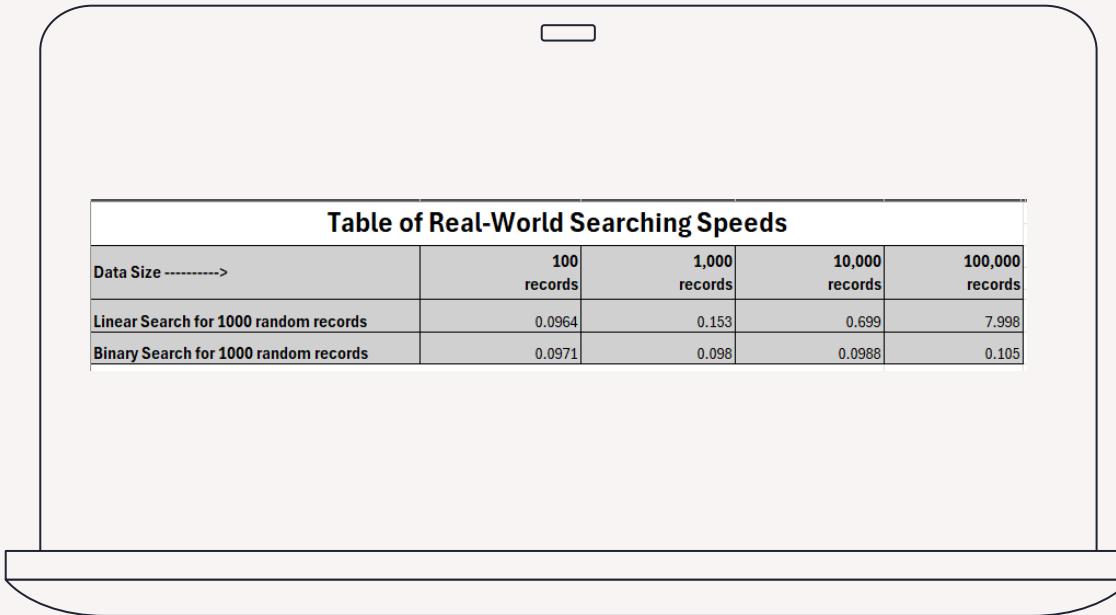
Searching 1000 random records...
171518, Irwin, Hakeem
112759, Vang, Coby
181223, Pierce, Josiah
165171, Vincent, Vance
166348, Blackburn, Tyler
130321, Harris, Micah
139095, Koch, Dustin
146099, Joyner, Gareth
108925, Maddox, Brynne
185049, Little, Patrick
180137, McCarthy, Aquila
118149, Vance, Quail
111028, Sharpe, Grant
156317, Weeks, Vivien
149421, Barker, Felix
131458, Kent, Tanek
127114, Joseph, Amy
144746, McCarthy, Ursula
106661, Park, Mohamed
```

```
165846, Brock, Channing
100264, Kane, Adele
199345, Gibson, Yuli
111059, Frost, Scott
119080, Shepard, Iona
122281, Tillman, Sebastian
184209, Wright, Eve
169183, Simpson, Wallace
125696, Bowman, Fiona
178682, House, Norman
115227, Cardenas, Aimee
143949, Suarez, Devin
120593, Forbes, Keelie
102783, Becker, Jana
Seconds to search 1000 records 0.10499095916748047

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



Searching Real World 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

```
28 def add_clients(clientList, binarySearch):
29     start_time = time.time()
30
31     for client in range(len(clientList)):
32         binarySearch.insert(clientList[client])
33
34     end_time = time.time()
35
36     elapsed_time = end_time - start_time
37
38     print(f"Seconds to add records {elapsed_time}.")
39 
```

```
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 
```

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

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

Seconds to add records 0.02770709991455078.
○ PS E:\CEIS295\Week 6 Project>
```

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}.")
```

```
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     |
```

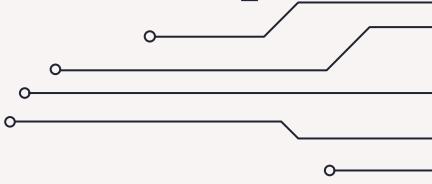
Remove Clients Output

```
● PS E:\CEIS295\Week 6 Project> & C:/Users/faith/AppData/Local/Programs/Python/Python311/python /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

```
52 def display_random_clients(clientList, binarySearch):
53     start_time = time.time()
54
55     for client in range(1000):
56         small_id = 100001
57         large_id = small_id + len(clientList)
58         rand = random.randint(small_id, large_id)
59         print(binarySearch.search(Client(rand)))
60
61     end_time = time.time()
62
63     elapsed_time = end_time - start_time
64
65     print(f"Seconds to display 1000 random records {elapsed_time}.")
66
67 > def add_remove_display_clients(clientList, binarySearch): ...
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84     if __name__ == "__main__":
85         nameAndDate()
86         file_path = 'ClientData.csv'
87         client_list = read_clients_from_csv(file_path)
88
89         bst = BinarySearchTree()
90
91         add_clients(client_list, bst)
92         display_random_clients(client_list, bst)
93
```



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] [104864]  
[100636] [105047]  
[104199] [109019]  
[102031] [108090]  
[100984] [105241]  
[103830] [102545]  
[107573] [103323]  
[106783] [100746]  
[106252] [101833]  
[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}.")

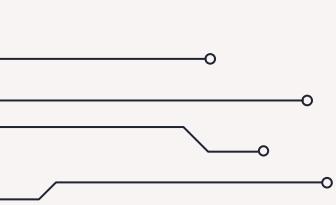
84
```

```
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
```

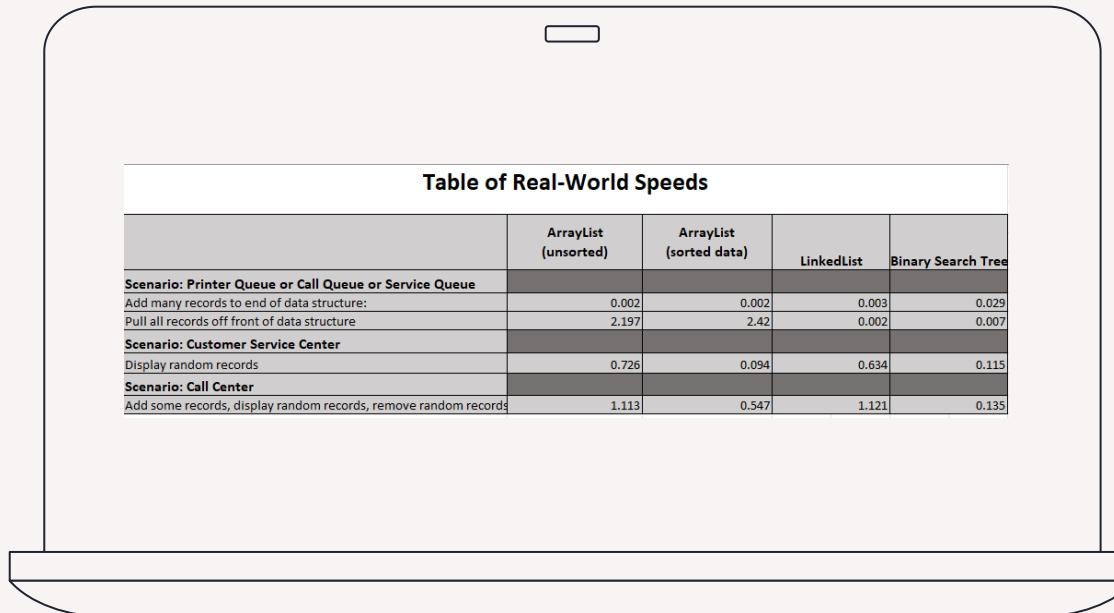
Add, Display Random, Remove Random Clients Output

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

Seconds to add records 0.025026559829711914.
[105085]
[100772]
[102345]
[102443]
[104479]
[107923]
[105353]
[101465]
[101830]
[101326]
[102492]
[105888]
[109959]
[108830]
[108679]
[107502]
[105897]
[104596]
[103332]
[100644]
[106658]
[104416]
Seconds to display 1000 random records 0.10297584533691406.
Seconds to add, randomly display and randomly remove clients 0.13507747650146484.
PS E:\CEIS295\Week 6 Project>
```



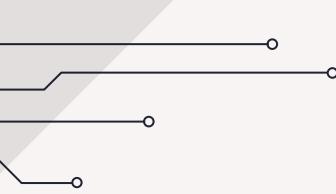
Binary Search Tree Real World Speeds



	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.

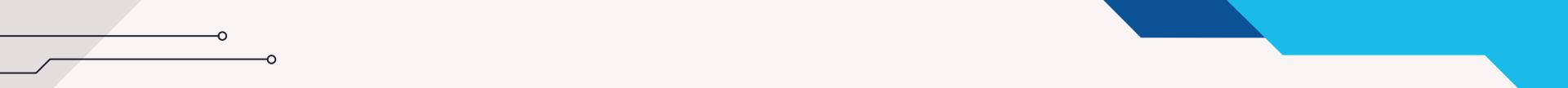


Challenges

- 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://faithbrntt.github.io/Portfolio/Portfolio/>



CREDITS: This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#), and infographics & images by [Freepik](#)