Danny Sullivan

CS 2223 – Algorithms

14 April 2017

Project 1

The first algorithm, a brute force algorithm, is a linear function. The function goes through the entire list and records the minimum and maximum value. The time efficiency of this method is O(n) in all cases, as no matter what, the function must run through the entire list. However, the best case could be O(1) only if the array consists of one element. The storage efficiency is constant for this, at O(n), as only two elements are stored. This brute force method runs as

```
1. Input: a non-empty array of integers
2. Set maximum and minimum equal to the first element in the array
3. If the length of the array is equal to 1
    a. Print “The minimum value is: ”, minimum
    b. Print “The maximum value is: ”, maximum
4. For x in range of 1 to array length
    a. If x element of the array is greater than the maximum
        i. The maximum is equal to the $x^{th}$ element of the array
    b. If x element of the array is less than the minimum
        i. The minimum is equal to the $x^{th}$ element of the array
5. Print “The minimum value is: ”, minimum
6. Print “The maximum value is: ”, maximum
```

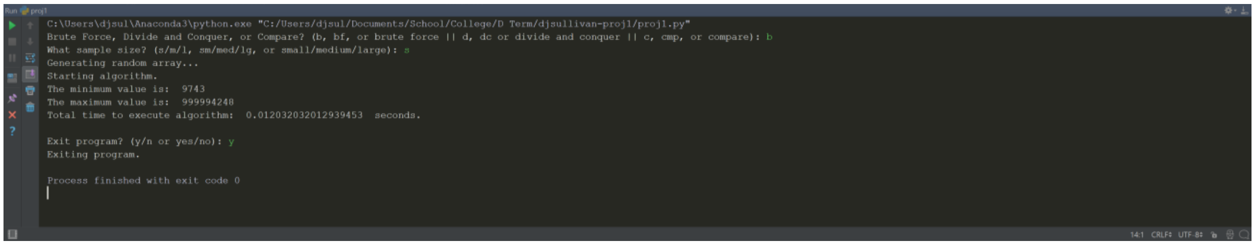
The other algorithm used was a divide and conquer algorithm. This algorithm divides the array by two until it is able to make a comparison between two values. This algorithm is consistent in time efficiency, where all the cases are the same. The best, average, and worst case are O(3n/2-2). The storage efficiency is the same, O(n), as there are 2 variables being changed and compared consistently throughout the algorithm. The divide and conquer method runs as

```
1. Input: array, beginning index (0), and the size of the array – 1
2. Define global minimum and maximum variables
3. If the size of the array-1 minus the index is greater than two
    a. Recursively call the function with the array, beginning index, and the
       beginning plus the end divided by two as parameters
    b. Recursively call the function with the array, beginning index plus the size of
       the array-1, and the size of the array-1 as the parameters
4. Else
    a. Splice the array between the beginning index and the size of the array-1
    b. If the size of the array-1 minus the index is equal to one
        i. Add index 0 back onto the array
    c. Set max equal to the number that is bigger between the first and second
       elements being compared
```

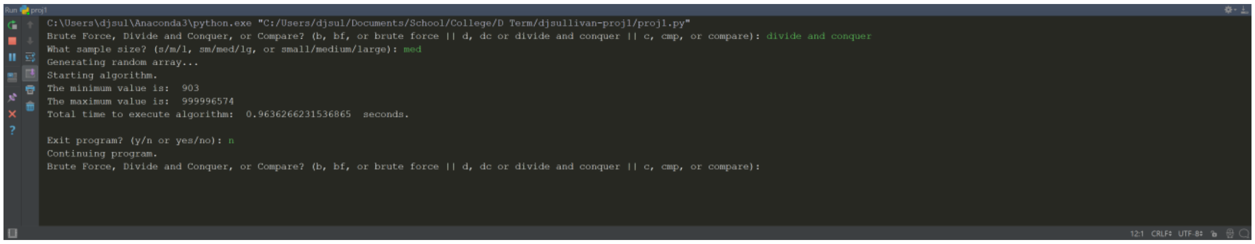
d. Set min equal to the number that is smaller between the first and second elements being compared

The data sets were created through a function called createArray(x), in which x is a variable that determines the number of elements in that array. An array is created and filled with random numbers between 1 and 999,999,999. This array is then passed into the algorithms to test them. When being compared, they are compared with the same random array so that they are compared properly.

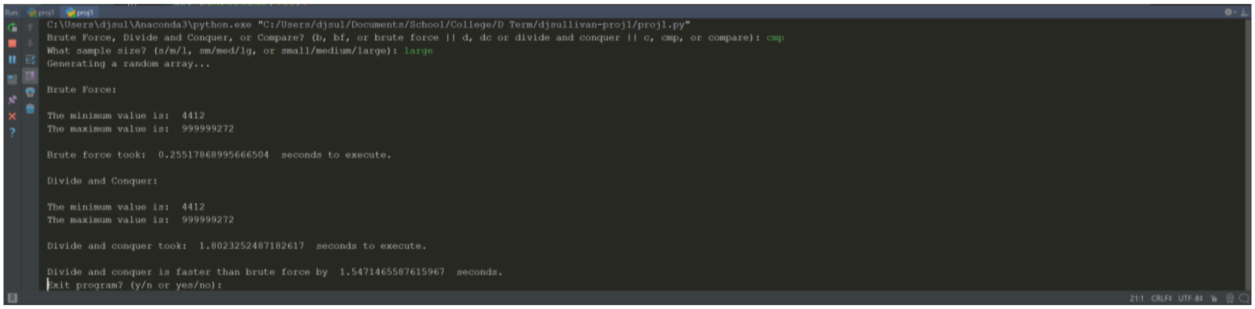
Screenshots (in IDE)



```
C:\Users\djsul\Anaconda3\python.exe "C:/Users/djsul/Documents/School/College/D Term/djsullivan-proj1/proj1.py"
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare): b
What sample size? (s/m/l, sm/med/lg, or small/medium/large): s
Generating random array...
Starting algorithm.
The minimum value is:  9743
The maximum value is:  999994248
Total time to execute algorithm:  0.012032032012939453  seconds.

Exit program? (y/n or yes/no): y
Exiting program.

Process finished with exit code 0
```

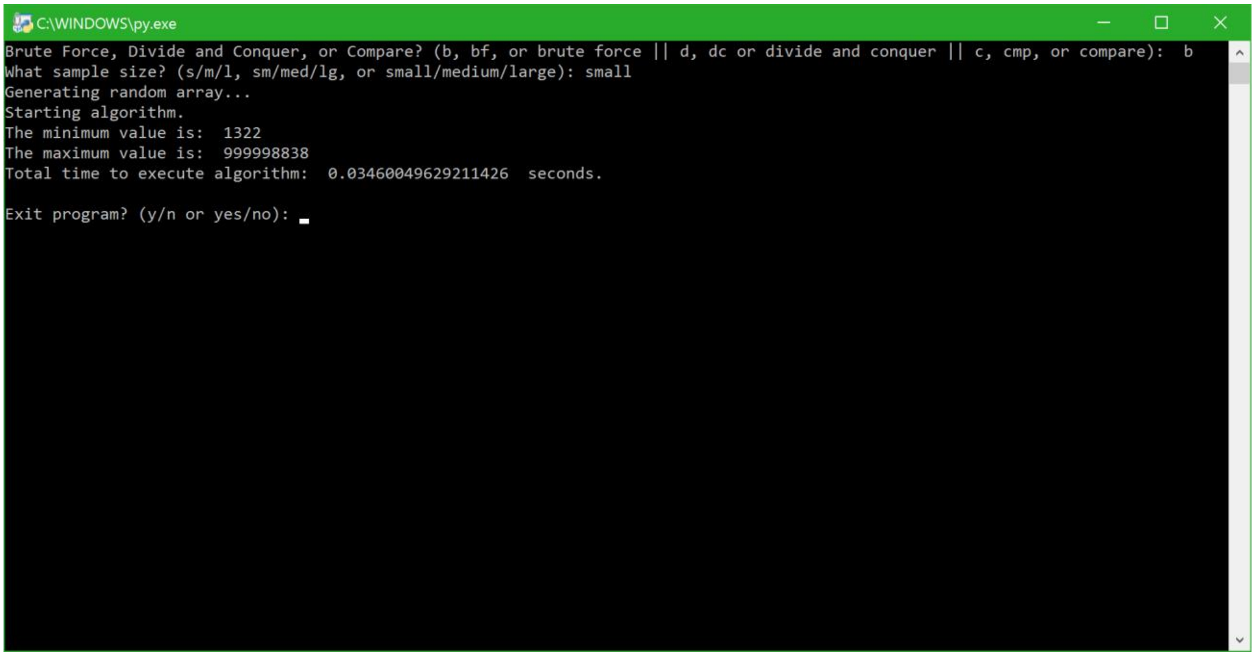


```
C:\Users\djsul\Anaconda3\python.exe "C:/Users/djsul/Documents/School/College/D Term/djsullivan-proj1/proj1.py"
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare): divide and conquer
What sample size? (s/m/l, sm/med/lg, or small/medium/large): med
Generating random array...
Starting algorithm.
The minimum value is:  903
The maximum value is:  999996574
Total time to execute algorithm:  0.9636266231536865  seconds.

Exit program? (y/n or yes/no): n
Continuing program.
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare):
```

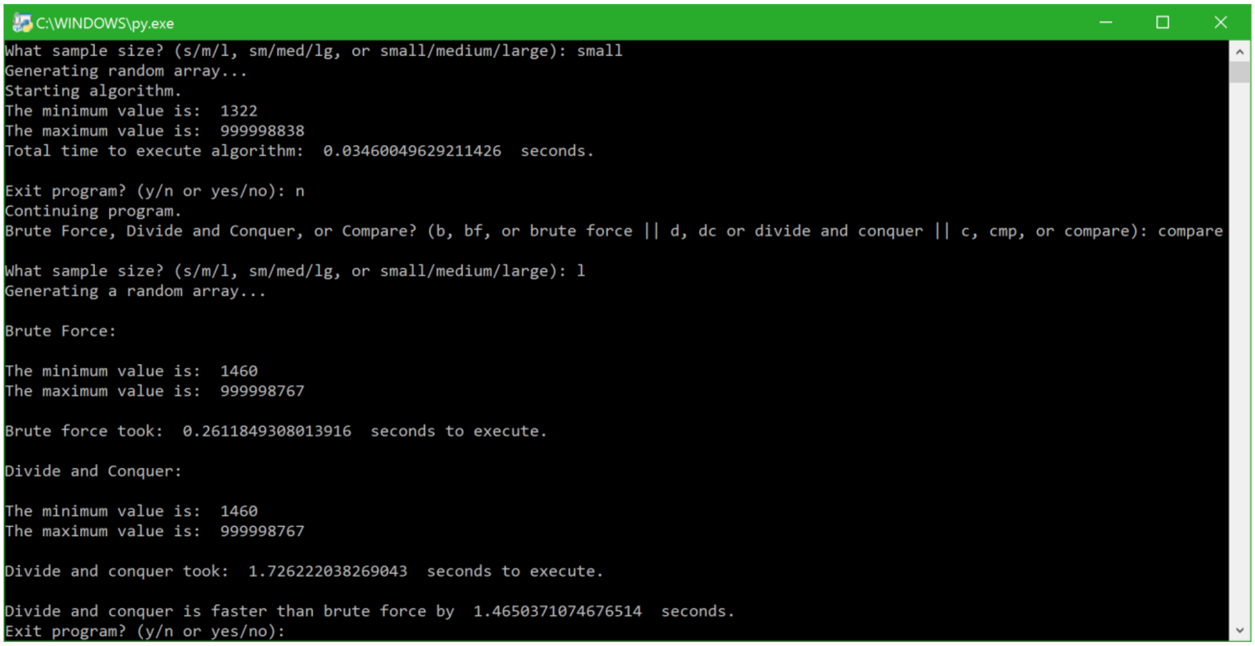


```
C:\Users\djsul\Anaconda3\python.exe "C:/Users/djsul/Documents/School/College/D Term/djsullivan-proj1/proj1.py"
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare): cmp
What sample size? (s/m/l, sm/med/lg, or small/medium/large): large
Generating a random array...

Brute Force:

The minimum value is:  4412
The maximum value is:  999999272

Brute force took:  0.25517868995666504  seconds to execute.

Divide and Conquer:

The minimum value is:  4412
The maximum value is:  999999272

Divide and conquer took:  1.8023252487182617  seconds to execute.

Divide and conquer is faster than brute force by  1.5471465587615967  seconds.
Exit program? (y/n or yes/no):
```

Screenshots (in command shell)

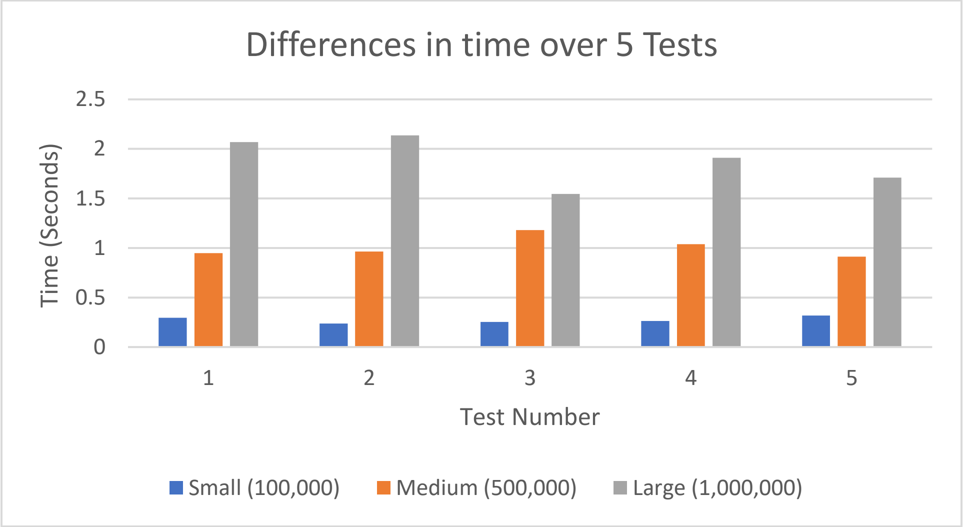


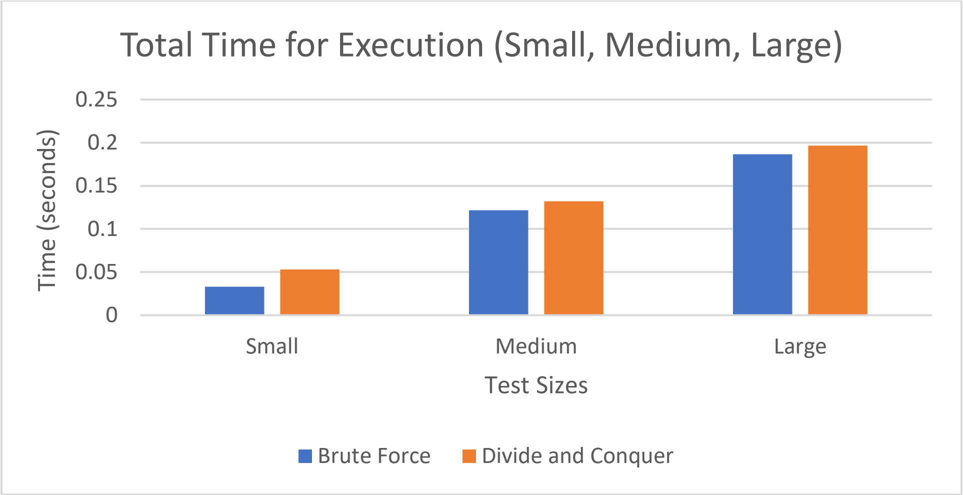
```
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare):  b
What sample size? (s/m/l, sm/med/lg, or small/medium/large): small
Generating random array...
Starting algorithm.
The minimum value is:  1322
The maximum value is:  999998838
Total time to execute algorithm:  0.03460049629211426  seconds.

Exit program? (y/n or yes/no): _
```

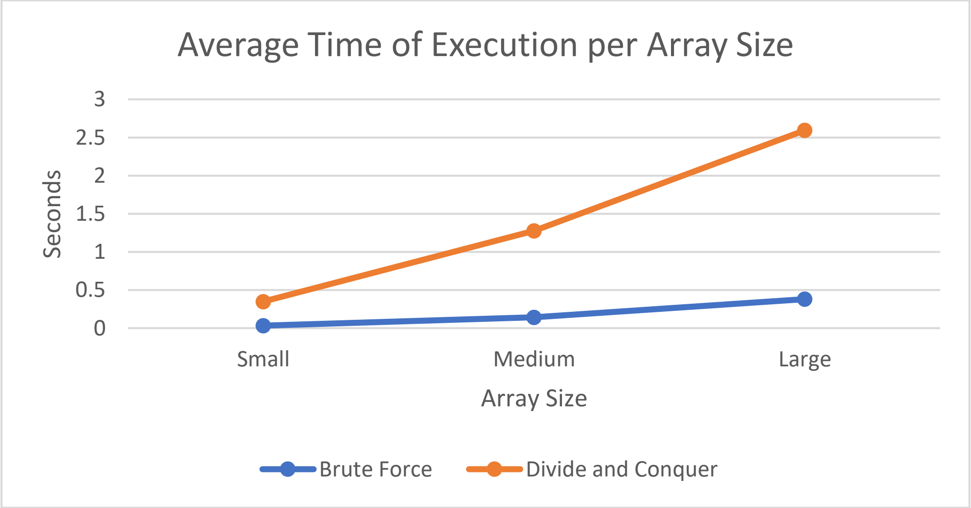


```
What sample size? (s/m/l, sm/med/lg, or small/medium/large): small
Generating random array...
Starting algorithm.
The minimum value is:  1322
The maximum value is:  999998838
Total time to execute algorithm:  0.03460049629211426  seconds.

Exit program? (y/n or yes/no): n
Continuing program.
Brute Force, Divide and Conquer, or Compare? (b, bf, or brute force || d, dc or divide and conquer || c, cmp, or compare): compare

What sample size? (s/m/l, sm/med/lg, or small/medium/large): l
Generating a random array...

Brute Force:

The minimum value is:  1460
The maximum value is:  999998767

Brute force took:  0.2611849308013916  seconds to execute.

Divide and Conquer:

The minimum value is:  1460
The maximum value is:  999998767

Divide and conquer took:  1.726222038269043  seconds to execute.

Divide and conquer is faster than brute force by  1.4650371074676514  seconds.
Exit program? (y/n or yes/no):
```

Graphs


Differences in time over 5 Tests
Time (Seconds)
2.5
2
1.5
1
0.5
0
1
2
3
4
5
Test Number
Small (100,000)
Medium (500,000)
Large (1,000,000)


Total Time for Execution (Small, Medium, Large)
Time (seconds)
0.25
0.2
0.15
0.1
0.05
0
Small
Medium
Large
Test Sizes
Brute Force
Divide and Conquer


Average Time of Execution per Array Size
Seconds
3
2.5
2
1.5
1
0.5
0
Small
Medium
Large
Array Size
Brute Force
Divide and Conquer