File - /home/dateraon/Desktop/CSC-120-Git/CSC-120/Lab06 starter code/main.py

```
11 def __list_generator(kind, length):
12     """
13     creates a <kind> list with <length> numbers
14     :param kind: data type like "random", "identical
   ", "reverse-sorted", "sorted"
15     :param length: number of numbers that should be
   placed into list
16     :return: the finished list
17     """
18     data = []
19     if kind == "random":
20         for i in range(length):
21             data.append(random.randint(1,length))
22     elif kind == "identical":
23         value = random.randint(1,length)
24         for i in range(length):
25             data.append(value)
26     elif kind == "sorted_list":
27         for count in range(length):
28             data.append(count)
29     elif kind == "reverse_sorted_list":
30         for count in range(length, 0, -1):
31             data.append(count)
32
33     return data
```

Sort 1

Kind of List Data	List Size	List References
Random	10	90
Random	100	1521
Random	200	3617
Random	500	9799
Random	1000	21963
Random	2000	46596
Random	2250	54647
Random	3000	78427
Random	5000	134554
Random	10000	283184
Random	20000	632694

Kind of List Data	List Size	List References
Identical	10	87
Identical	100	1511
Identical	200	3423
Identical	500	9351
Identical	1000	20703
Identical	2000	45407
Identical	2250	55023
Identical	3000	72511
Identical	5000	131167
Identical	10000	282335
Identical	20000	604671

Kind of List Data	List Size	List References
Sorted List	10	110
Sorted List	100	5645
Sorted List	200	21295
Sorted List	500	128245
Sorted List	1000	506495
Sorted List	2000	2012995
Sorted List	2250	2545870
Sorted List	3000	4519495
Sorted List	5000	12532495
Sorted List	10000	50064995
Sorted List	20000	200129995

Kind of List Data	List Size	
Reverese Sorted	10	105
Reverese Sorted	100	5595
Reverese Sorted	200	21195
Reverese Sorted	500	127995
Reverese Sorted	1000	505995
Reverese Sorted	2000	2011995
Reverese Sorted	2250	2544745
Reverese Sorted	3000	4517995
Reverese Sorted	5000	12529995
Reverese Sorted	10000	50059995
Reverese Sorted	20000	200119995

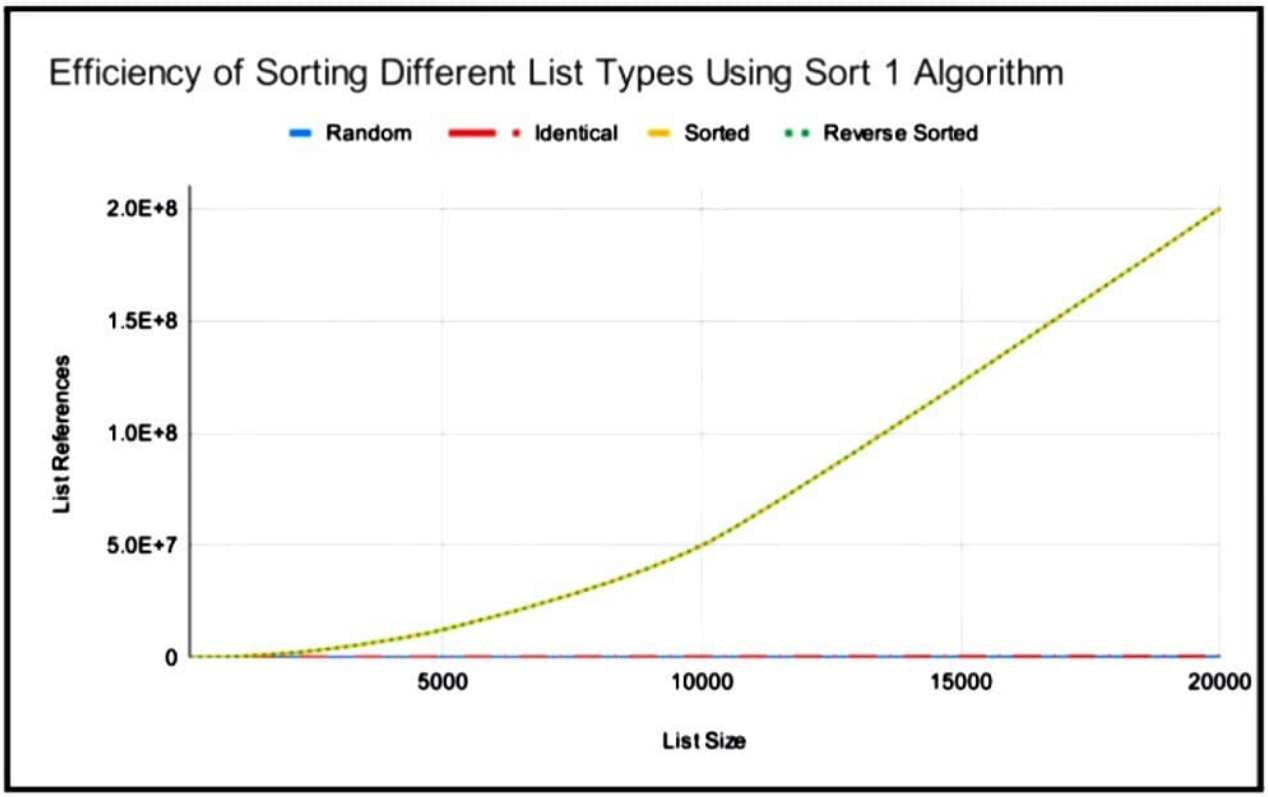
Sort 2

Kind of List Data	List Size	List References
Random	10	29999980
Random	100	29999800
Random	200	29999600
Random	500	29999000
Random	1000	29998000
Random	2000	29996000
Random	2250	29995500
Random	3000	29994000
Random	5000	73788878
Random	10000	297804290
Random	20000	1196928824

Kind of List Data	List Size	List References
Identical	10	29999980
Identical	100	29999800
Identical	200	29999600
Identical	500	29999000
Identical	1000	29998000
Identical	2000	29996000
Identical	2250	29995500
Identical	3000	29994000
Identical	5000	29990000
Identical	10000	29980000
Identical	20000	29960000

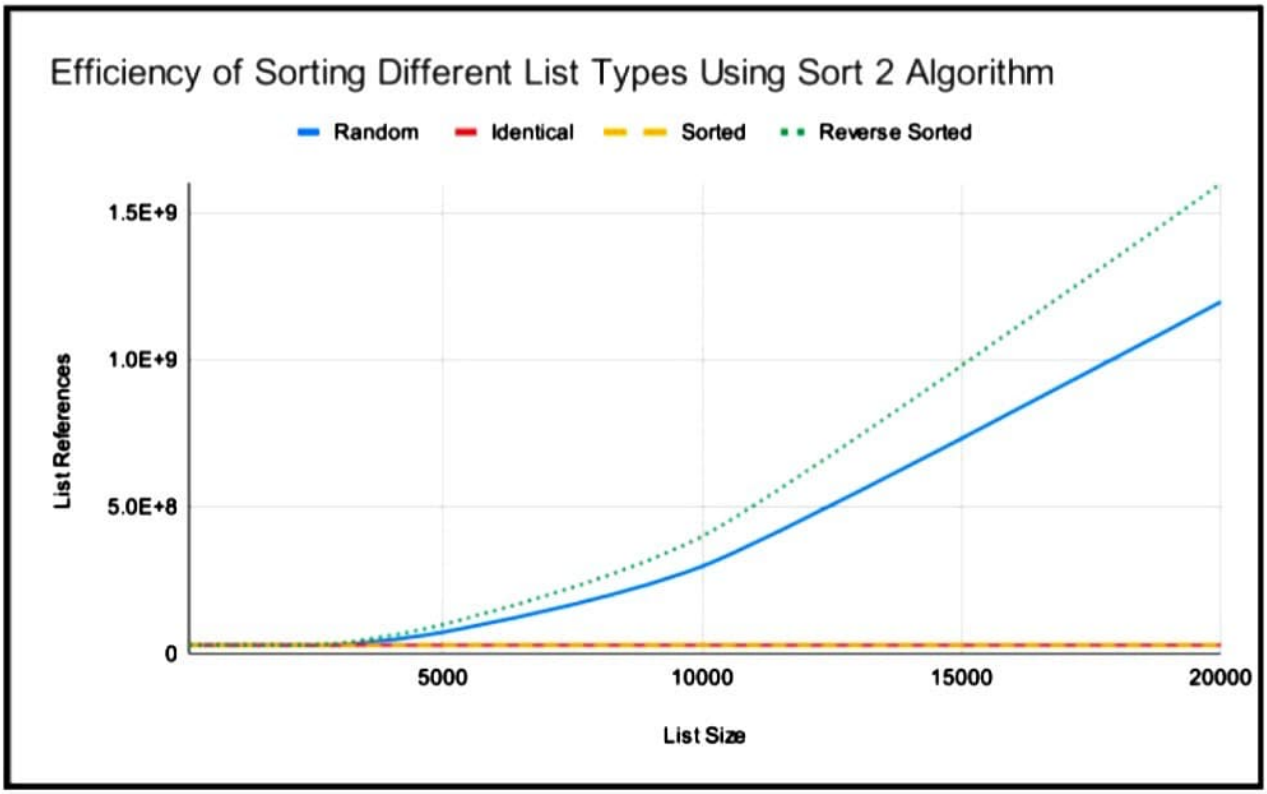
Kind of List Data	List Size	List References
Sorted List	10	29999980
Sorted List	100	29999800
Sorted List	200	29999600
Sorted List	500	29999000
Sorted List	1000	29998000
Sorted List	2000	29996000
Sorted List	2250	29995500
Sorted List	3000	29994000
Sorted List	5000	29990000
Sorted List	10000	29980000
Sorted List	20000	29960000

Kind of List Data	List Size	
Reverese Sorted	10	29999980
Reverese Sorted	100	29999800
Reverese Sorted	200	29999600
Reverese Sorted	500	29999000
Reverese Sorted	1000	29998000
Reverese Sorted	2000	29996000
Reverese Sorted	2250	29995500
Reverese Sorted	3000	35994000
Reverese Sorted	5000	99990000
Reverese Sorted	10000	399980000
Reverese Sorted	20000	1599960000



WORST CASE: $O(n^2)$

Sorted and Reverse Sorted have similar data so we can choose either one to determine worst case runtime. Looking at the graph, these two lines have more of a parabolic shape and Random and Identical have a linear shape. Random and Identical run in linear time i.e O(n). We can determine that sorted and reverse sorted is $O(n^2)$ by looking directly at the data. If we look at the list size from 5000 to 20000, the list references goes from 12,529,995 to 200,119,995. If we divide 200,119,995 by 12,529,995, we get 15.97 or about 16. This means that when the list size quadruples, the number of list references increases by a factor of 16. This is indicative of n^2 since 4^2 is 16. We can double check this by going from a list size of 1000 to 20000. The list references with a list size of 1000 was 505,995. If we divide 200,119,995 by 505,995, we get 395.497 which is about 400. Thus when the list size increases by a factor of 20, the number of list references increases by a factor of 400. This is indicative of n^2 as 20^2 is 400. Thus, the worst case of Sort 1 is $O(n^2)$



WORST CASE: O(n^2)

Looking at the chart, we can see that the worst case is for the Reverse Sorted List. Identical and the Sorted List run in constant time, O(1) and both Random and Reverse Sorted run in constant time until the input size reaches about 3000. Then, the function becomes n^2. Looking at reverse sorted with a list size from 5000 to 20000, the list references goes from 99,990,000 to 1,599,960,000. If we divide 1,599,960,000 by 99,990,000, we get 16.01, meaning that when the list size increases by a factor of 4, the number of list references increases by a factor of 16. This is indicative of n2 because 4^2 is 16. We can double check this by checking the list size from 3000 to 20,000. At a list size 3000, the number of list references is 35,994,000. If we divide 1,599,960,000 by 35,994,000, we get 44.4. 20,000/3,000 is around 6.66 and 6.66^2 is 44.4, indicating that this runs in quadratic time or n^2. However, as mentioned before, this function runs in constant time until it reaches a list size of 3000, but we assume the worst case possibility for run time complexity, thus Sort 2 is O(n^2).