Activity 1. Time measurements for sorting algorithms

Selection measurements:

n	sorted(t)	inverse(t)	random(t)
10000	136	234	265
20000	481	928	967
40000	1898	3710	3818
80000	7567	14731	15174
160000	30316	58520	60723
320000	122958	233710	242588
640000	490348	930424	964239
1280000	too much time	too much time	too much time

And now the theoretical values with its complexity:

n	sorted(t)	inverse(t)	random(t)
10000	136	234	265
20000	544	936	1060
40000	1924	3712	3868
80000	7592	14840	15272
160000	30268	58924	60696
320000	121264	234080	242892
640000	491832	934840	970352
1280000	1961392	3721696	3856956
Complexity	O(n^2)	O(n^2)	O(n^2)

As I saw with the formula, I obtained O(n^2) values in all cases which is its best case. The formula I used is: t2 = ((n2^2)/(n1^2)*t1).

Quicksort with the central element as the pivot measurements:

n	sorted(t)	inverse(t)	random(t)
10000	3	7	14
20000	10	13	19
40000	10	23	40
80000	8	43	80
160000	20	44	166
320000	41	94	352
640000	86	298	745

1280000	170	416	1552

And now the theoretical values with its complexity:

n	sorted(t)	inverse(t)	random(t)
10000	3	7	14
20000	6,451544993	15,05360498	30,10720997
40000	21,39980421	27,81974548	40,65962801
80000	21,30824021	49,00895247	85,23296083
160000	16,98233562	91,28005394	169,8233562
320000	42,31378213	93,09032069	351,2043917
640000	86,48388504	198,2801267	742,4957935
1280000	180,9175946	626,9005023	1567,251256
Complexity	O(n log n)	O(n log n)	O(n log n)

All the cases except of sorted are similar to the theoretical complexity so I suppose that something went wrong or it was too fast to be measured, the complexity is O(n log n) which is the best . The formula I used is: t2 = (log(n2)/log(n1))*t1

Insertion measurements:

n	sorted(t)	inverse(t)	random(t)
10000	6	260	132
20000	1	641	324
40000	1	2564	1269
80000	0	10169	5063
160000	1	41004	20109
320000	1	164053	80695
640000	2	651582	324380
1280000	6	too much time	too much time

And now the theoretical values:

n	sorted(t)	inverse(t)	random(t)
10000	6	260	132
20000	12	1040	528
40000	2	2564	1296
80000	2	10256	5076
160000	0	40676	20252

320000	2	164016	80436
640000	2	656212	322780
1280000	4	2606328	1297520
Complexity	O(n)	O(n^2)	O(n^2)

As we can see the complexity is for sorted O(n) which is the best case and the inverse and random I obtained both the average case and the worst case O(n^2) the formula I used is: for O(n) t2=(n2/n1)*t1 and for O(n^2) t2 = ((n2^2)/(n1^2)*t1).

Bubble measurements:

n	sorted(t)	inverse(t)	random(t)
10000	1	1107	1184
20000	0	4390	4696
40000	1	17480	22253
80000	4	70450	98784
160000	0	280260	417019
320000	0	1112270	1702963
640000	3	too much time	too much time
1280000	5	too much time	too much time

And now the theoretical values with its complexity:

n	sorted(t)	inverse(t)	random(t)
10000	1	1107	1184
20000	4	4428	4736
40000	0	17560	18784
80000	4	69920	89012
160000	16	281800	395136
320000	0	1121040	1668076
640000	0	4449080	6811852
1280000	12	UKNOWN	UKNOWN
Complexity	O(n^2)	O(n^2)	O(n^2)

All the cases except of sorted are similar to the theoretical complexity so I suppose that something went wrong or it was too fast to be measured, the complexity is O(n^2) which is the worst case. The formula I used is t2 = ((n2^2)/(n1^2)*t1).

Activity 2. QuicksortFateful.

I think that the pivot is the first element of the array as it takes the “left” variable position inside the array and since “left” is zero the first element of the array that is going to be sorted it is the first element.