Activity 1. Bubble algorithm

TABLE 1			
n	t ordered	t reverse	t random
10000	593	2967	1351
2*10000	2294	9188	6039
2**2*10000	9435	39006	22477
2**3*10000	37229	Oot	Oot
2**4*10000	Oot	Oot	Oot

Complexities:

Best case – O(n^2)

Worst case – O(n^2)

Average case – O(n^2)

For the Bubble algorithm all cases have the same complexity so if we make the proper computations, we can see that the results obtained follow the complexity expected. T2 = k^2*t1, where k = n2/n1. The following table shows the computed time with the complexity expected for a problem size of 40000.

TABLE 1			
n	t ordered	t reverse	t random
10000	593	2967	1351
2*10000	2294	9188	6039
2**2*10000	9435	39006	22477
2**3*10000	37229	Oot	Oot
2**4*10000	Oot	Oot	Oot
Computed t for n = 40000	9176	36752	24156

Activity 2. Selection algorithm

Algorithmics	Student information	Date	Number of session
	UO: 293860	20/02/2024	4
	Surname: López Álvarez		
	Name: Juan		

TABLE 2			
n	t ordered	t reverse	t random
10000	477	568	479
2*10000	1885	3139	1930
2**2*10000	7769	9013	11219
2**3*10000	31346	36893	49354
2**4*10000	Oot	Oot	Oot

Complexities:

Best case – O(n^2)

Worst case – O(n^2)

Average case – O(n^2)

In the selection algorithm we have the same complexities as in the previous algorithm so we can use the same formula.

For the ordered vector we get t1 = 1885ms, n1 = 20000, n2 = 40000, with these values we get that t2 = 7540ms quite close to the results obtained.

For the reverse vector we do the same with the same n1 = 40000, n2 = 80000 and t1 = 9013ms, and the result is t2 = 36052ms.

Finally we repeat the process with the random vector with the values n1 = 10000, n2 = 20000 and t1 = 479ms, the result is t2 = 1916ms.

Activity 3. Insertion algorithm

Algorithmics	Student information	Date	Number of session
	UO: 293860	20/02/2024	4
	Surname: López Álvarez		
	Name: Juan		

TABLE 3			
n	t ordered	t reverse	t random
10000	LoR	804	365
2*10000	LoR	3435	1454
2**2*10000	LoR	12103	5943
2**3*10000	LoR	49671	23483
2**4*10000	LoR	Oot	Oot
2**5*10000	LoR	Oot	Oot
2**6*10000	LoR	Oot	Oot
2**7*10000	LoR	Oot	Oot
2**8*10000	61	Oot	Oot
2**9*10000	118	Oot	Oot
2**10*10000	233	Oot	Oot
2**11*10000	464	Oot	Oot
2**12*10000	933	Oot	Oot
2**13*10000	1865	Oot	Oot

Complexities:

Best case – O(n)

Worst case – O(n^2)

Average case – O(n^2)

If we do the calculations for the best case we can see that it follows the complexity. T2 = k*t1, where k = n2/n1. Applying this formula to two consecutive results we get the following: t2= $((2^{10} * 10000)/ (2^{9} * 10000)) * 118$ = 236ms, that is very close to the result obtained in the execution.

We do the same with the worst case and average case which have the same complexity:

Worst case – t1=804ms, n1 = 10000, n2 =2*10000. T2 = k^2*t1; t2 = $((2*10000)^2 / (10000)^2)*804$ = 3216ms

Average - t1=365ms, n1 = 10000, n2 =2*10000. Applying the same formula we get that t2 = 1460ms.

Activity 4. Quicksort algorithm

Algorithmics	Student information	Date	Number of session
	UO: 293860	20/02/2024	4
	Surname: López Álvarez		
	Name: Juan		

TABLE 4			
n	t ordered	t reverse	t random
25000	50	56	353
2*25000	102	112	397
2**2*25000	211	231	547
2**3*25000	435	505	1158
2**4*25000	933	987	2829
2**5*25000	1883	2048	8419
2**6*25000	3979	6482	25071

Complexities:

Best case – O(n*log n)

Worst case – O(n^2)

Average case – O(n*log n)

The times obtained make sense because when we get a good pivot as we do in this algorithm, we always tend towards the best-case scenario. The division scheme analysis of this algorithm gives us that a=2, b=2, and k = 1, therefore as a = b^k the complexity is O(n^k * log n). In the random sorted vector times, we can't really rely on the results because times can vary a lot depending on the vector itself and the pivot chosen as a result of the median of three, but more or less are consistent with the complexity of the algorithm.

The time it would take the previous algorithms to complete the sorting of a vector of 16 million elements is the following:

- Bubble: t = (16M/10000)^2*1351ms = 3.458.560.000ms which is more or less 40 days.
- Selection: t = (16M/10000)^2*479ms = 1.226.240.000ms which is more or less 14 days.
- Insertion: t = (16M/10000)^2*365ms = 934.400.000ms which is approximately 11 days.

Activity 5. Quicksort + Insertion algorithm

To measure this algorithm the size of the problem used was 16 million elements and the threshold to switch from the quicksort algorithm to the insertion one was a parameter passed to the function so that we could change it.

TABLE 5	
n	t random
Quicksort	23002
Quicksort + Insertion (k=5)	19371
Quicksort + Insertion (k=10)	17303
Quicksort + Insertion (k=20)	17141
Quicksort + Insertion (k=30)	16668
Quicksort + Insertion (k=50)	16856
Quicksort + Insertion (k=100)	15552
Quicksort + Insertion (k=200)	14622
Quicksort + Insertion (k=500)	22376
Quicksort + Insertion (k=1000)	49120

As we can see the best results are obtained when k is between 30 and 200, this is because the insertion algorithm works well with a small size problem. However, when we increase the threshold, we see that the times get closer to the Insertion algorithm, which makes sense since we are using more this algorithm, which is slower. A similar thing happens on the other direction, when we reduce a lot the threshold, we get times similar to the Quicksort algorithm times because we are barely using the insertion one.