


Algorithmics	Student information	Date	Number of session
	UO:300895	21/2/2025	4
	Surname: Franco Martinez	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Ivan		



Activity 1. [BUBBLE ALGORITHM]

Table1:			
n	t ordered	t reverse	t random
10000	329	1539	1066
2*10000	1284	6080	4265
2**2*10000	5130	24933	17075
2**3*10000	20848	OoT	OoT
2**4*10000	OoT	OoT	OoT

The obtained times match the expected results because the Bubble Sort algorithm performs one additional iteration for each unordered element in the vector. That's why the "tordered" time is the smallest since the vector is already sorted. In this case, the algorithm detects that no swaps are needed and stops early, requiring only a few iterations. The "treverse" time is the largest because this is the worst-case scenario, where every element is in the opposite order, requiring the maximum number of iterations and swaps. The "trandom" time falls between the two, as its execution time depends on how disorganized the vector is. The more unordered the elements, the more iterations and swaps are needed.

Activity 2. [SELECTION ALGORITHM]

Table2:			
n	t ordered	t reverse	t random
10000	331	307	326
2*10000	1277	1184	1281
2**2*10000	5087	4736	5108
2**3*10000	20482	18821	20393
2**4*10000	OoT	OoT	OoT

The times obtained match with the expected results since the selection algorithm will do the same number of iterations independently on how disordered the vector is. That's why we obtain similar times in the three cases

Algorithmics	Student information	Date	Number of session
	UO:300895	21/2/2025	4
	Surname: Franco Martinez		
	Name: Ivan		

Activity 3. [INSERTION ALGORITHM]

Table3:			
n	t ordered	t reverse	t random
10000	LoR	309	157
2*10000	LoR	1215	613
2**2*10000	LoR	4883	2442
2**3*10000	LoR	19590	9765
2**4*10000	LoR	OoT	38987
2**5*10000	LoR	OoT	OoT
2**6*10000	LoR	OoT	OoT
2**7*10000	LoR	OoT	OoT
2**8*10000	LoR	OoT	OoT
2**9*10000	96	OoT	OoT
2**10*10000	190	OoT	OoT
2**11*10000	376	OoT	OoT
2**12*10000	753	OoT	OoT
2**13*10000	1514	OoT	OoT

The times obtained match with the expected results since insertion algorithm iterates the vector until it find a element in its wrong place, the it picks the element and iterates until it finds the right place. In “tordered” as the vector is already ordered the execution time is fust, in the “t reverse”, would be the worst case since it has to iterate the maximum number of iterations, and in the “t random” it would depend on who much disorganized the vector is.

Algorithmics	Student information	Date	Number of session
	UO:300895	21/2/2025	4
	Surname: Franco Martinez		
	Name: Ivan		

Activity 4. [QUICKSORT ALGORITHM]

Table4:			
n	t ordered	t reverse	t random
250000	LoR	LoR	100
2*250000	63	74	198
2**2*250000	129	150	421
2**3*250000	268	305	904
2**4*250000	553	624	1966
2**5*250000	1139	1287	4461
2**6*250000	2362	2649	10859

The times obtained match with the expected results since quicksort algorithm use the divide and conquer policy dividing it into many subsectors and ordering it individually.

The “tordered” will be a little bit faster than “treverse” , since it doesn’t have to order the vectors, but not much faster since the division into subsectors is done anyway, and also .

In the “trandom” times would be influenced depending on what pivot the algorithm choose since it

Activity 5. [QUICKSORT + INSERTION ALGORITHM]

Algorithmics	Student information	Date	Number of session
	UO:300895	21/2/2025	4
	Surname: Franco Martinez		
	Name: Ivan		

Table5:

n	t random
Quicksort	10260
Quicksort +Insertion (k=5)	10678
Quicksort +Insertion (k=10)	10477
Quicksort +Insertion (k=20)	10318
Quicksort +Insertion (k=30)	10124
Quicksort +Insertion (k=50)	9854
Quicksort +Insertion (k=100)	8903
Quicksort +Insertion (k=200)	7380
Quicksort +Insertion (k=500)	10181
Quicksort +Insertion (k=1000)	18076

The times match with the results obtained since the insertion algorithm works better than the quicksort for less amount of data and quick sort works better than insertion with large amount of data, in the times of the table we can see how in k=200 is the equilibrium point where quicksort and insertion works better together, then as k continuous increasing and insertion works worst with more data the times increase.