Algorithmics	Student information	Date	Number of session
	UO: 300829	20/02/2025	2
	Surname: Cid Lazcano		
	Name: Izan		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Bubble Algorithm

n	t ordered	t reverse	t random
10000	679	2635	1929
2*10000	2761	10238	7353
2**2*10000	9101	40773	47100
2**3*10000	50733	OoT	OoT
2**4*10000	OoT	OoT	OoT

The measurements align with the expected results since we are getting the lower values when the vector is already ordered, which is the best case for the bubble algorithm, and then in the reverse and random ordered vectors we get the quadratic complexity $O(n^2)$ which is the one expected for average and worst cases with the bubble algorithm.

Activity 2. Selection Algorithm

n	t ordered	t reverse	t random
10000	640	633	670
2*10000	2323	2333	2422
2**2*10000	9037	8950	9311
2**3*10000	35847	34051	36438
2**4*10000	OoT	OoT	OoT

The values obtained agree with what is expected since they all share the same $O(n^2)$ complexity, just as it should be for the worst, average and best-case scenario when using selection algorithm.

Algorithmics	Student information	Date	Number of session
	UO: 300829	20/02/2025	2
	Surname: Cid Lazcano		
	Name: Izan		

Activity 3. Insertion Algorithm

n	t ordered	t reverse	t random
10000	LoR	637	340
2*10000	LoR	2197	1165
2**2*10000	LoR	9375	4407
2**3*10000	LoR	35050	17299
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	109	OoT	OoT
2**9*10000	182	OoT	OoT
2**10*10000	353	OoT	OoT
2**11*10000	681	OoT	OoT
2**12*10000	1381	OoT	OoT
2**13*10000	2747	OoT	OoT

The obtained results are coherent with the expectations since we are getting a linear complexity O(n) for the ordered vector, that is the best-case scenario, and then quadratic complexities $O(n^2)$ for the other two cases. Furthermore, the worst-case scenario is the reverse one since the insertion algorithm works better with small sizes or partially ordered elements, and we can see that reflected in the results.

Activity 4. Quicksort Algorithm

n	t ordered	t reverse	t random
250000	59	66	247
2*250000	130	165	401
2**2*250000	294	310	824
2**3*250000	528	571	1880
2**4*250000	1034	1231	3777
2**5*250000	2149	2783	8598
2**6*250000	4392	7309	20453

Algorithmics	Student information	Date	Number of session
	UO: 300829	20/02/2025	2
	Surname: Cid Lazcano		
	Name: Izan		

Activity 5. Quicksort + Insertion Algorithm

n	t random
Quicksort	20687
Quicksort + Insertion (k=5)	24785
Quicksort + Insertion (k=10)	19979
Quicksort + Insertion (k=20)	19700
Quicksort + Insertion (k=30)	17604
Quicksort + Insertion (k=50)	14704
Quicksort + Insertion (k=100)	8293
Quicksort + Insertion (k=200)	4782
Quicksort + Insertion (k=500)	2772
Quicksort + Insertion (k=1000)	2565

With a very small k (like 5), the algorithm makes many recursive calls before switching to insertion sort. This increases overhead, as the recursive Quicksort calls—even on very small arrays—carry some constant time costs. Insertion sort has a best-case time complexity of O(n) when the array is nearly sorted. With higher k, more elements are sorted using insertion sort on subarrays that are already partially sorted by previous Quicksort partitions. This explains the improvement in performance as k increases. If k becomes too high, there could be a point where insertion sort's $O(n^2)$ worst-case behaviour might start to hurt performance if the subarrays are not already nearly sorted.

n	t random
Quicksort + Insertion (k=1050)	36465
Quicksort + Insertion (k=1100)	38050