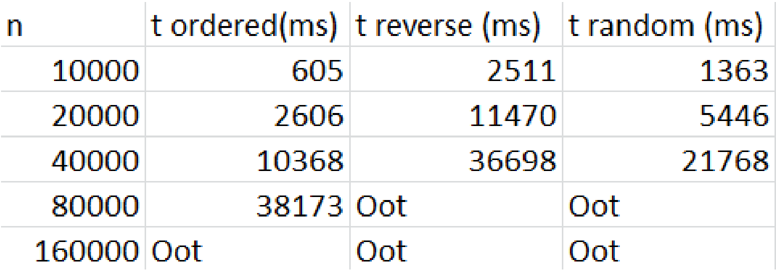
Algorithmics	Student information	Date	Number of session
	UO: 295368		4
	Surname: Álvarez Hernández		
	Name: Miguel		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [Direct exchange or Bubble algorithm]

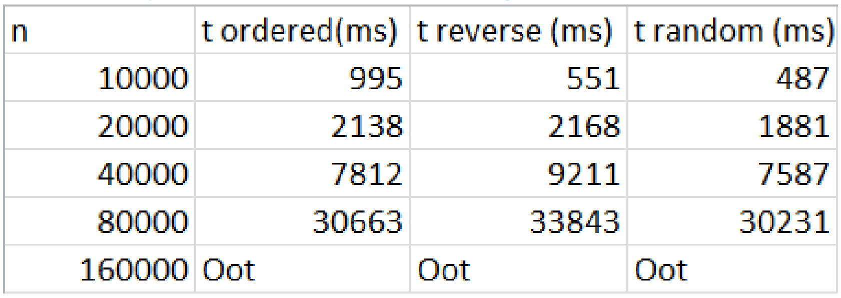
The measurements are taken with a processor i7-4790 and 8 GB DDR3



n	t ordered(ms)	t reverse (ms)	t random (ms)
10000	605	2511	1363
20000	2606	11470	5446
40000	10368	36698	21768
80000	38173	Oot	Oot
160000	Oot	Oot	Oot

As expected, the ordered array is the one that spends less time and the reverse the one which spends more time. This is because with the ordered array it doesn`t have to do much work while with the reverse one it has to change all the positions. The random one spends less time than the reverse one as some positions will be already ordered so the algorithm has to do less work.

Activity 2. [Selection algorithm]

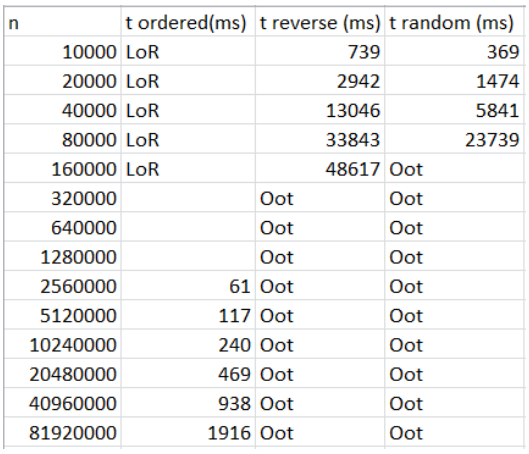


n	t ordered(ms)	t reverse (ms)	t random (ms)
10000	995	551	487
20000	2138	2168	1881
40000	7812	9211	7587
80000	30663	33843	30231
160000	Oot	Oot	Oot

It agrees with what is expected as the selection algorithm takes the lowest element every time and put it at the beginning of the array. This is why the reverse array works better this time than the ordered array because this one requires to go to the end of the array searching for the lowest element although it is the already selected. Random needs less time as some are already ordered.

Algorithmics	Student information	Date	Number of session
	UO: 295368		4
	Surname: Álvarez Hernández		
	Name: Miguel		

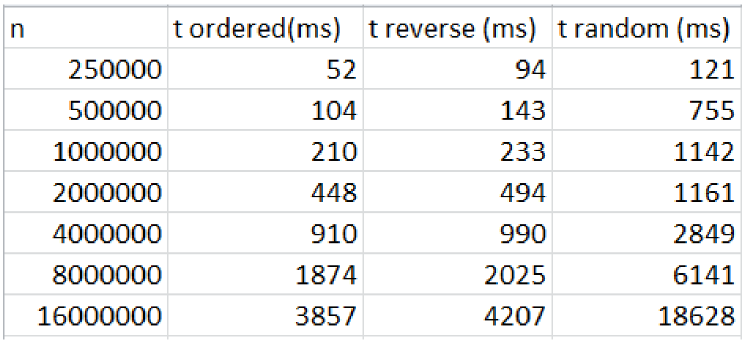
Activity 3. [Insertion algorithm]



n	t ordered(ms)	t reverse (ms)	t random (ms)
10000	LoR	739	369
20000	LoR	2942	1474
40000	LoR	13046	5841
80000	LoR	33843	23739
160000	LoR	48617	Oot
320000		Oot	Oot
640000		Oot	Oot
1280000		Oot	Oot
2560000	61	Oot	Oot
5120000	117	Oot	Oot
10240000	240	Oot	Oot
20480000	469	Oot	Oot
40960000	938	Oot	Oot
81920000	1916	Oot	Oot

As we expected the times are reasonable because this algorithm inserts each element into its position inside a sub vector already ordered so for the ordered vector requires less time than the random one and the reverse one as it is already ordered so all interactions are omitted. The random spends less time than the reverse one as is completely the other way around, in the reverse one it has to be inserting the element one by one traversing the entire vector while in the random some elements are already ordered so it has to do less interactions.

Activity 4. [Quicksort algorithm]



n	t ordered(ms)	t reverse (ms)	t random (ms)
250000	52	94	121
500000	104	143	755
1000000	210	233	1142
2000000	448	494	1161
4000000	910	990	2849
8000000	1874	2025	6141
16000000	3857	4207	18628

As expected, these algorithms spend little time in any kind of array, showing us that it is the most effective sorting algorithms in terms of performance. We can see that the random vector spends a little more time than the others two this is probably because it requires more interactions.

We can see how many days would each of those three methods (Bubble, Selection and Insertion) take in doing the same by doing a rule of 3 in each case taking in account the size of 16M

We use the formula t2 = k^n*t1 where k = n2/n1

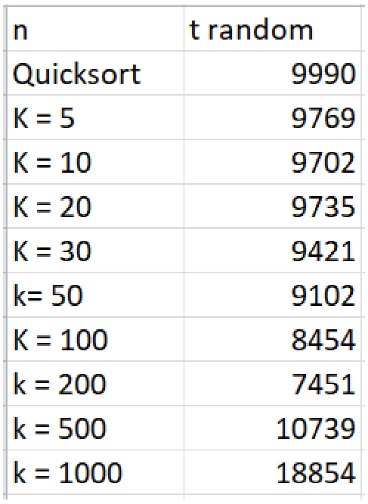
Bubble -> t = (16M/10000)^2* 1363ms = 3489280000ms = 40,39 days

Selection -> t = (16M/10000)^2*487ms = 1246720000ms = 14,3 days

Insertion -> t = (16M/10000)^2*369ms = 944640000ms = 10,9 days

Activity 5. [Quicksort + Insertion algorithm]

This time the measurements are done with a processor i5-12600KF and 32GB DDR5



n	t random
Quicksort	9990
K = 5	9769
K = 10	9702
K = 20	9735
K = 30	9421
k= 50	9102
K = 100	8454
k = 200	7451
k = 500	10739
k = 1000	18854

After taking the measurements, we can conclude that the range in where we can see the difference in terms of improving the performance is between k = 50 and k = 200. So we can see that it is interesting to mix algorithms to get a better performance.