Algorithmics	Student information	Date	Number of session
	UO: 294866	23/02/2024	3
	Surname: Menendez		
	Name: Javier		

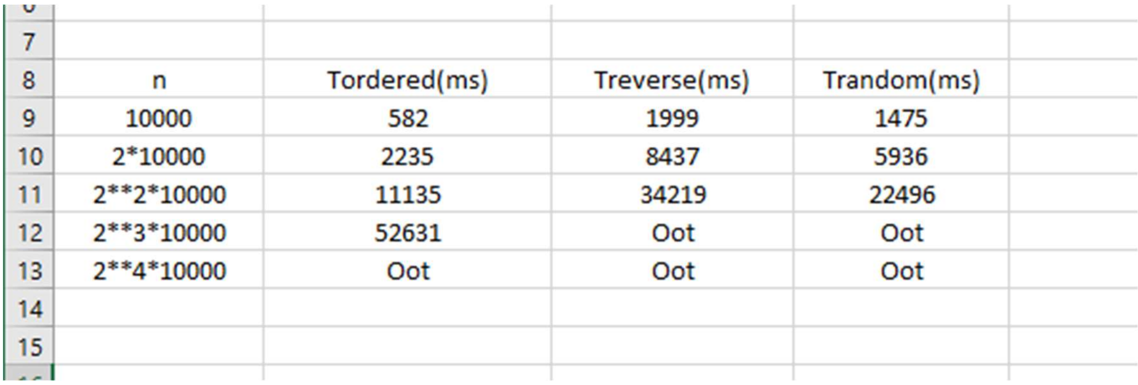
Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [Direct exchange or Bubble algorithm]

Measurements were taken with:

- CPU: Intel® Core™ i7-4790 CPU @ 3.60 GHz
- RAM: 8 GB DDR3

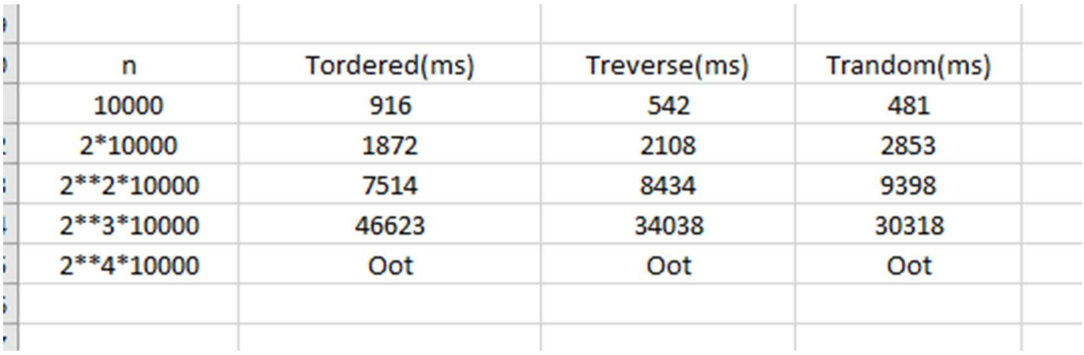


7				
8	n	Tordered(ms)	Treverse(ms)	Trandom(ms)
9	10000	582	1999	1475
10	2*10000	2235	8437	5936
11	2**2*10000	11135	34219	22496
12	2**3*10000	52631	Oot	Oot
13	2**4*10000	Oot	Oot	Oot
14				
15				

In the sorted one the vector is already sorted so the Bubble algorithm takes less time. Then in the random one some are well allocated and others not, but it makes the algorithm expend less time ordered every element of the vector, so if we receive the vector at reserve order is normal that takes much more time than in the two cases before.

Algorithmics	Student information	Date	Number of session
	UO: 294866	23/02/2024	3
	Surname: Menendez		
	Name: Javier		

Activity 2. [Selection Algorithm]

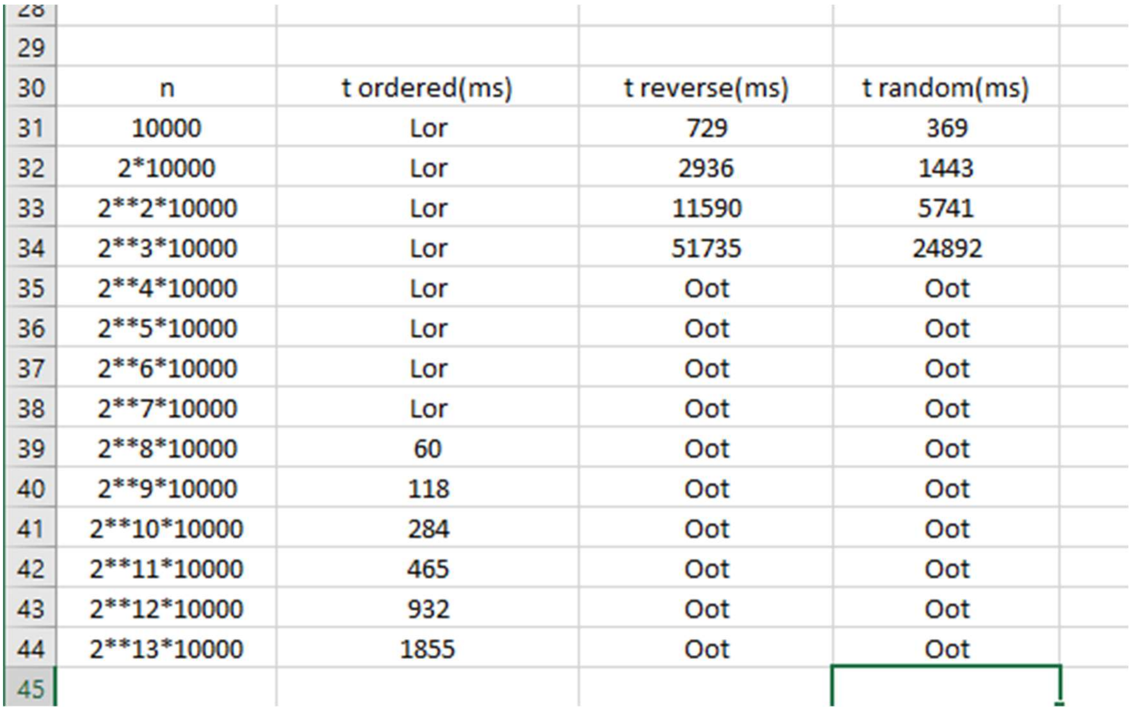


n	Tordered(ms)	Treverse(ms)	Trandom(ms)
10000	916	542	481
2*10000	1872	2108	2853
2**2*10000	7514	8434	9398
2**3*10000	46623	34038	30318
2**4*10000	Oot	Oot	Oot

Yes, it completely agrees with what it was expected because in this algorithm it constantly takes the lowest element so in a reverse vector the lowest element is at the beginning of the queue. Random is randomly generated so some of the elements help for the complexity and others do not, and in ordered require to constantly in each iteration going until the end comparing the lowest element with others so is normal that it takes more time.

Algorithmics	Student information	Date	Number of session
	UO: 294866	23/02/2024	3
	Surname: Menendez		
	Name: Javier		

Activity 3. [Insertion Algorithm]

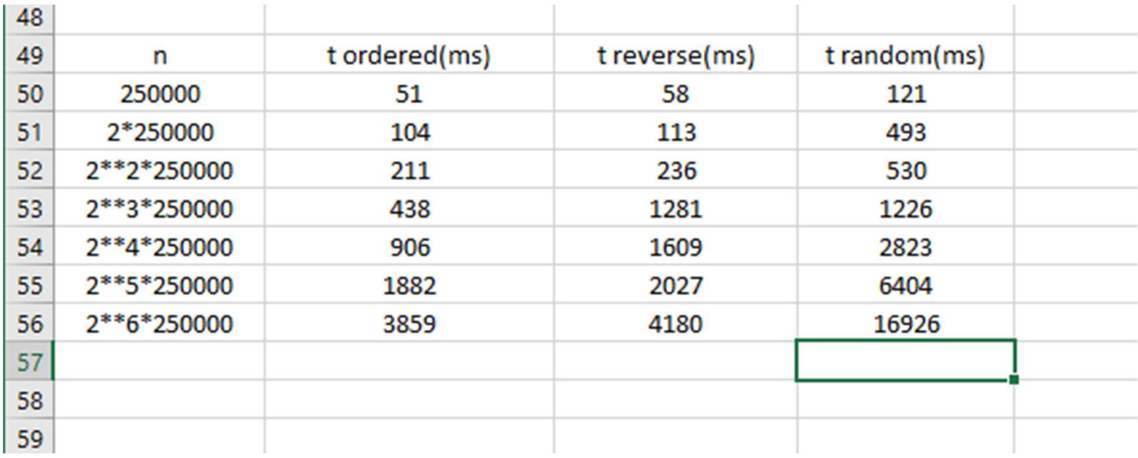


n	t ordered(ms)	t reverse(ms)	t random(ms)
10000	Lor	729	369
2*10000	Lor	2936	1443
2**2*10000	Lor	11590	5741
2**3*10000	Lor	51735	24892
2**4*10000	Lor	Oot	Oot
2**5*10000	Lor	Oot	Oot
2**6*10000	Lor	Oot	Oot
2**7*10000	Lor	Oot	Oot
2**8*10000	60	Oot	Oot
2**9*10000	118	Oot	Oot
2**10*10000	284	Oot	Oot
2**11*10000	465	Oot	Oot
2**12*10000	932	Oot	Oot
2**13*10000	1855	Oot	Oot

The time taken accords with what was expected because this algorithm inserts each element into its position inside a sub vector already ordered so is normal that the ordered vector requires less time than the random one and this, less than a vector like the reverse one which is completely the other way around.

	Student information	Date	Number of session
Algorithmics	UO: 294866	23/02/2024	3
	Surname: Menendez		
	Name: Javier		

Activity 4. [Quicksort Algorithm]



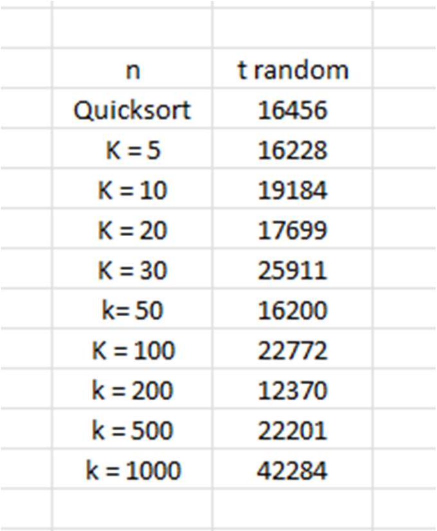
48				
49	n	t ordered(ms)	t reverse(ms)	t random(ms)
50	250000	51	58	121
51	2*250000	104	113	493
52	2**2*250000	211	236	530
53	2**3*250000	438	1281	1226
54	2**4*250000	906	1609	2823
55	2**5*250000	1882	2027	6404
56	2**6*250000	3859	4180	16926
57				
58				
59				

It accords with the times that were expected and we can see the real potential of this algorithm, because in every situation of the vector and practically regardless of the iteration number it expends little time to done it.

Under the time taken by doing the Bubble algorithm into a random vector, with 16M it would take 43.70 days. (t = ((16M/10.000) ^2) * 1475ms = 3776000000 ms (43.70 days))
Under the time taken by doing the Selection algorithm into a random vector, with 16M it would take 14.25 days. (t = ((16M/10.000) ^2) * 481ms = 1231360000 ms (14.25 days))
Under the time taken by doing the Insertion algorithm into a random vector, with 16M it would take 10.935 days. (t = ((16M/10.000) ^2)*369 ms = 944640000 ms (10.935 days))

	Student information	Date	Number of session
Algorithmics	UO: 294866	23/02/2024	3
	Surname: Menendez		
	Name: Javier		

Activity 5. [Quicksort + Insertion Algorithm]



n	t random
Quicksort	16456
K = 5	16228
K = 10	19184
K = 20	17699
K = 30	25911
k= 50	16200
K = 100	22772
k = 200	12370
k = 500	22201
k = 1000	42284

The conclusion for this algorithm is that being able to change between algorithms when one of them is better than the other to complete the execution is worth. I think that combining them is quite interesting because it can be more useful than using them independently.