Algorithmics	Student information	Date	Number of session
	UO: UO294786	21/02/24	2
	Surname: Álvarez Iglesias		
	Name: Rafael		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [BUBBLE ALGORITHM]

<u>YOU ARE REQUESTED TO:</u> After measuring times, fill in the table:

n	Time(ms)		
	Ordered	Reverse	Random
10000	570	1954	1338
2*10000	2283	8478	7652
2**2*10000	9272	32911	28443
2**3*10000	38391	129802	104165
2**4*10000	189348	OoT	OoT

Explain whether the different times obtained agree with what is expected, according to the time complexity studied:

The bubble sort algorithm has a complexity of $O(n^2)$ for every case. Therefore, by applying the formula for the higher possible values (for the higher possible accuracy), the following results are obtained:

<u>Ordered:</u>

$T_{theoretical}(n = 2^4 * 10^4) = 153564$

$T_{obtained}(n = 2^4 * 10^4) = 189348$

<u>Reverse:</u>

$T_{theoretical}(n = 2^3 * 10^4) = 131644$

$T_{obtained}(n = 2^3 * 10^4) = 129802$

<u>Random:</u>

$T_{theoretical}(n = 2^3 * 10^4) = 113772$

$T_{obtained}(n = 2^3 * 10^4) = 104165$

The obtained values are close enough to the theoretical obtained values, so that the times fall within what was expected.

Algorithmics	Student information	Date	Number of session
	UO: UO294786	21/02/24	2
	Surname: Álvarez Iglesias		
	Name: Rafael		

Activity 2. [SELECTION ALGORITHM]

<u>YOU ARE REQUESTED TO:</u> Implement a SelectionTimes.java class to help you to fill in the following table:

n	Time(ms)		
	Ordered	Reverse	Random
10000	498	557	616
2*10000	1944	3091	2491
2**2*10000	9233	11022	11654
2**3*10000	34081	41916	42945
2**4*10000	OoT	OoT	OoT

Explain whether the different times obtained agree with what is expected, according to the time complexity studied:

The selection algorithm has a complexity of $O(n^2)$ for every case. Therefore, by applying the formula for the higher possible values (for the higher possible accuracy), the following results are obtained:

<u>Ordered:</u>

$T_{theoretical}(n = 2^3 * 10^4) = 36932$

$T_{obtained}(n = 2^3 * 10^4) = 34081$

<u>Reverse:</u>

$T_{theoretical}(n = 2^3 * 10^4) = 44088$

$T_{obtained}(n = 2^3 * 10^4) = 41916$

<u>Random:</u>

$T_{theoretical}(n = 2^3 * 10^4) = 46616$

$T_{obtained}(n = 2^3 * 10^4) = 42945$

The obtained values are close enough to the theoretical obtained values, so that the times fall within what was expected.

Algorithmics	Student information	Date	Number of session
	UO: UO294786	21/02/24	2
	Surname: Álvarez Iglesias		
	Name: Rafael		

Activity 3. [INSERTION ALGORITHM]

<u>YOU ARE REQUESTED TO:</u> Implement an InsertionTimes.java class to help you to fill in the following table:

n	Time(ms)		
	Ordered	Reverse	Random
10000	LoR	1696	383
2*10000	LoR	4292	3105
2**2*10000	LoR	12122	7255
2**3*10000	LoR	49661	24535
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	128	OoT	OoT
2**10*10000	388	OoT	OoT
2**11*10000	804	OoT	OoT
2**12*10000	1401	OoT	OoT
2**13*10000	2054	OoT	OoT

Explain whether the different times obtained agree with what is expected, according to the time complexity studied:

The bubble sort algorithm has a complexity of O(n) for the best case (when the array is ordered) and a complexity of O(n^2) for every other case. Therefore, by applying the formula for the higher possible values (for the higher possible accuracy), the following results are obtained:

Escuela de Ingeniería Informática

Ordered:

$T_{theoretical}(n = 2^{13} * 10^4) = 2802$

$T_{obtained}(n = 2^{13} * 10^4) = 2054$

Reverse:

$T_{theoretical}(n = 2^3 * 10^4) = 48488$

$T_{obtained}(n = 2^3 * 10^4) = 49661$

Random:

$T_{theoretical}(n = 2^3 * 10^4) = 29020$

$T_{obtained}(n = 2^3 * 10^4) = 24535$

The obtained values are close enough to the theoretical obtained values, so that the times fall within what was expected.

Algorithmics	Student information	Date	Number of session
	UO: UO294786	21/02/24	2
	Surname: Álvarez Iglesias		
	Name: Rafael		

Activity 4. [QUICKSORT ALGORITHM]

<u>**YOU ARE REQUESTED TO:**</u> **Implement a QuicksortTimes.java class to help you to fill in the following table:**

n	Time(ms)		
	Ordered	Reverse	Random
250000	78	76	161
2*250000	146	155	344
2**2*250000	313	326	728
2**3*250000	624	660	1585
2**4*250000	1281	1359	3552
2**5*250000	2648	2788	8394
2**6*250000	5447	5751	21523

Explain whether the different times obtained agree with what is expected, according to the time complexity studied:

The bubble sort algorithm has a complexity of $O(n^2)$ for the worst case (that is when the elements in the array are on reverse order) and a complexity of $O(n * \log(n))$ for the rest of the cases. Therefore, by applying the formula for the higher possible values (for the higher possible accuracy), the following results are obtained:

<u>Ordered:</u>

$T_{theoretical}(n = 2^6 * 25*10^4) = 5526$

$T_{obtained}(n = 2^6 * 25*10^4) = 5447$

<u>Reverse:</u>

$T_{theoretical}(n = 2^6 * 25*10^4) = 11152$

$T_{obtained}(n = 2^6 * 25*10^4) = 5751$

<u>Random:</u>

$T_{theoretical}(n = 2^6 * 25*10^4) = 17520$

$T_{obtained}(n = 2^6 * 25*10^4) = 21523$

The obtained values are close enough to the theoretical obtained values, so that the times fall within what was expected.

After seeing how long it takes to sort 16 million items initially in random order, calculate and compare (from the complexities and data in the tables above), how many days would each of those three methods (Bubble, Selection and Insertion) take in doing the same?

Taking the highest time obtained, and the complexity for each of the algorithms, the theoretical time for that execution can be calculated applying the corresponding formula:

<u>Bubble</u>:

$T(n = 2^6 * 25*10^4) = 48.23$ days.

<u>Selection</u>:

$T(n = 2^6 * 25*10^4) = 19.88$ days.

<u>Insertion</u>:

$T(n = 2^6 * 25*10^4) = 11.36$ days.

Algorithmics	Student information	Date	Number of session
	UO: UO294786	21/02/24	2
	Surname: Álvarez Iglesias		
	Name: Rafael		

Activity 5. [QUICKSORT + INSERTION ALGORITHM]

<u>YOU ARE REQUESTED TO:</u> Implement a QuicksortInsertion.java class that orders the elements of the vector as noted above. It is proposed to take times that, although not conclusive and decisive for comparing them, will be indicative. To do this, implement a class QuicksortInsertionTimes.java that will take times (for a size of n=16 million elements initially generated randomly) to fill in the following table:

n	t random
Quicksort	21523
Quicksort+Insertion (k=5)	22597
Quicksort+Insertion (k=10)	22300
Quicksort+Insertion (k=20)	21965
Quicksort+Insertion (k=30)	21632
Quicksort+Insertion (k=50)	20704
Quicksort+Insertion (k=100)	18515
Quicksort+Insertion (k=200)	14934
Quicksort+Insertion (k=500)	23672
Quicksort+Insertion (k=1000)	48405

Explain conclusions obtained from the previous table:

The quicksort + insertion makes sense when using a k that should be optimized. The data obtained shows that when k is higher than 20, some improvements are produced, up to almost 7 seconds of improvement with 200. However, going past that 20 returns times higher than the original algorithm, as high as double the original time.