Algorithmics	Student information	Date	Number of session
	UO: 258220	25-02-21	2
	Surname: Cuesta Martínez		
	Name: Miguel		

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Time measurements for sorting algorithms

1.1. Insertion

n	*sorted(µs)*	*inverse(µs)*	*random(µs)*
10000	10	49927	35050
20000	13	198239	122885
40000	25	603364	456712
80000	48	2290324	2030247
160000	92	9213810	8214324
320000	192	36352235	28308325

Insertion has a very good complexity for sorted collections, O(n). However, for any other case it is pretty slow with a $O(n^2)$ complexity. The number of interchanges is also fairly high, although at least it doesn't exceed the ammount we woud get with the Bubble algorithm.

1.2. Selection

n	*sorted(µs)*	*inverse(µs)*	*random(µs)*
10000	10858	37386	32383
20000	44616	144205	128095
40000	172736	577728	505102

80000	682727	2311075	2025116
160000	2739020	9246793	8082745
320000	10924285	36975962	32330133

The complexity for every collection type is $O(n^2)$. While this may make it seem like Insertion is at least better for already sorted collections, the lower times and reliability of performance in Selection would make it a better option overall. The number of exchanges is also lower than the other quadratic complexity options

1.3. Bubble

n	*sorted(µs)*	*inverse(µs)*	*random(µs)*
10000	20247	26900	33762
20000	82158	104897	140197
40000	329350	419028	535847
80000	1317694	1671267	2247269
160000	5270138	6705274	8987624
320000	21077288	26816815	37951546

The complexity is, again, $O(n^2)$ for every case. The trend in growth is of course the same as Selection because of that, but times consistently seem to be double the ones in Selection. While the reliability (or in other words, simility between times for all collection types) is the highest, there are few reasons apart from that to choose Bubble over Selection or Insertion.

Algorithmics	Student information	Date	Number of session
	UO: 258220	25-02-21	2
	Surname: Cuesta Martínez		
	Name: Miguel		

1.4. Quicksort with Central Element

n	*sorted(µs)*	*inverse(µs)*	*random(µs)*
10000	243	264	904
20000	266	346	1585
40000	501	660	3595
80000	1259	1448	7179
160000	2874	2983	15054
320000	5768	5778	32450
640000	11512	12960	67519

The complexity of Quicksort with central element is O(nlogn) for every type of collection. Out of the 4 algorithms presented, it's clearly the best: it has the shortest times for every case but sorted (where Insertion takes even less), and it is also very reliable in times. Sorted is also the most trivial scenario presented, so taking Insertion as better just for a shorter time there isn't really justifiable.

Activity 2. QuicksortFateful

The criteria for choosing the pivot is fairly simple: it takes the first element in the list. Because of this, most of the times we will get very bad times (around quadratic complexity). It is true that choosing the first element reduces the decision time of what to take as pivot drastically, but the optimal solution will be reached very sparingly. Because of this unreliability in performance, the criteria is pretty bad and easily outclassed by Median of Three or Central Element.