Activity 1. Time measurements for sorting algorithms.

[ANSWER].

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm: | Bubble |  |  |
| N | Sorted | Inverse | Random |
| 10000 | 9880 | 6165 | 11081 |
| 20000 | 39147 | 24154 | 49867 |
| 40000 | 163778 | 98625 | 211182 |
| 80000 | 521218 | 393854 |  |
| 160000 | - |  |  |

This algorithm does meet all the conditions as the complexity is n squared in all the cases, as it happened here. I had to stop measuring because the times were increasing really fast.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm: | Insertion |  |  |
| N | Sorted | Inverse | Random |
| 10000 | 12 | 1774 | 841 |
| 20000 | 4 | 6822 | 3127 |
| 40000 | 5 | 25836 | 12659 |
| 80000 | 20 | 104728 | 50648 |

Insertion methos is the only one of the three worse one that has a better complexity in the best-case scenario this can be seen in the times shown in the table. And in general it got me the impression that even though the theoretical complexity is the same is a bit faster than the three others.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm: | Selection |  |  |
| N | Sorted | Inverse | Random |
| 10000 | 11490 | 3142 | 1340 |
| 20000 | 45697 | 11504 | 4594 |
| 40000 | 182487 | 45476 | 17653 |

This method meets the expected complexity.

I did not add the quicksort because I faced some issues when trying to measure and I was not able to find the error. Also note that the sorted was done with power saving on so that is why some of them would be faster when inversed like selection sort.

Activity 2. Quicksort Fateful

I think that when you take the leftmost value as pivot it is only a good choice when random values are taken, but the moment the numbers start getting bigger it is more probable that it will choose a bad number. That is why we take the median, as that ensures the pivot will be closer to all the different numbers rather than taking a random number, which is what we do when taking the first element.