In this homework we created four different classes of different ways to sort elements.

Class Heap adds elements to a list already into an order way, so the time that it is required to add would be higher, but the time to get the ‘get\_min’ and ‘get\_max’ would be less because it requires just the time to have access to the element which takes constant time. Instead, Class QuickSort,Class Bubble and Class BinaryTree add elements to a list not in an order way so the operation which takes the most is the get\_max and get\_min since we need to sort the list. However, while the QuickSort and BinaryTree take linear time to get the ‘get\_min’ and ‘get\_max’, Bubble take exponential time so we can guess that it is less efficient.

Class BinaryTree, Class QuickSort and Class Heap running time grows in linear time, while the growth of the running time in the Class Bubble takes exponential time.

Finally, we can state that the most efficient class is the Heap because every time we add an element it, somewhat, sorts the elements in a linear time, but when we search for the min and max it sorts the elements again in a linear time. We can, also, say that the class that takes the most time to look for the min and max, even though it does not take much time for adding, is the BubbleSort; in fact we can see a growth in an exponential time when we compute the max and min, and a growth in linear time for adding elements.

| HEAP SORT | ADD | MIN | MAX |
| --- | --- | --- | --- |
|  | 0.00011388460795084636 | 4.124641418457031e-05 | 4.267692565917969e-05 |
|  | 0.0003715356190999349 | 0.00011873245239257812 | 0.00011452039082845052 |
|  | 0.00034618377685546875 | 0.00013256072998046875 | 0.0001347064971923828 |
|  | 0.0007932186126708984 | 0.0003077189127604167 | 0.0002944469451904297 |

| QUICK SORT | ADD | MIN | MAX |
| --- | --- | --- | --- |
|  | 6.548563639322917e-05 | 0.00024580955505371094 | 0.0002346038818359375 |
|  | 0.0001541773478190104 | 0.0008838176727294922 | 0.000903924306233724 |
|  | 0.0001480579376220703 | 0.001222689946492513 | 0.0012704531351725261 |
|  | 0.00026337305704752606 | 0.0024312337239583335 | 0.0023796558380126953 |

| BINARY TREE | ADD | MIN | MAX |
| --- | --- | --- | --- |
|  | 0.00036406517028808594 | 0.00016760826110839844 | 0.0001819133758544922 |
|  | 0.0010865529378255208 | 0.0005361239115397135 | 0.000370025634765625 |
|  | 0.0037711461385091147 | 0.0012714068094889324 | 0.0012641747792561848 |
|  | 0.006189823150634766 | 0.002784570058186849 | 0.001410086949666341 |

| BUBBLE SORT | ADD | MIN | MAX |
| --- | --- | --- | --- |
|  | 0.04535555839538574 | 4.442532857259115e-05 | 4.8160552978515625e-05 |
|  | 0.42771180470784503 | 0.00021409988403320312 | 0.0001560846964518229 |
|  | 1.4994855721791585 | 0.000383456548055013 | 0.00026671091715494793 |
|  | 3.385268052419027 | 0.0005773703257242838 | 0.00038091341654459637 |





