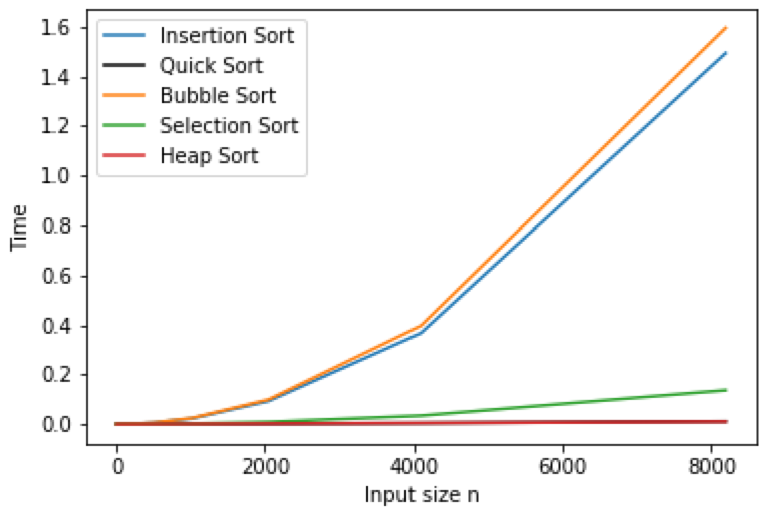
Homework Sorting 1

Plasencia Palacios Milton Nicolás

1. The implementations are in the ”src” folder.

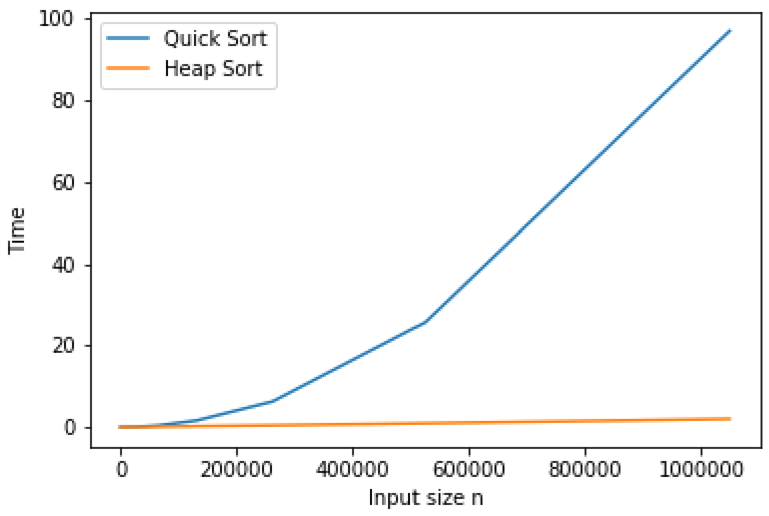
2. For each implemented algorithm, draw a curve to represent the relation between the input size and the execution time.



As expected from the analysis of complexity of each algorithm (shown in the following table):

Algorithm	Insertion Sort	Quick Sort	Bubble Sort	Selection Sort	Heap Sort
Complexity	$\Omega(n)$ - $O(n^2)$	$\Theta(n\log(n))$ - $\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(n\log(n))$

We can see that Insertion Sort and Bubble Sort are the slowest ones. In this range of values of the input size we cannot see the differences between Heap sort and Quick sort so we will take a closer look at them in another graph:



As we can see Heap sort outperforms Quicksort.

3a. Heap sort on a array A whose length is n takes time $O(n)$.
As we have seen the complexity of Heap sort is $O(n \log(n))$ in general. We are allowed to say that this sorting algorithm takes time $O(n)$ only if we are considering a Heap that is build from an array that have all elements (keys) equal, in this case the 'extract min' function takes time $\Theta(1)$ and no more $O(\log n)$ because we do not need to use heapify to restore the heap property every time we remove the min.

3b. Heap sort on a array A whose length is n takes time $\Omega(n)$.
Again, if we are considering a Heap with all elements equal we have that 'extract min' function takes time $\Theta(1)$ and so, in this case, the overall complexity of Heap sort will be $\Theta(n)$, which is by definition both $O(n)$ and $\Omega(n)$.

3c. What is the worst complexity fo Heap sort?
The worst complexity for Heap sort is its upper bound $O(n \log(n))$.

3d. Quick sort on a array A whose length is n takes time $O(n^3)$.
During lectures we have seen that the worst case scenario for Quick sort is when the partition algorithm select as a pivot an element that divide the original array in two sub arrays(B and C) with length $|B| = 0$ and $|C| = |A| - 1$. In this case the complexity is $O(n^2)$ which is a subset of $O(n^3)$ $(O(n^2) \subseteq O(n^3))$.

3e. What is the complexity of Quick sort?
The complexity of Quick sort is, in the worst case scenario $O(n^2)$, in the best

case scenario instead is $\Theta(n\log(n))$.

3f. Bubble sort on a array A whose length is n takes time $\Omega(n)$.
Bubble sort has complexity of $\Theta(n^2)$ but, if we consider an already sorted array we get a complexity of $\Theta(n) = O(n) \cap \Omega(n)$ and so the statement can be considered true in this case.

3g. What is the complexity of Bubble sort?
As we have seen before the overall complexity of Bubble sort is $\Theta(n^2)$.

4. We can solve the equation using recursion tree. We can also observe that:

- Each node has 3 children
- The cost each node is $\Theta(n^{3/2})$ and so we take as representative $n^{3/2}$
- If $n = 32$ we have $T(n) = \Theta(1)$

Thus, we can find out the height of the tree by solving the following equation:

$$\frac{n}{4^i} = 32 \leftrightarrow \frac{n}{32} = 2^{2i} \leftrightarrow \log_2(n) - \log_2(32) = 2i \leftrightarrow i = \frac{log_2(n) - 5}{2} \tag{1}$$

And so the tree has $\frac{log_2(n)-5}{2} + 1 = \frac{log_2(n)-3}{2}$ levels. We obtain that the numbers of nodes in the last level is $3^{\frac{log_2(n)-3}{2}} = n^{\frac{log_2 3}{2}}$.

So the complexity will be:

$$T(n) = \sum_{i=0}^{\frac{log_2(n)-3}{2}} 3^i c\left(\frac{n}{4^i}\right)^{(3/2)} + \Theta(n^{\frac{log_2 3}{2}}) \leq cn^{(3/2)} \sum_{i=0}^{+\infty} \left(\frac{3}{8}\right)^i + \Theta(n^{\frac{log_2 3}{2}}) =$$
$$= cn^{(3/2)} \frac{1}{1-(3/8)} + \Theta(n^{\frac{log_2 3}{2}}) = \frac{8}{5} cn^{(3/2)} + \Theta(n^{\frac{log_2 3}{2}}) \in O(n^{(3/2)})$$