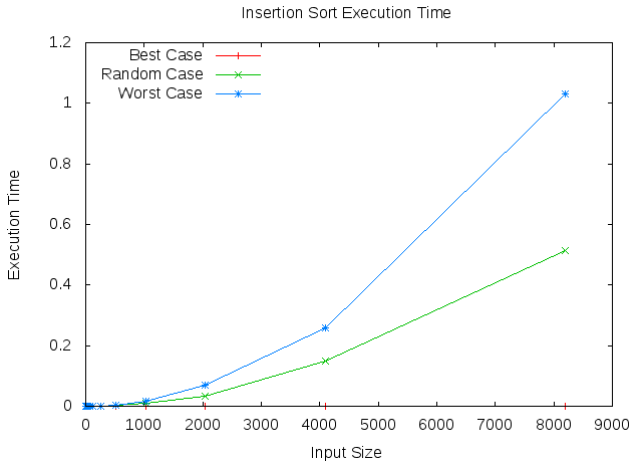
Sorting

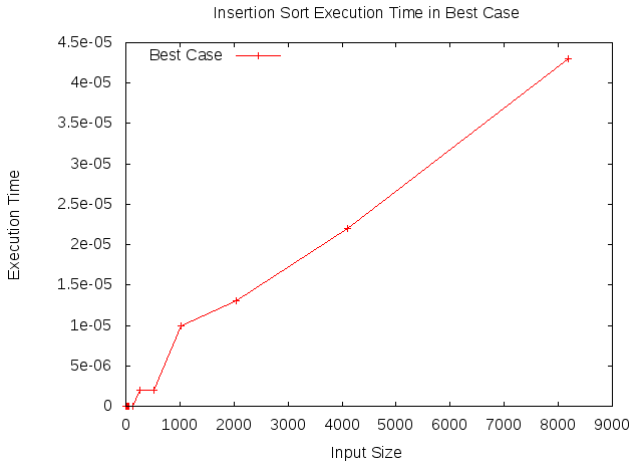
Maria Grazia Berni

Execution Time of Sorting Algorithms

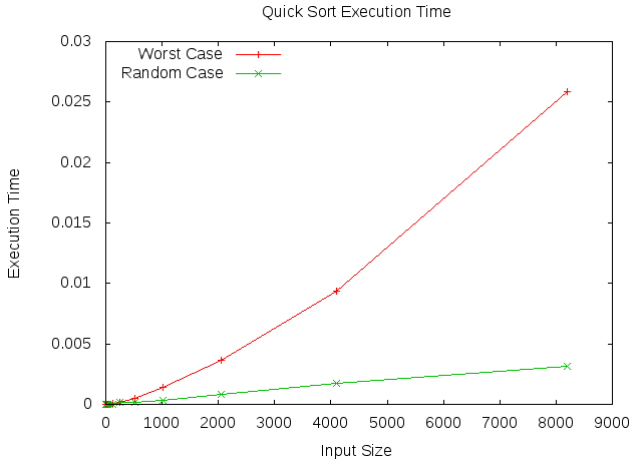
Insertion Sort takes time $\Theta(n^2)$ in a random case and in the worst case, which occurs when the array we want to sort is reverted. This is clear from the following plot:



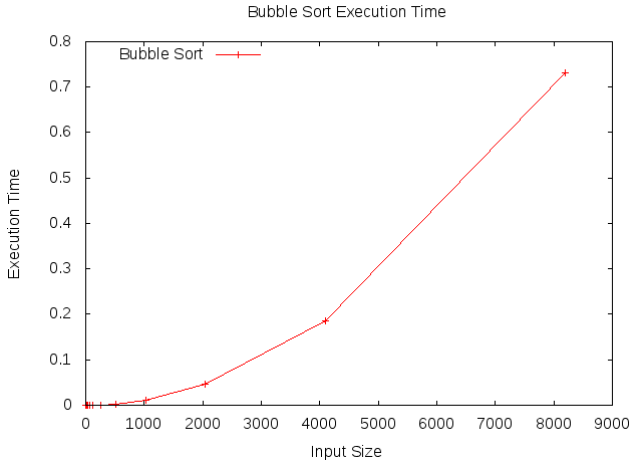
When the array is already sorted, which is the best case possible, it takes linear time:



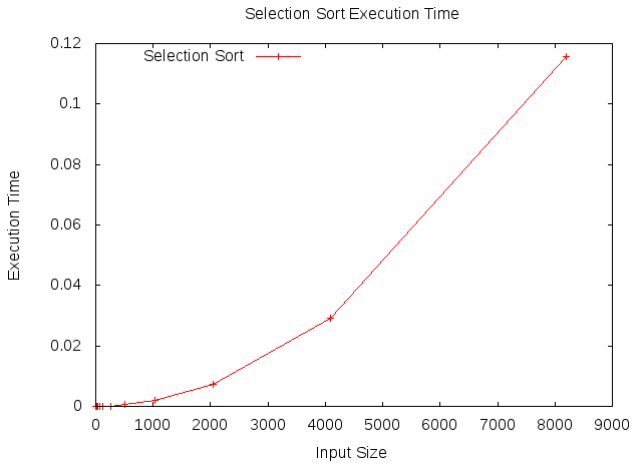
The complexity of quick sort, in case of balanced partition, is $\Theta(n \log n)$, while, in the worst case, is $\Theta(n^2)$.



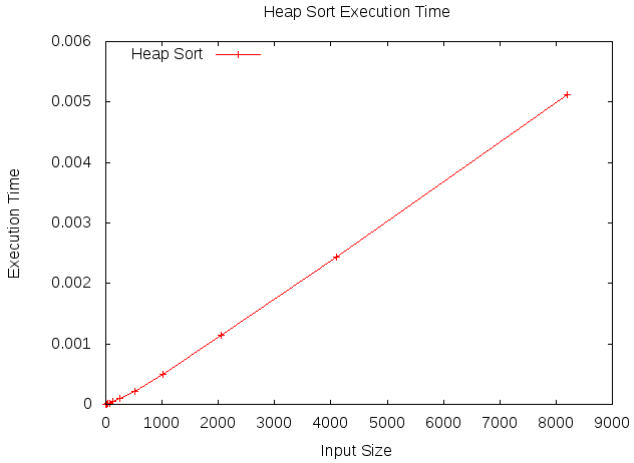
The complexity of Bubble Sort is always $\Theta(n^2)$



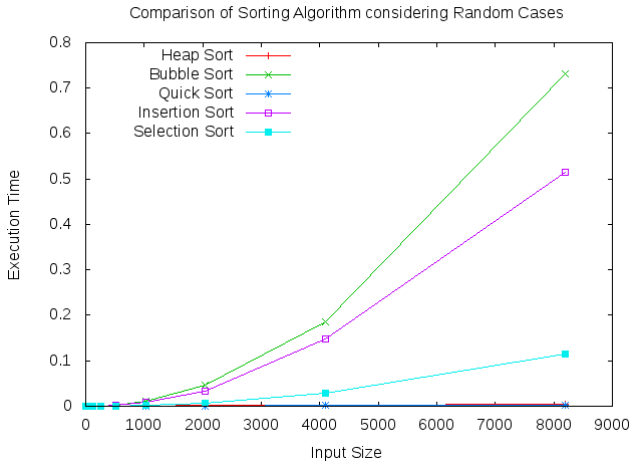
Selection Sort always performs $\mathcal{O}(n^2)$



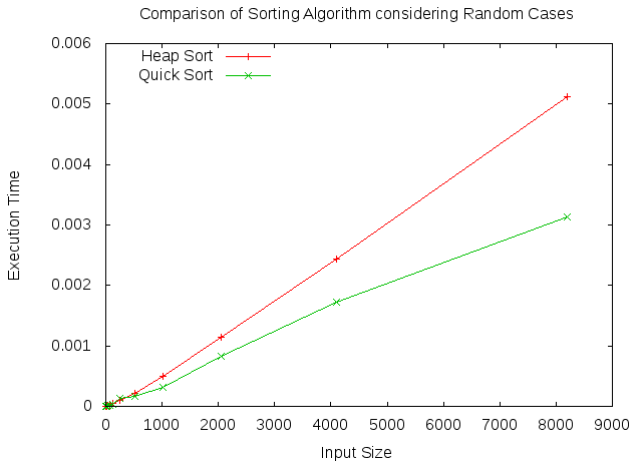
and Heap Sort is always $\mathcal{O}(n \log(n))$



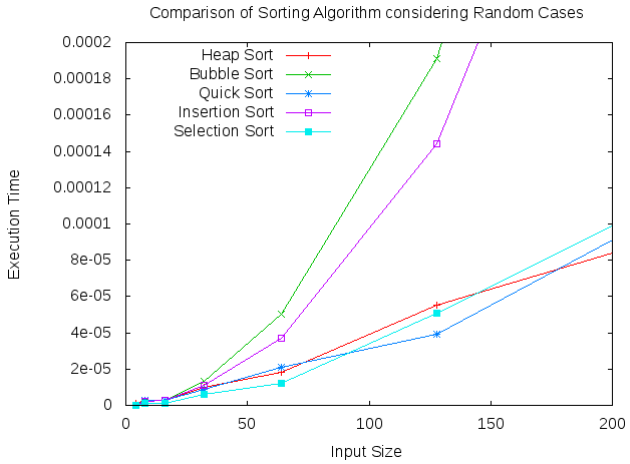
Comparing all the execution time :



quick sort and heap sort outperforms the other sorting algorithm



while, considering small sizes of the input, selection sort is the faster one.



Exercise 3

Argue about the following sentences and answer the questions.

a) Heap Sort on an array A whose length is n takes time $\mathcal{O}(n)$

The complexity of heap sort is $\mathcal{O}(n \log(n))$ in any case, there are no best or worst scenario. Even if the array is already sorted, the extraction of the minimum will put the right-most leaf element in the root, which will be a maximum, and a call to heapify will propagate in many level of the "heap array" (in fact this is actually the worst case) , so that its complexity will be, as usual, $\mathcal{O}(\log(n))$, which is the total complexity of the minimum extraction operation. This operation repeated n times leads to a complexity

$\mathcal{O}(n \log(n))$, so $\mathcal{O}(n)$ is not a proper upper bound for the complexity of heap sort.

b) Heap Sort on an array A whose length is n takes time $\Omega/n)$

The sentence is correct, because $\Omega(n))$ is a proper lower bound for heap sort. In fact the most strict upper bound is $\mathcal{O}(n \log(n))$, but in general it performs better, because after the minimum extraction in general a call to heapify does not propagate in all the level, and even because the number of element in the heap decrease with the extraction. But in any case, it can never perform better then $\Theta/n)$, so it is a $\Omega(n))$.

c) What is the worst case complexity for HEAP SORT?

The worst scenario for heap sort happens when a call to heapify, after the minimum extraction, propagate in all the heap level, and this happen when the array is already sorted. But the complexity is always $\mathcal{O}(n \log(n))$.

d) QUICK SORT on an array whose length is n takes time $\mathcal{O}(n^3)$?

In the worst case quick sort takes time $\Theta(n^2)$, so $\mathcal{O}(n^3)$ it's an upper bound for the complexity, but it's not the most strict one (which is $\mathcal{O}(n^2)$). In any case, the sentence in not wrong.

e) Which is the complexity of QUICK SORT?

The complexity of quick sort, in case of a balanced partition, is $\mathcal{O}(\log(n))$, while, in the worst case, is $\Theta(n^2)$, so for sure, is always a $\mathcal{O}(n^2))$.

f) BUBBLE SORT on an array A whose length is n takes time $\Omega(n)$

The complexity of Bubble Sort is $\Theta(n^2)$, and this doesn't change if the array is already sorted, because in any case it is necessary to execute the two for nested loop. Anyway, is not wrong to say that a lower bound for bubble sort is $\Omega(n)$, but it's not the most strict one, which instead is $\Omega(n^2)$

g) What is the complexity of BUBBLE SORT?

The complexity of bubble sort is $\Theta(n^2)$ in every situation, see point f.

Exercise 4

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 32 \\ 3 \cdot T(n/4) + \Theta(n^{3/2}) & \text{otherwise} \end{cases}$$

TREE ANALYSIS

CASE I: $n = 4^\ell$

LEVEL	DIMENSION	CALL COST	NUMBER OF CALLS	LEVEL COST
0	n	$n^{3/2}$	1	$n^{3/2}$
1	$n/4$	$(n/4)^{3/2}$	3	$3(n/4)^{3/2}$
2	$n/16$	$(n/16)^{3/2}$	3^2	$3^2(n/16)^{3/2}$
...	...	...	...	...
i	$n/4^i$	$(n/4^i)^{3/2}$	3^i	$3^i(n/4^i)^{3/2}$
...	...	...	...	...
$\ell = \log_4 n$	1	$T(1)$	$3^{\log_4 n}$	$3^{\log_4 n}$

$$T(n) = \sum_{i=0}^{\log_4 n - 1} 3^i \left(\frac{n}{4^i}\right)^{3/2} + 3^{\log_4 n}$$

$$= n^{3/2} \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{8}\right)^i + 3^{\log_4 n}$$

$$\leq n^{3/2} \sum_{i=0}^{\infty} \left(\frac{3}{8}\right)^i + 3^{\log_4 n} = n^{3/2} \frac{8}{5} + 3^{\log_4 n}$$

$$\leq \frac{8}{5} n^{3/2} + n^{\log_4 3}$$

$$= O(n^{3/2})$$

CASE $n = 32 \cdot 4^{\ell}$

LEVEL	DIMENSION	CALL COST	NUMBER OF CALLS	LEVEL COST
0	n	$n^{3/2}$	1	$n^{3/2}$
1	$n/4$	$(n/4)^{3/2}$	3	$3(n/4)^{3/2}$
...	...	...	...	...
i	$n/4^i$	$(n/4^i)^{3/2}$	3^i	$3^i(n/4^i)^{3/2}$
...	...	...	...	...
$\ell = \log_4(n/32)$	32	$\Theta(1)$	$3^{\log_4(n/32)}$	$3^{\log_4(n/32)}$

$$T(n) = \sum_{i=0}^{\log_4(n/32)-1} 3^i \left(n/4^i\right)^{3/2} + 3^{\log_4(n/32)}$$

$$\leq n^{3/2} \sum_{i=0}^{\infty} (3/8)^i + 3^{\log_4\left(\frac{n}{32}\right)}$$

$$\leq \frac{8}{5} n^{3/2} + \left(\frac{n}{32}\right)^{\log_4 3}$$

$$= O(n^{3/2})$$