Sorting: Homework

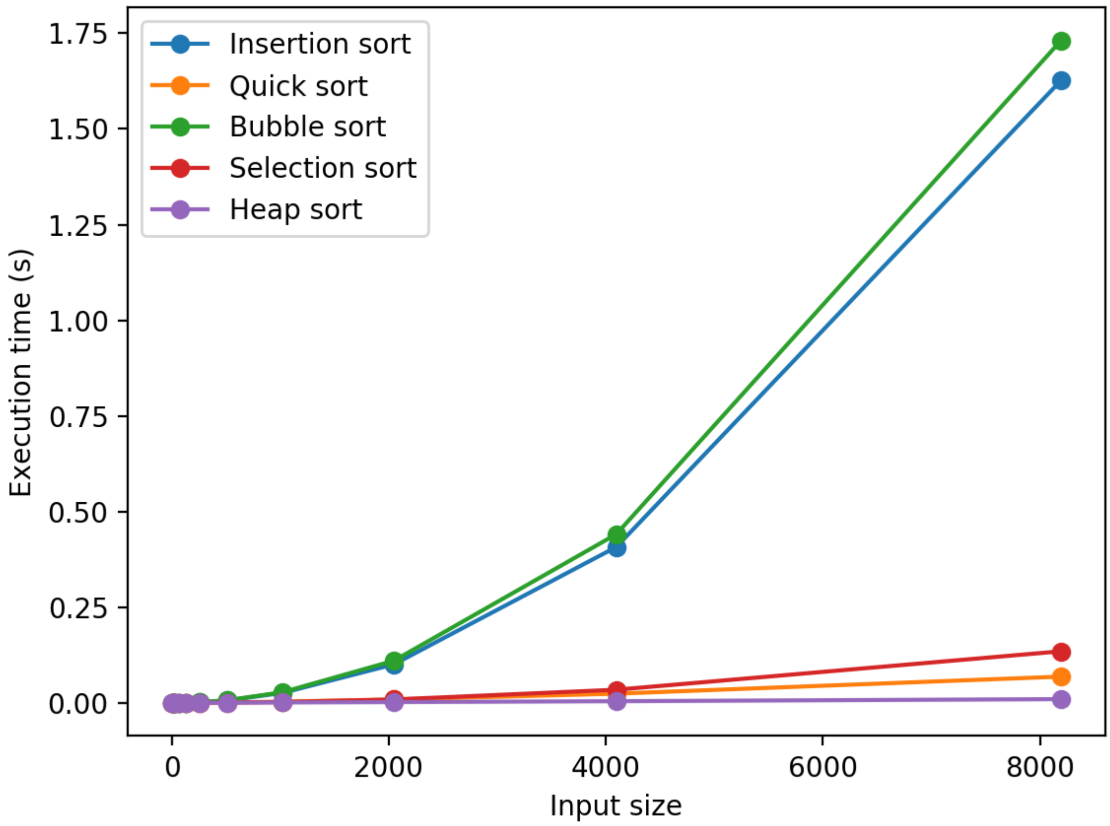
This repository contains the implementation and the testing of `INSERTION SORT`, `QUICK SORT`, `BUBBLE SORT`, `SELECTION SORT`, `HEAP SORT` and `QUICK_SORT-SELECT`.

A Makefile can be produced by using [cmake](https://cmake.org/) as follows, linking the `binheap` library for `HEAP_SORT`:

```
cmake -G "Unix Makefiles" -DBINHEAP_PATH=<BINHEAP_INSTALL_DIR> CMakeLists.txt
```

Exercise 2

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



From this plot we can see that the best sorting algorithms with respect to the input size are `QUICK SORT` and `HEAP SORT`.

Exercise 3

Argue about the following statement and answer the questions:

(a) `HEAP SORT` on an array A whose length is n takes time $O(n)$.

The complexity of `HEAP SORT` is determined by the complexity of `build_heap` which is $\Theta(n)$ and the complexity of `extract_min` which is $O(logn)$. Thus, the overall complexity if $O(nlogn)$ which is a greater upper bound than $O(n)$; thus, in general this statement is not true.

In the case in which `extract_min` takes $\Theta(1)$, the time complexity can be upper bounded by $O(n)$.

(b) `HEAP SORT` on an array A whose length is n takes time $\Omega(n)$.

If `extract_min` takes $\Theta(1)$, then the expected running time is $\Theta(n)$, which is both $\Omega(n)$ and $O(n)$. Thus, the time complexity is lower bounded by $\Omega(n)$ and this statement is true.

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

The worst case complexity for `HEAP SORT` is $O(nlog(n))$ since building the heap takes $\Theta(n)$ (which is upper bounded by $O(n)$) and the overall cost of extracting the minimum from the heap takes $O(nlog(n))$.

(d) `QUICK SORT` on a array A whose length is n takes time $O(n^3)$.

The expected running time for `QUICK SORT` is $\Theta(nlogn)$. Thus, on average, the time complexity is both $O(nlogn)$ and $\Omega(nlogn)$. In the worst case scenario, the running time is $\Theta(n^2)$ and the complexity is both $O(n^2)$ and $\Omega(n^2)$. Since $O(nlogn) \subset O(n^3)$ and $O(n^2) \subset O(n^3)$, this statement is formally true.

(e) What is the complexity of `QUICK SORT`?

The worst-case behavior for quicksort occurs when the partitioning algorithm produces one subproblem with $n-1$ elements and one with 0 elements. In this case, the running time is $\Theta(n^2)$.

In best-case scenario, `partition` produces two subproblems, each of size no more than $\frac{n}{2}$. The recurrence for the running time has solution $\Theta(nlogn)$.

(f) `BUBBLE SORT` on a array A whose length is n takes time Ω(n).

The running time of `BUBBLE SORT` on an array of length n is $\Theta(n^2)$. Thus, the complexity is both $O(n^2)$ and $\Omega(n^2)$. Since $\Omega(n^2) \subset \Omega(n)$, this statement is formally true.

(g) What is the complexity of `BUBBLE SORT`?

In `BUBBLE SORT`, we have to scan the entire array `A` we want to sort and then we swap the maximum element of `A` found until now with the next element if it is less then the current maximum. The time-complexity of the conditional statement is $\Theta(1)$ and thus, if $|A|$ is the length of our array, we have:

$T_{bubble}(|A|) = \sum_{i=1}^{|A|} \sum_{j=1}^{i-1} \Theta(1) = \Theta(|A^2|)$

So the complexity of `BUBBLE SORT` is quadratic.

Exercise 4

Solve the following recursive equation:

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

In order to solve the recursive equation, I built a recursion tree.

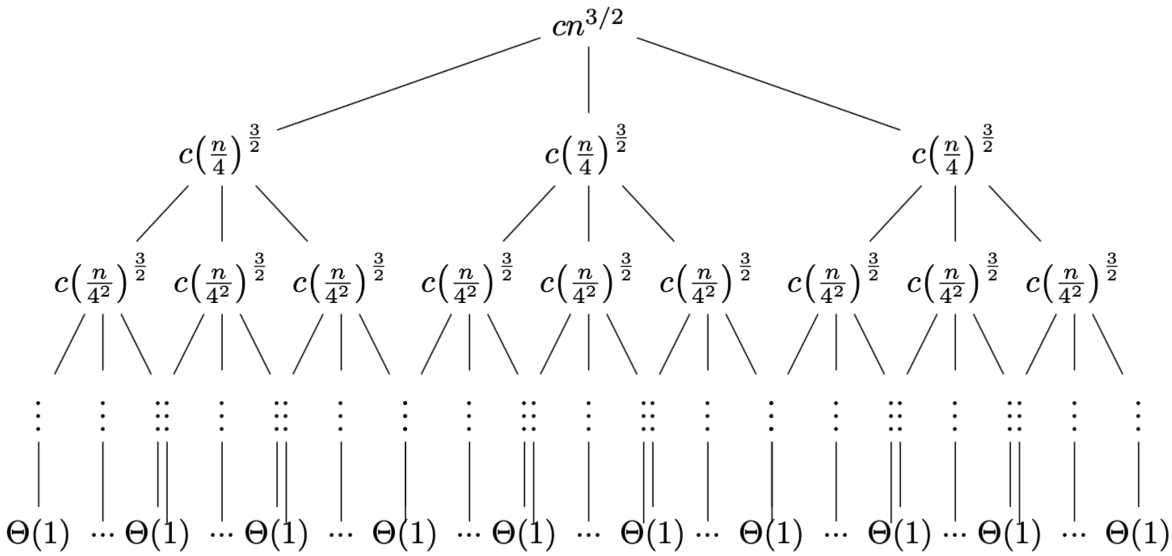


Figure 1: Recursion tree

The root of the tree has cost $cn^{\frac{3}{2}}$, and it has 3 children, each with cost $c(\frac{n}{4})^{\frac{3}{2}}$. Each of these children has 3 children, making 3^2 nodes at depth 2, and each of the 3 children has cost $c(\frac{n}{4^2})^{\frac{3}{2}}$. In general, there are 3^i nodes at depth i, and each has cost $c(\frac{n}{4^i})^{\frac{3}{2}}$. The cost for $n = 32$ is $T(32) = \Theta(1)$ and this is reached at depth $log_4(\frac{n}{32})$, since $\frac{n}{4^{log_4(\frac{n}{32})}} = 32$. There are $3^{log_4(\frac{n}{32})} = \frac{n}{32}^{log_4 3}$ nodes at depth $n = 32$ in the tree.

By summing the costs of the nodes at each depth in the tree we can obtain the given recursive equation:

$$
\begin{aligned}
T(n) &= \Theta(n^{log_4 3}) + cn^{\frac{3}{2}} + \frac{3}{4^{\frac{3}{2}}} cn^{\frac{3}{2}} + \ldots + \left(\frac{3}{4^{\frac{3}{2}}}\right)^{log_4 \frac{n}{32} - 1} cn^{\frac{3}{2}} \\
&= \Theta(n^{log_4 3}) + cn^{\frac{3}{2}} \sum_{i=0}^{log_4 \frac{n}{32} - 1} \left(\frac{3}{8}\right)^i \\
&\leq \Theta(n^{log_4 3}) + cn^{\frac{3}{2}} \sum_{i=0}^{+\infty} \left(\frac{3}{8}\right)^i \\
&= \Theta(n^{log_4 3}) + \frac{8}{5} cn^{\frac{3}{2}} \in O(n^{\frac{3}{2}})
\end{aligned}
$$

Since the cost at the root is $\Theta(n^{\frac{3}{2}})$, the complexity at the root is both $\Omega(n^{\frac{3}{2}})$ and $O(n^{\frac{3}{2}})$ and so $\Omega(n^{\frac{3}{2}})$ must be a lower bound for the recurrence. Since $O(n^{\frac{3}{2}})$ is an upper bound for the recurrence, the overall complexity of the given recursive equation is $\Theta(n^{\frac{3}{2}})$.