Homework 4: Sorting(1)

Azza M. N. Abdalghani

Exercise 1

By using the code at:

`https://github.com/albertocasagrande/AD_sorting`

implement INSERTION SORT, QUICK SORT, BUBBLE SORT, SELECTION SORT, and HEAP SORT.

All algorithms have been implemented inside *AD_sorting* folder, in the *src* folder all required functions have been implemented and in the *include* folder you can find the prototypes of these functions.

Exercise 2

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

For each algorithm, a plot was made to show the relation between its execution time and the input size, as it follows :

- Insertion Sort :

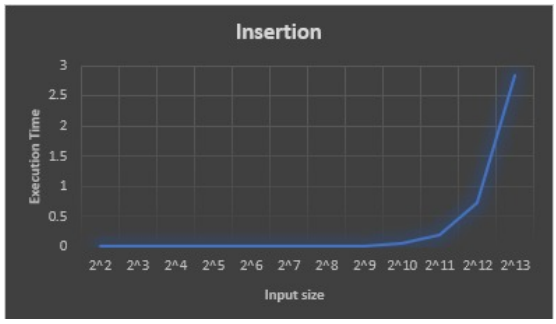

Figure 1: Benchmark of insertion sort algorithm

- Quick Sort :

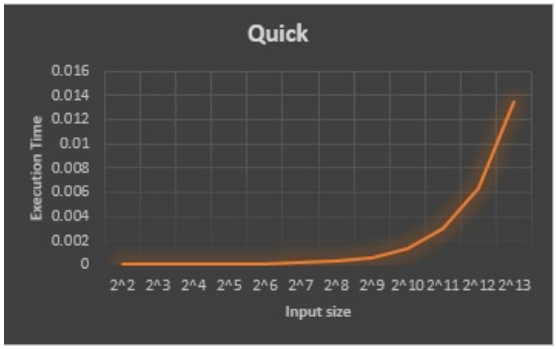

Figure 2: Benchmark of Quick sort algorithm

- Bubble Sort :

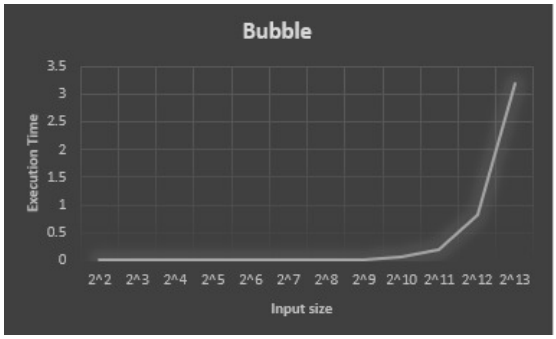

Figure 3: Benchmark of bubble sort algorithm

- Selection Sort :

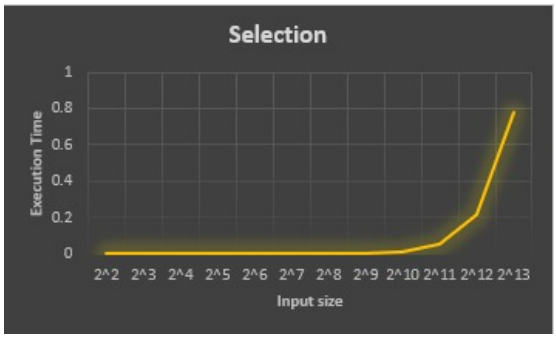

Figure 4: Benchmark of Selection sort algorithm

- Heap Sort :

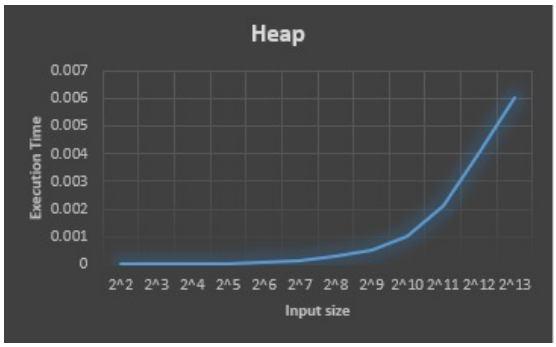

Figure 5: Benchmark of Heap sort algorithm

And to compare all algorithms together on the same plot, this plot shows the difference between each algoritm with respec to the relation between the execution time and the input size.

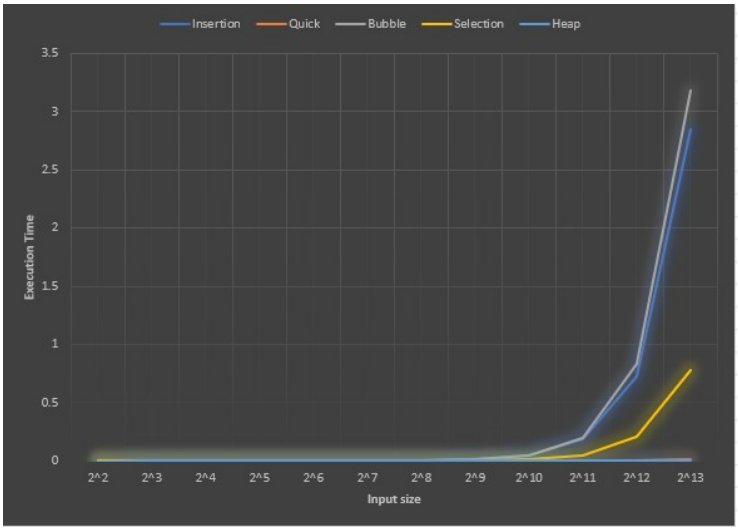

Figure 6: Benchmark of the sorting algorithms

From Figure 6, it is shown that the heap sort algorithm is the faster algoritm compared with other algorithms. However, quick sort is quite the same as heap algorithm, on the other hand the bubble sort is the slowest algorithm.

Exercise 3

Argue about the following statement and answer the questions:

(a) **HEAP SORT on an array A whose length is n takes time $\mathcal{O}(n)$.**

This statement is False, since HEAP SORT basically calls two functions : $BUILD\_HEAP$ (which costs $\Theta(n)$) and $EXTRACT\_MIN$ (which costs $\mathcal{O}(\log i)$ per iteration), Thus in total :

$$\begin{aligned} T_H(n) &= \Theta(n) + \sum_{i=2}^{n} \mathcal{O}(\log i) \\ &\leq \mathcal{O}(n) + \mathcal{O}(\sum_{i=2}^{n} \log n) = \mathcal{O}(n \log n) \end{aligned}$$

So, the overall complexity of HEAP SORT is $\mathcal{O}(n \log n)$, not $\mathcal{O}(n)$.

On the other hand, in the best case scenario of $EXTRACT\_MIN$, it costs $\Theta(1)$ and thus the overall complexity of HEAP SORT becomes $\Theta(n)$, as it can show the upper bound of complexity is $\mathcal{O}(n)$.

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

We have shown in the previous statement that the over all complexity of HEAP SORT in the best case scenario is $\Theta(n)$, which means that the upper bound cost is $\mathcal{O}(n)$ and the lower bound cost is $\Omega(n)$. Thus, this statement is True.

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

Consider calling HEAP SORT on an array which is sorted in decreasing order, every time $A[1]$ is swapped with $A[i]$, MAX-HEAPIFY will be recursively called a number of times equal to the height h of the max-heap containing the elements of positions 1 through $i-1$, and has runtime of $\mathcal{O}(n)$. Since there are 2^k nodes at height k, the runtime is bounded below by :

$$\sum_{i=1}^{\lfloor \log n \rfloor} 2^i \log(2^i) = \sum_{i=1}^{\lfloor \log n \rfloor} i2^i = 2 + (\lfloor \log n \rfloor - 1)2^{\lfloor \log n \rfloor} = \Omega(n \log n)$$

Thus, the worst case complexity of HEAP SORT is $\Omega(n \log n)$.

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

QUICK SORT costs on average and on best cases $\Theta(n \log n)$ for an array with length of n, which means that the upper bound complexity is $\mathcal{O}(n \log n)$. Also, since $\mathcal{O}(n \log n)$ is a subset of $O(n^3)$, we can say that QUICK SORT costs $\mathcal{O}(n^3)$ on average.

Moreover, for the worst case scenario, QUICK SORT costs $\Theta(n^2)$, which is also equal to say that the upper bound cost is $\mathcal{O}(n^2)$ and the lower bound cost is $\Omega(n^2)$. Thus, it is clear that $\mathcal{O}(n^2)$ is a subset of $\mathcal{O}(n^3)$, so we can say that QUICK SORT costs $\mathcal{O}(n^3)$ in worst case scenario and the statement holds.

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

The complexity of QUICK SORT in average case is $\Theta(n \log_2 n)$ and so as in the best case scenario (which is for balanced pertition). However, the complexity of QUICK SORT in the worst case is $\Theta(n^2)$.

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

As it was shown during the lecture, the overall complexity of BUBBLE SORT is $\Theta(n^2)$ for an array A of length n. Thus, we can say that the statement is False since $\Omega(n^2)$ is not a subset of $\Omega(n)$.

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

As it has mentioned in the previous statement, the overall complexity of BUBBLE SORT is $\Theta(n^2)$ for an array A of length n.

Exercise 4

Solve the following recursive equation:

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

We can use recursive tree method to solve it, firstly we need to choose a representation for $\Theta(n^{3/2})$, which can be $cn^{3/2}$. Then, we can see from the equation that there are 3 recusrive calls, each call considers a quarter of the given n size, So we can build the tree as follows :

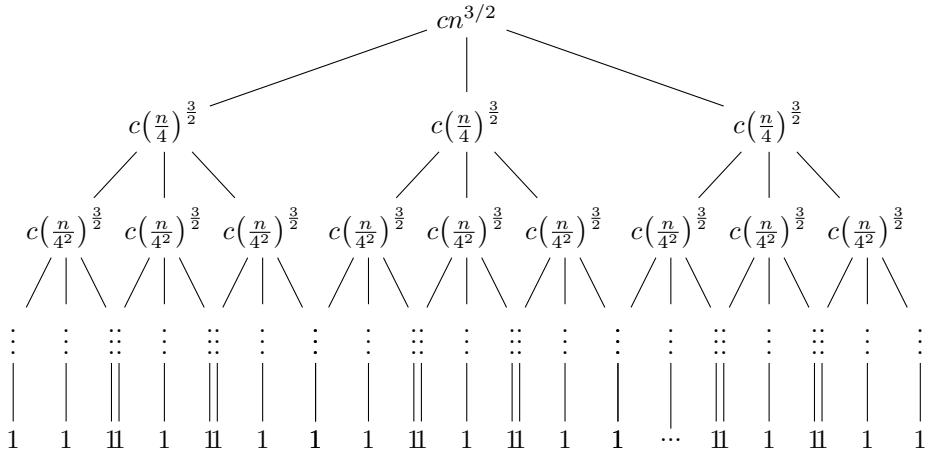

Figure 7: Recursion tree for T(n)

From the tree, we can see that at level **i** we have 3^i nodes, and the total cost at level **i** is $3^i.c.(\frac{n}{4^i})^3/2$. Also, the length of the tree can be found by $(\frac{n}{4^i}) = 32$, where $n = 32$ is the base case that costs $\Theta(1)$:

$$(\frac{n}{4^i}) = 32 \Rightarrow i = \log_4(\frac{n}{32})$$

Thus, we can write the equation by summing total cost at each level as follows :

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

Thus, $T(n) \in \mathcal{O}(n^{3/2})$.