Sorting: Homework 2

1. *Generalize the* `SELECT` *algorithm to deal also with repeated values and prove that it still belongs to* $O(n)$.
2. *Download the latest version of the code from*

   https://github.com/albertocasagrande/AD_sorting

   and
   - *Implement the* `SELECT` *algorithm of Ex. 1.*
   - *Implement a variant of the* `QUICK SORT` *algorithm using above-mentioned* `SELECT` *to identify the best pivot for partitioning.*
   - *Draw a curve to represent the relation between the input size and the execution-time of the two variants of* `QUICK SORT` *(i.e, those of Ex. 2 and Ex. 1 of this file) and discuss about their complexities.*
3. *(Ex. 9.3-1 in [1]) In the algorithm* `SELECT`*, the input elements are divided into chunks of* 5*. Will the algorithm work in linear time if they are divided into chunks of* 7*? What about chunks of* 3*?*
4. *(Ex. 9.3-5 in [1] Suppose that you have a "black-box" worst-case linear-time subroutine to get the position in* A *of the value that would be in position* $n/2$ *if* A *was sorted. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary position* i*.*
5. *Solve the following recursive equations by using both the recursion tree and the substitution method:*
   - $T_1(n) = 2 * T_1(n/2) + O(n)$

     Using the **recursion tree**, we have that, choosing cn as representative for $O(n)$:

$$\begin{aligned} T_1(n) &= 2 * T_1(n/2) + O(n) \\ &\leq 2 * T_1(n/2) + cn \\ &\leq 2 * (2 * T_1(n/4) + cn/2) + cn \\ &= 4 * T_1(n/4) + 2cn \\ &\leq 4 * (2 * T_1(n/8) + cn/4) + 2cn \\ &= 8 * T_1(n/8) + 3cn \\ &\leq \dots \\ &\leq 2^{\log_2 n} T_1(0) + \log_2 n * cn \\ &= n * 0 + cn \log_2 n \in O(n \log n) \end{aligned}$$

     or alternatively

$$T_1(n) = 2 * T_1(n/2) + O(n) \leq \sum_{i=0}^{\log_2 n} 2^i c \frac{n}{2^i} = \sum_{i=0}^{\log_2 n} cn = cn \sum_{i=0}^{\log_2 n} 1 = cn \log_2 n \in O(n \log n)$$

Using the **substitution method**, we guess that $T_1(n) \in O(n \log n)$. Select a representative for $O(n \log n)$ and $O(n)$, e.g. $cn \log n$ and $c'n$. Assume that $\forall m < n,\ T_1(n) \leq cm \log m$. If this is the case,

$$\begin{aligned} T_1(n) &= 2 * T_1(n/2) + c'n \\ &\leq 2 * cn/2 \log(n/2) + c'n \\ &\leq cn(\log n - \log 2) + c'n \\ &\leq cn \log n - cn \log 2 + c'n \end{aligned}$$

if $c'n - cn \log n \leq 0$, then $c'n \leq cn \log n$ so $c \geq \frac{c'}{\log n}$ then $T_1(n) \leq cn \log n$. Thus we have proved by induction that $\forall n \in \mathbb{N},\ T_1(n) \leq cn \log n$ for a proper c. So $T_1(n) \in O(n \log n)$.

- $T_2(n) = T_2(\lceil n/2 \rceil) + T_2(\lfloor n/2 \rfloor) + \Theta(1)$

  Using the **recursion tree**, we have that, choosing c as representative for $O(1)$:

$$\begin{aligned} T_2(n) &= T_2(\lceil n/2 \rceil) + T_2(\lfloor n/2 \rfloor) + \Theta(1) \\ &= T_2(\lceil n/2 \rceil) + T_2(\lfloor n/2 \rfloor) + c \\ &= \end{aligned}$$

- $T_3(n) = 3 * T_3(n/2) + O(n)$
- $T_4(n) = 7 * T_4(n/2) + \Theta(n^2)$

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press. MIT Press, 2009.