# Homework 05: Sorting (Part 2)

Roberto Corti

July 11, 2020

## Exercise 1

**Generalize the SELECT algorithm to deal also with repeated values and prove that it still belongs to $O(n)$.**

## Exercise 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 31/3/2020) and discuss about their complexities.**

The new version of SELECT that deals with repeated values described in the previous section is implemented inside the file `select.c` and `quick_sort.c` for the tri-partition implementation.
Once applied this algorithm, I implemented a variant of QUICK SORT using the SELECT method in order to choose the best pivot:

```
Quick-Sort Select(A, left, right):
  while left ≤ right do
     pivot ← select_pivot(A, left, right)
     Quick-Sort Select(A, left, k.first)
     left ← k.second+1
  end while
```

The results, compared with the classic version of QUICK SORT, are represented in the following plot.

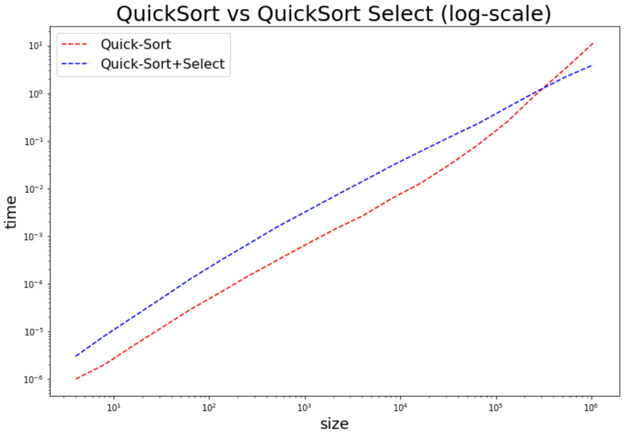


**Figure 1:** Benchmark of the `quick_sort` method compared to its version when the `select` algorithm is involved in the selection of the pivot.

As it can be observed, the classic QUICK SORT has better performances on small list sizes. However, for big arrays the cost of partitioning becomes relevant and in those cases QUICK-SORT SELECT results to be more efficient.

# Exercise 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?**

In the case of having divided into chunks of 7, as we did at lesson, we can get a lower bound on the number of elements that are greater than the median-of-median (say $\hat{m}$), that is

$$4\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{7} \right\rceil \right\rceil - 2\right) \geq \frac{2n}{7} - 8.$$

Then, the upper bound for the number of elements smaller or equal to $\hat{m}$ is

$$n - \left(\frac{2n}{7} - 8\right) = \frac{5n}{7} + 8,$$

and this leads to the following recurrence:

$$T_S(n) = T_S(\lceil n/7 \rceil) + T_S(5n/7 + 8) + \Theta(n).$$

Selecting $cn$ and $c'n$ as representatives of $O(n)$ and $\Theta(n)$ and assuming that $T_S(m) \leq cm\ \forall m < n$

$$\begin{aligned}
T(n) &\leq c\lceil n/7\rceil + c(5n/7 + 8) + c'n \\
&\leq cn/7 + c + 5cn/7 + 8c + c'n \\
&= 6cn/7 + 9c + c'n \\
&= cn + (-cn/7 + 9c + c'n).
\end{aligned}$$

which is at most $cn$ if $(-cn/7 + 9c + c'n) \leq 0$. This is equivalent to $c \geq \frac{7c'n}{n-63}$ and picking $n \geq 126$ and $c \geq 14c'$, we get $T_S(n) \leq cn$ and $T_S(n) \in O(n)$.

If we divide in groups of 3, the number of elements that are greater than the median-of-medians $\hat{m}$ is:

$$2\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{3}\right\rceil \right\rceil - 2\right) \geq \frac{n}{3} - 4,$$

so the upper bound for the number of elements smaller or equal to $\hat{m}$ is

$$n - \left(\frac{n}{3} - 4\right) = \frac{2n}{3} + 4.$$

The recurrence is thus

$$T(n) = T(\lceil n/3\rceil) + T(2n/3 + 4) + \Theta(n).$$

We guess that $T(n) > cn$ and bounding the non-recursive term with $c'n$:

$$\begin{aligned}
T(n) &> c\lceil n/3\rceil + c(2n/3 + 2) + an \\
&> cn/3 + c + 2cn/3 + 2c + an \\
&= cn + 3c + an \\
&> cn,
\end{aligned}$$

which holds for any $c > 0$.

# Exercise 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.**

Let $A$ be the array and denote by $i$ the arbitrary position. The black-box subroutine if applied to $A$ returns the $n/2$ element. If $i = n/2$ the problem is solved. Otherwise, we can partition $A$ into $A_1$, $A_2$ those elements are respectively lower and greater than $A[n/2]$. If $i < n/2$, we will call recursively the algorithm on $A_1$ by searching its $i$th element. If $i > n/2$, the recursive call will be done on $A_2$ by looking for the element $i - n/2$.

```
SELECTION(A,i):
  BLACK-BOX(A)
  if i=n/2 return A[n/2]
  DIVIDE(A)
  if i<n/2 return SELECTION(A1,i)
  else return SELECTION(A2,i-n/2)
```

The cost of using the black-box subroutine is $O(n)$ and the cost of dividing the array is $O(n)$ too. Let $T(n)$ be the cost of the algorithm described above. Then,

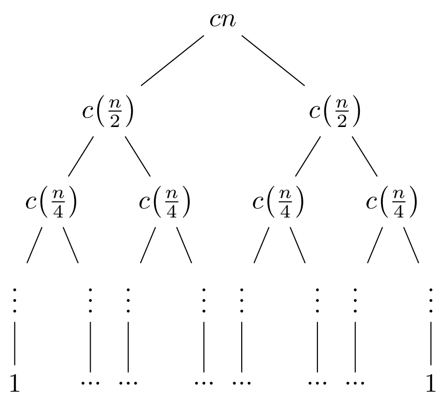
$$\begin{aligned} T(n) &\leq cn + T(n/2) \\ &= c(n + n/2 + n/4 + ... + T(1)) \\ &\leq 2cn \in O(n). \end{aligned}$$

# Exercise 5

**Solve the following recursive equations by using both the recursion tree and the substitution method:**

1. $T_1(n) = 2 \cdot T_1(n/2) + O(n)$
2. $T_2(n) = 2 \cdot T_2(\lceil n/2 \rceil) + T_2(\lfloor n/2 \rfloor) + \Theta(1)$
3. $T_3(n) = 3 \cdot T_3(n/2) + O(n)$
4. $T_4(n) = 7 \cdot T_4(n/2) + \Theta(n^2)$

1. *Recursion Tree*:
The recurrent equation $T_1(n) = 2 \cdot T_1(n/2) + O(n)$ gives rise to the following recursive tree


$cn$
$c\left(\frac{n}{2}\right)$
$c\left(\frac{n}{2}\right)$
$c\left(\frac{n}{4}\right)$
$c\left(\frac{n}{4}\right)$
$c\left(\frac{n}{4}\right)$
$c\left(\frac{n}{4}\right)$
1
1

Summing up all the contribution given at each tree level we end up with

$$
\begin{aligned}
T_1(n) &= \sum_{i=0}^{\log_2 n - 1} cn + \Theta(n), \\
&= cn \log_2 n + \Theta(n) \in O(n \log_2 n),
\end{aligned}
$$

thus $T_1(n) \in O(n)$ $\square$

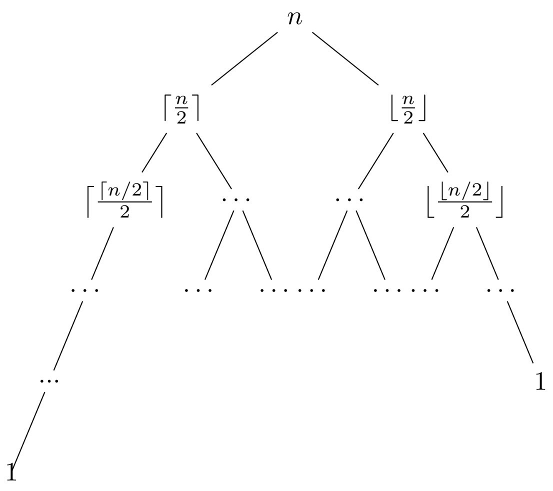
*Substitution Method*:

- **Guess**: $T_1(n) \in O(n \log_2 n)$,
- **Inductive hypothesis**: $\forall m < n,\ T_1(m) \leq cm \log_2 m$

In order to prove the guess for $n$ consider $c'n$ as a representative of $O(n)$. Then, considering the equation and using the inductive hypothesis,

$$
\begin{aligned}
T_1(n) &= 2T_1(n/2) + c'n, \\
&\leq 2c\frac{n}{2} \log_2 \frac{n}{2} + c'n, \\
&\leq cn \log_2 n - cn + c'n \leq cn \log_2 n \Leftrightarrow c \geq c' \quad \square
\end{aligned}
$$

2.*Recursion Tree*
The recurrent equation $T_2(n) = 2 \cdot T_2\left(\lceil n/2 \rceil\right) + T_2\left(\lfloor n/2 \rfloor\right) + \Theta(1)$ give rise to an unbalanced tree in which the left-most branch involves only ceiling operation, while the right-most branch will involve only floor operations.


$n$
$\lceil \frac{n}{2} \rceil$
$\lfloor \frac{n}{2} \rfloor$
$\lceil \frac{\lceil n/2 \rceil}{2} \rceil$
$\lfloor \frac{\lfloor n/2 \rfloor}{2} \rfloor$
1
1

The height of the left most branch is at most $\log(2n)$ since $\forall n$ there exists a power of two between the interval $[n, 2n]$ and in case of $n$ equal to this power we have the maximum height of this branch. For what concerns the right most branch, we look for the closer power of two before $n$ and so the height of this branch will be for sure greater than $\log(n/2)$. Thus, considering $c$ as a representative of $\Theta(1)$ and given these two results we can obtain

$$\begin{aligned} T_2(n) &\geq \sum_{i=0}^{\log(n/2)} c2^i, \\ &\geq c\frac{2^{\log(n/2)+1} - 1}{2-1}, \\ &\geq cn - c \Rightarrow T_2(n) \in \Omega(n), \end{aligned}$$

and, at the same time,

$$\begin{aligned} T_2(n) &\leq \sum_{i=0}^{\log(2n)} c2^i, \\ &\leq c(2^{\log 2n+1} - 1), \\ &\leq 4cn - c \Rightarrow T_2(n) \in O(n). \end{aligned}$$

Then, putting these two results together, $T_2(n) \in \Theta(n)$ $\square$

*Substitution Method*
In this particular case our goal is to prove that $T_2(n) \in \Theta(n)$ and in order to achieve this thesis we will demonstrate that $T_2(n) \in O(n)$ and, at the same time, $T_2(n) \in \Omega(n)$.

- **Guess**: $T_2(n) \in O(n)$,
- **Inductive hypothesis**: $\forall m < n,\ T_2(m) \leq cm - d$

In order to prove the guess consider $c'$ as a representative of $\Theta(1)$. Then, considering the recurrent equation and the inductive hypothesis,

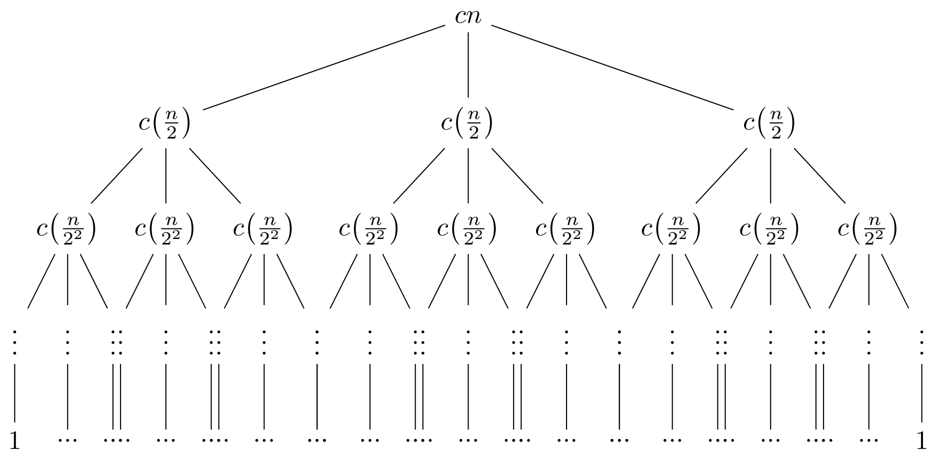
$$\begin{array}{rcl} T_2(n) & = & T_2\left(\lceil n/2 \rceil\right) + T_2\left(\lfloor n/2 \rfloor\right) + c', \\ & \leq & c\lceil n/2 \rceil - d + \rfloor n/2 \rfloor - d + c' \\ & \leq & cn - d - d + c' \leq cn - d \Leftrightarrow d \geq c' \quad \Box \end{array}$$

- **Guess**: $T_2(n) \in \Omega(n)$,
- **Inductive hypothesis**: $\forall m < n,\ T_2(m) \geq cm$

Also in this case considering $c'$ as a representative of $\Theta(1)$, the recurrent equation and the inductive hypothesis,

$$\begin{array}{rcl} T_2(n) & = & T_2\left(\lceil n/2 \rceil\right) + T_2\left(\lfloor n/2 \rfloor\right) + c', \\ & \geq & cn + c' \geq cn \quad \Box \end{array}$$

3.*Recursion Tree*
The recurrent equation $T_3(n) = 3 \cdot T_3(n/2) + O(n)$ gives rise to the following recursive tree



$cn$
$c\left(\frac{n}{2}\right)$
$c\left(\frac{n}{2}\right)$
$c\left(\frac{n}{2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
$c\left(\frac{n}{2^2}\right)$
1
1

Summing up all the contribution given at each tree level we end up with

$$\begin{aligned}
T_3(n) &= cn + \frac{3}{2}cn + ... + \left(\frac{3}{2}\right)^{\log_2 n - 1} cn + \Theta(n^{\log_2 3}), \\
&= cn \sum_{i=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^i + \Theta(n^{\log_2 3}), \\
&\leq \frac{1}{1 - 3/2} cn + \Theta(n^{\log_2 3})
\end{aligned}$$

Thus, taking $c'n^{\log_2 3}$ as a representative of $\Theta(n^{\log_2 3})$, it holds

$$T_3(n) \leq c'n^{\log_2 3} \Rightarrow T_3(n) \in O(n^{\log_2 3}) \quad \square$$

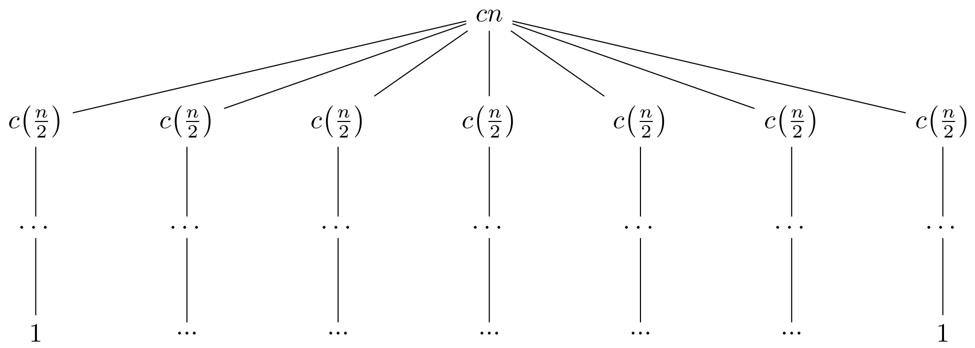
*Substitution Method*:

- **Guess**: $T_3(n) \in O(n^{\log_2 3})$,
- **Inductive hypothesis**: $\forall m < n,\ T_3(m) \leq cm^{\log_2 3} - dm$

In order to prove the guess for $n$ consider $c'n$ as a representative of $O(n)$. Then, considering the equation and using the inductive hypothesis,

$$\begin{aligned}
T_3(n) &= 3T_3(n/2) + c'n, \\
&\leq 3c(n/2)^{\log_2 3} - \frac{3}{2}dn + c'n, \\
&\leq cn^{\log_2 3} - dn - \frac{1}{2}dn + c'n \leq cn^{\log_2 3} \Leftrightarrow d \geq 2c' \quad \square
\end{aligned}$$

4.*Recursion Tree*
The recurrent equation $T_4(n) = 7 \cdot T_4(n/2) + \Theta(n^2)$ gives rise to the following recursive tree



Summing up all the contribution given at each tree level we end up with

$$\begin{aligned} T_3(n) &= cn^2 + \frac{7}{4}cn + ... + \left(\frac{7}{4}\right)^{\log_2 n-1} cn + \Theta(n^{\log_2 7}), \\ &= cn^2 \sum_{i=0}^{\log_2 n-1} \left(\frac{7}{4}\right)^i + \Theta(n^{\log_2 7}), \\ &\leq \frac{1}{1-7/4}cn^2 + \Theta(n^{\log_2 7}). \end{aligned}$$

Thus, taking $c'n^{\log_2 7}$ as a representative of $\Theta(n^{\log_2 7})$, it holds

$$T_3(n) \leq c'n^{\log_2 7} \Rightarrow T_3(n) \in O(n^{\log_2 3}) \quad \square$$

*Substitution Method*:

- **Guess**: $T_4(n) \in O(n^{\log_2 7})$,
- **Inductive hypothesis**: $\forall m < n,\ T_4(m) \leq cm^{\log_2 7} - dm^2$

In order to prove the guess for $n$ consider $c'n^2$ as a representative of $\Theta(n^2)$. Then, considering the equation and using the inductive hypothesis,

$$\begin{aligned} T_4(n) &= 7T_4(n/2) + c'n^2, \\ &\leq 7c(n/2)^{\log_2 7} - \frac{7}{4}d + c'n^2, \\ &\leq cn^{\log_2 7} - dn^2 - \frac{3}{4}dn^2 + c'n^2 \leq cn^{\log_2 7} - dn^2 \Leftrightarrow d \geq \frac{4}{3}c' \quad \square \end{aligned}$$