Sorting Homework 2

Exercise 1

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

Solution

The implementation of the SELECT algorithm that we implemented during the lectures is based on the assumption that all the values of the array are distinct. In order to deal with repeated values in the array we have to modify a bit the algorithm.

A possible solution is using the 3-way-partitioning. Given a pivot k, instead of dividing the original array A into two arrays S, containing all the elements $< k$, and G, containing all the elements $> k$, we could divide A into three sub-arrays. Specifically we will obtain S, containing all the elements $< k$, G, containing all the elements $> k$, and E, containing all the elements $= k$.

```
def threewaypartition(A, i, j, p):

    swap(A, i, p) // swap the pivot p and the left-most element in A
    i++
    k = i

    while(k <= j):
        if(A[k] < A[p]):
            swap(A, i, k)
            i++
            k++
        else if(A[k] > A[p])
            swap(A, k, j)
            j--
         else
            k++
    swap(A, p, i-1)
    return (i-1, j+1)
```

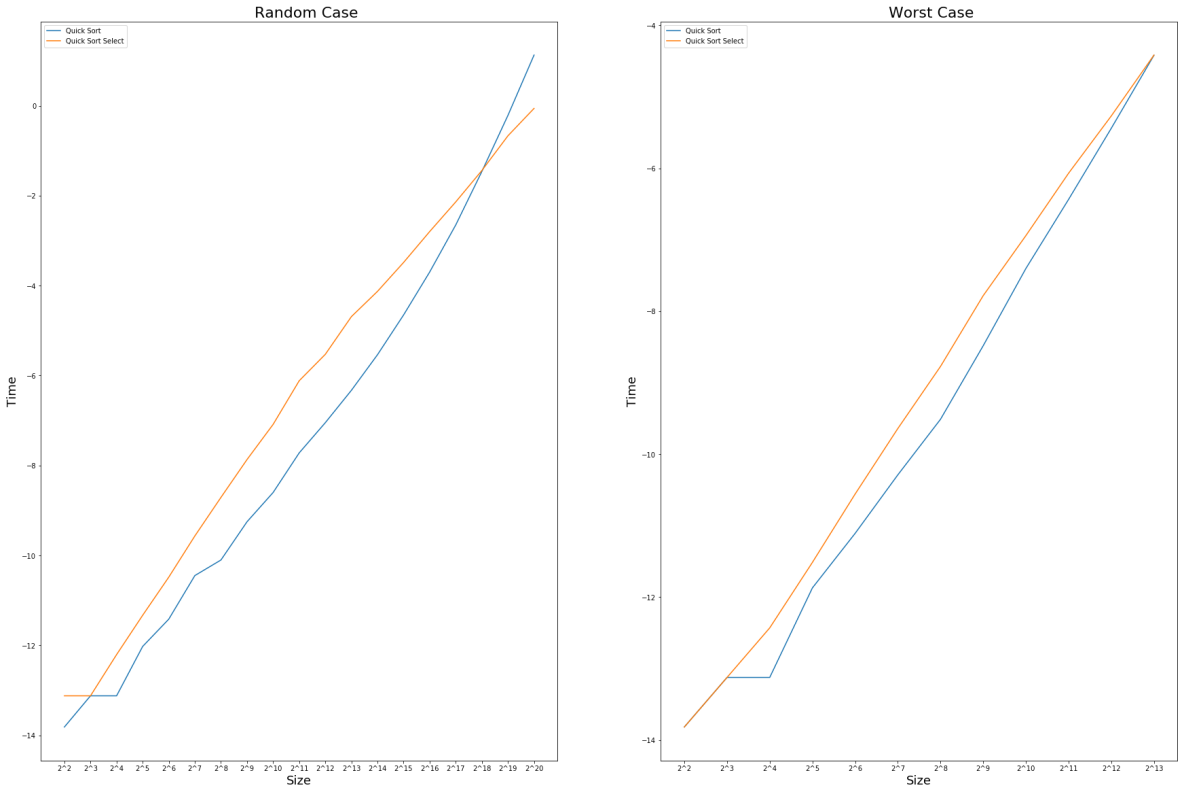
It's easy to see that the complexity of this algorithm is the same with respect to the complexity of the partition algorithm introduced before. The complexity of the swap does not change, the number of iterations of the while-loop is $i - j$ since one index is incremented at each repetition. Hence the complexity is still $O(n)$.

Exercise 2

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.

Solution

In the following plots we can see the logarithm transformation of the execution time of the two variants of the Quick Sort algorithm, with and without the select. As we can see the performances are very similar. In fact, as we know both the algorithms have the same complexity.



Exercise 3 (Ex. 9.3-1)

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?

Solution

Let's try to divide the input into chunks of 7 $C_1, \ldots, C_{\frac{n}{7}}$, find the median m_i of each C_i and the median m of the m_i.

$\lceil \frac{n}{7} \rceil$: number of chunks

$\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil$: number of $m_i \geq m$

$\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2$: number of chunks that have at least 4 elements $> m$

$4(\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2)$: number of elements (at least) that are $> m$

$\implies 4(\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2) \geq \frac{4n}{14} - 8$

The upper bound for the number of elements $\leq m$ is $n - (\frac{4n}{14} - 8) = \frac{5n}{7} + 8$.

Hence the complexity is $T_S(n) = T_S(\lceil \frac{n}{7} \rceil) + T_S(\frac{5n}{7} + 8) + \Theta(n)$

Select cn and $c'n$ as representatives of $O(n)$ and $\Theta(n)$ and assume $T_S(m) \leq cm \forall m < n$.

$$
\begin{aligned}
T_S(n) &\leq c\lceil\frac{n}{7}\rceil + c(\frac{5n}{7}+8) + c'n \\
&\leq c(\frac{n}{7}+1) + c(\frac{5n}{7}+8) + c'n \\
&\leq \frac{6}{7}cn + c'n + 9c
\end{aligned}
$$

Hence $T_S(n) \leq cn \iff$

$$
\begin{aligned}
\frac{6}{7}cn + c'n + 9c &\leq cn \\
9c + c'n &\leq \frac{cn}{7} \\
c'n &\leq c(\frac{n}{7} - 9) \\
c &\geq \frac{c'n}{\frac{n}{7} - 9} = \frac{7c'n}{n-63}
\end{aligned}
$$

So if $n \geq 63$ and $c \geq \frac{7c'n}{n-63}$ then $T_S(n) \in \Theta(n)$

Let's now consider chunks of 3. The upper bound for the number of elements $\leq m$ is $n - (\frac{2n}{6} - 4) = \frac{4n}{6} + 4$

Select cn and $c'n$ as representatives of $O(n)$ and $\Theta(n)$ and assume $T_S(m) \leq cm \forall m < n$.

$$
\begin{aligned}
T_S(n) &\leq c\lceil\frac{n}{3}\rceil + c(\frac{4n}{6}+4) + c'n \\
&\leq c(\frac{n}{3}+1) + c(\frac{4n}{6}+4) + c'n \\
&\leq cn + c'n + 5c
\end{aligned}
$$

Hence $T_S(n) \leq cn \iff$

$$
\begin{aligned}
cn + c'n + 5c &\leq cn \\
5c + c'n &\leq 0 \\
c'n &\leq -5c \\
c &\leq \frac{c'n}{5}
\end{aligned}
$$

So if $c \leq \frac{c'n}{5}$ then $T_S(n) \in \Theta(n)$

Exercise 4 (Ex. 9.3-5)

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 $\frac{n}{2}$ if A was sorted. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary position i.

Solution

Thanks to the "black-box" we are able to identify the median of A in linear-time. After having identified the median we can partition A taking the median as pivot in order to have all the elements smaller then the median in the first half of A and all the elements bigger then the median in the second half of A. The we can apply again the "black-box" on the half of A in which the element we are searching for is contained, partition that half taking the median as pivot an so on.

If we chose an algorithm that partition the array in linear time, the procedure will take time $O(n)$.

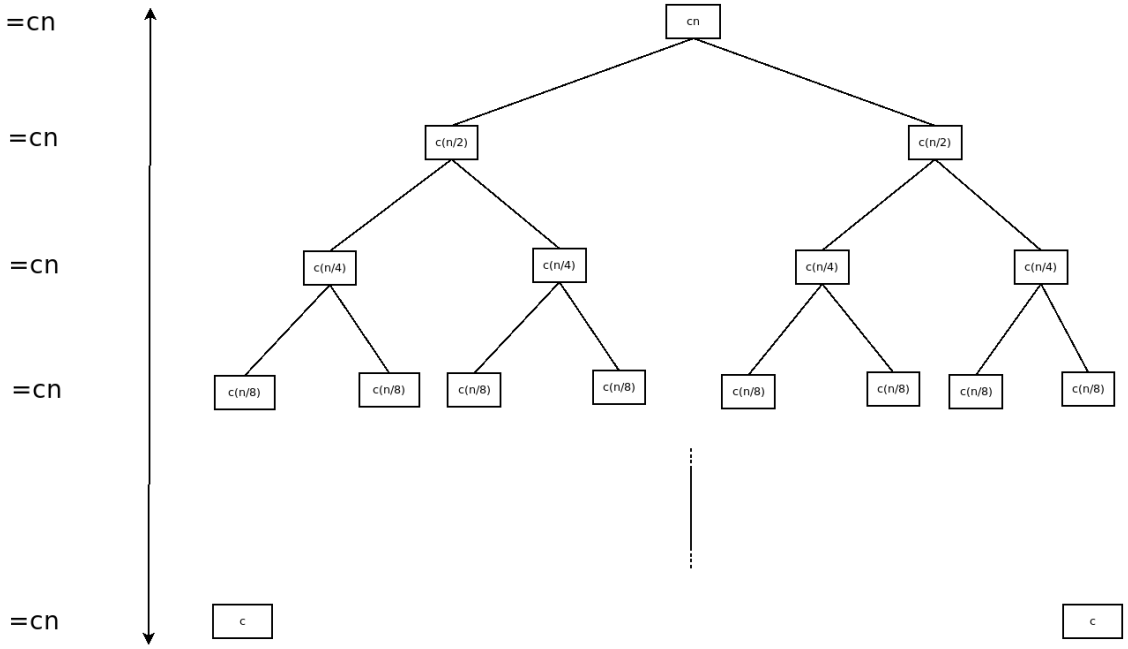
Exercise 5

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

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

Solution

1. ***Recursion tree***



There are 2 recursive calls at each step, so at the $i-th$ step there will be 2^i recursive calls, each one of them will take time $c(\frac{n}{2^i})$ and so the cost of each call is always cn. The tree will be high $log_2 n$. So

$T_1(n) = \sum_{i=0}^{\log_2 n} cn = cn \log_2 n \in O(n \log_2 n)$.

Substitution method

Let's assume that $T_1(n) \in= O(n \log_2 n)$ and let's chose $cn \log_2 n$ as a representative for the class $O(n \log_2 n)$ and $c'n$ for the class $O(n)$.

Assume that for every $m < n \rightarrow T_1(m) \leq cm \log_2 m$

$T_1(n) \leq 2c\frac{n}{2}\log_2 \frac{n}{2} + c'n = cn \log_2 n - cn \log_2 2 + c'n = cn \log_2 n - n(c - c')$

If $c - c' \geq 0$ so $c' \leq c$

$T_1(n) \leq cn \log_2 n \in O(n \log_2 n)$

2. ***Recursion tree***

At the first step there will be 2 recursive calls: one on $\lceil \frac{n}{2} \rceil$ and the other one on $\lfloor \frac{n}{2} \rfloor$. At the second step there will be 4 recursive calls, respectively on $\lceil \frac{\lceil \frac{n}{2} \rceil}{2} \rceil$, $\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2} \rceil$, $\lfloor \frac{\lceil \frac{n}{2} \rceil}{2} \rfloor$ and on $\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \rfloor$. And so on. As we can easily understand the branches of the tree have different length. The left branch is the longest of the tree and it has length equal to $\log_2 2n$. The right most branch is the shortest of the tree and it has length $\log_2 \frac{n}{2}$. So we can prove that:

$$
\begin{aligned}
T_2(n) &\geq \sum_{i=0}^{\log_2 \frac{n}{2}} c2^i = c\frac{2^{\log_2 \frac{n}{2}+1}-1}{2-1} = c2^{\log_2 n - \log_2 2 + 1} - c = c2^{\log_2 n} - c \\
&\geq cn - c \in \Omega(n) \\
T_2(n) &\leq \sum_{i=0}^{\log_2 2n} c2^i = c(2^{\log_2 2n+1} - 1) = c(2^{\log_2 n + \log_2 2 + 1} - 1) = c(2^2 2^{\log_2 n} - 1) \\
&\leq 4cn - c \in O(n)
\end{aligned}
$$

Hence since $T_2(n) \in \Omega(n) \land T_2(n) \in O(n) \rightarrow T_2(n) \in \Theta(n)$

Substitution Method

Let's prove that $T_2 \in O(n)$ by choosing $cn - d \in O(n)$ to represent the class $O(n)$ and $1 \in \Theta(n)$ to represent the class $\Theta(n)$.

$$
\begin{aligned}
T_2(n) &= T_2(\lceil \frac{n}{2} \rceil) + T_2(\lfloor \frac{n}{2} \rfloor) + 1 \\
&\leq c\lceil \frac{n}{2} \rceil - d + c\lfloor \frac{n}{2} \rfloor - d + 1 \\
&\geq cn - 2d + 1
\end{aligned}
$$

if $d \geq 1 \rightarrow T_2(n) \leq cn - d \rightarrow T_2(n) \in O(n)$

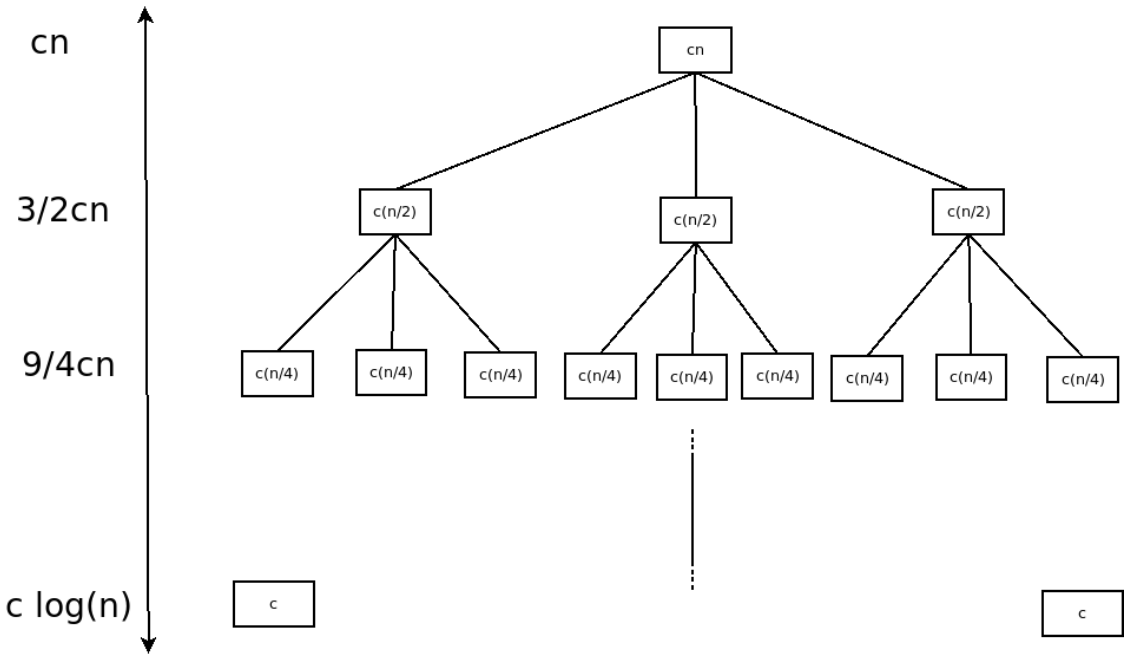
Now let's prove that $T_2(n) \in \Omega(n)$ by choosing $cn \in \Omega(n)$ to represent the class $\Omega(n)$.

$$
\begin{aligned}
T_2(n) &= T_2(\lceil \frac{n}{2} \rceil) + T_2(\lfloor \frac{n}{2} \rfloor) + 1 \\
&\geq c\lceil \frac{n}{2} \rceil + c\lfloor \frac{n}{2} \rfloor + 1 \\
&\geq cn + 1 \geq cn \in \Omega(n)
\end{aligned}
$$

So $T_2(n) \in \Omega(n)$.

Knowing that $T_2(n) \in \Omega(n) \land T_2(n) \in O(n) \rightarrow T_2(n) \in \Theta(n)$

3. ***Recursion Tree***



There are 3 recursive calls at each step, so at the $i-th$ step there will be 3^i recursive calls, each one of them will take time $c(\frac{n}{2})^i$ and so the cost of the $i-th$ call is $cn(\frac{3}{2})^i$ The tree will be high $log_2 n$. So

$$
\begin{aligned}
T_3(n) &= \sum_{i=0}^{\log_2 n} cn\frac{3}{2}^i = cn\sum_{i=0}^{\log_2 n}\frac{3}{2}^i \\
&= cn\frac{\frac{3}{2}^{log_2(n)+1}-1}{\frac{3}{2}-1} = 2cn(\frac{3}{2}^{\log_2 n+1}-1) = 2cn(\frac{3}{2}\frac{3}{2}^{\log_2 n}-1) \\
&= 2cn(\frac{3}{2}n^{\log_2 \frac{3}{2}}-1) = 2cn(\frac{3}{2}n^{\log_2 3-1}-1) \\
&= 3cn^{\log_2 3} - 2cn \in O(n^{\log_2 3})
\end{aligned}
$$

Substitution Method

Let's assume that $T_3(n) \in O(n^{\log_2 3})$ and let's chose $cn^{\log_2 3}$ as a representative for the class $O(n^{\log_2 3})$ and $c'n$ for the class $O(n)$.

$T_3(n) \leq 3c(\frac{n}{2})^{\log_2 3} + c'n = 3c\frac{n^{\log_2 3}}{2^{\log_2 3}} + c'n = cn^{\log_2 3} + c'n$

We get stuck here because we have chosen the wrong representative for $O(n^{\log_2 3})$. Let's now chose $cn^{\log_2 3} - dn$.

$T_3(n) \leq 3c(\frac{n}{2})^{\log_2 3} - 3d\frac{n}{2} + c'n = 3c\frac{n^{\log_2 3}}{2^{\log_2 3}} - n(\frac{3}{2}d + c') = cn^{\log_2 3} - n(\frac{3}{2}d - c')$

If $\frac{3}{2}d - c \geq 0 \rightarrow c' \leq \frac{3}{2}d$

$T_3(n) \leq cn^{\log_2 3} \in O(n^{\log_2 3})$

4. ***Recursion tree***

There are 7 recursive calls at each step, so at the $i-th$ step there will be $cn^2\frac{7}{2^2}^i$ calls. The height of the tree is $\log_2 n$. So:

$$
\begin{aligned}
T_4(n) &= \sum_{i=0}^{\log_2 n} cn^2\frac{7}{2^2}^i = cn^2\sum_{i=0}^{\log_2 n}\frac{7}{2^2}^i \\
&= cn^2\frac{\frac{7}{2^2}^{\log_2 n+1}-1}{\frac{7}{2^2}-1} = \frac{2^2cn^2}{7}(\frac{7}{2^2}\frac{7}{2^2}^{\log_2 n}-1) \\
&= \frac{2^2cn^2}{7}(\frac{7}{2^2}n^{\log_2(\frac{7}{2^2})}-1) = \frac{2^2cn^2}{7}(\frac{7}{2^2}n^{(\log_2 7-\log_2 2^2)}-1) = \frac{2^2cn^2}{7}(\frac{7}{2^2}n^{(\log_2 7-2)}-1) \\
&= cn^{\log_2 7} - \frac{2^2cn^2}{7} \in O(n^{\log_2 7})
\end{aligned}
$$

Substitution Method

Let's assume $T_4(n) \in O(n^{\log_2 7})$ and let's chose $cn^{\log_2 7}$ as a representative for the class $O(n^{\log_2 7})$ and $c'n^2$ for the class $O(n)$. So:

$T_4(n) \leq 7c(\frac{n}{2})^{\log_2 7} + c'n^2 = 7c\frac{n^{\log_2 7}}{2^{\log_2 7}} + c'n^2 = cn^{\log_2 7} + c'n^2$

We get stuck here because we have chosen the wrong representative for $O(n^{\log_2 7})$. Let's now chose $cn^{\log_2 7} - dn^2$.

$T_4(n) \leq 7c(\frac{n}{2})^{\log_2 7} - 7d(\frac{n}{2})^2 + c'n^2 = 7c\frac{n^{\log_2 7}}{2^{\log_2 7}} - n^2(\frac{7d}{4} - c') = cn^{\log_2 7} - n^2(\frac{7d}{4} - c')$

If $\frac{7d}{4} - c' \geq 0 \rightarrow c' \leq \frac{7d}{4}$

$T_4(n) \leq cn^{\log_2 7} \in O(n^{\log_2 7})$