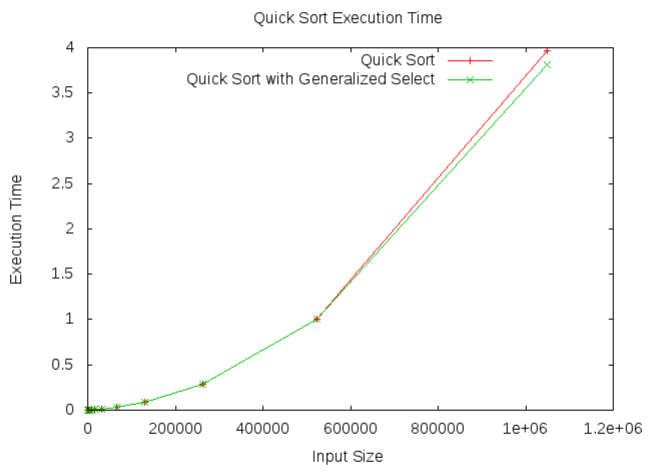
Sorting 2

Maria Grazia Berni

Implementation of Quick Sort using Select Algorithm

Exercise 1 2

In the Quick Sort algorithm, in order to deal with repeated values, I implemented the select algorithm as usual, splitting the array to sort in $\frac{n}{5}$ chunks of size 5, sorting them (using selection sort) to find the median, and then the median of the median. So the complexity is still $\mathcal{O}(n)$. To deal with repeated values I changed the partition procedure, using the three-partition method. When the chosen pivot is a value with some repetition, in the three partition procedure all the values equal the pivot are moved near it, at the beginning of the array, and then all this block is moved in its definitive position, and next iterations of quick sort will not deal with elements of this block. The complexity of three partition procedure is $\mathcal{O}(n)$. Regarding the overall complexity of the sorting procedure, quick sort without the select procedure performs $\Theta(n \log n)$ in case of balanced partition, while quick sort with the select procedure that doesn't deal with repetition always performs $\Theta(n \log n)$. When the select algorithm deal with repeated values, both the upper and lower bounds change. In fact it is a $\mathcal{O}(n \log n)$, and to derive a lower bound just imagine the case in which the array to sort contains only one value with n repetitions. In this case select and three partition procedure will be executed only one time, so $\Omega(n)$ is the most strict lower bound for this implementation of quick sort.



Exercise 3

Partitioning the array using the median of median obtained dividing the elements in chunks of 7, the minimum number of elements greater then the selected pivot will be:

$$4 * (\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2) >= \frac{2n}{7} - 8.$$

This holds even for the element which are smaller then the pivot. Thus, the size in which the problem is divided after the partition algorithm, is at most $n - \frac{2n}{7} - 8 = \frac{5n}{7} + 8$, which leads to the recurrence relation:

$$T(n) = T(\lceil \frac{n}{7} \rceil) + T(\lceil \frac{5n}{7} + 8) + \mathcal{O}(n)$$

which is still $\mathcal{O}(n)$. In fact, using the substitution method, choosing $c * n$ as representative of $\mathcal{O}(n)$ and assuming that $T(n) <= c * n$, this leads to the equation:

$$T(n) <= cn - \frac{c * n}{7} + 8 * c + n$$

which is true for

$$-\frac{c * n}{7} + 8 * c + n < 0$$

$$c > \frac{7n}{n - 56}$$

so the relation is proved with $n > 56$ and c as already calculated. In case we divide in chunks of 3 elements, the number of elements greater then the pivot will be:

$$2 * (\lceil \frac{1}{2} \lceil \frac{n}{3} \rceil \rceil - 2) >= \frac{n}{3} - 4.$$

same for the elements that are smaller. So, the size in which the problem is divided after the partition algorithm, is at most $n - \frac{n}{3} - 4 = \frac{2n}{3}48$, which leads to the recurrence relation:

$$T(n) = T(\lceil \frac{n}{3} \rceil) + T(\lceil \frac{2n}{3} + 4) + \mathcal{O}(n)$$

which is not $\mathcal{O}(n)$.

Exercise 4

The goal is to know the element that if the array was sorted would be in position i, knowing only the element that would be in position $n/2$. A possible linear algorithm that do it, executes the following steps:

```
1)read the median
2)if i= n/2 return the median
3)partition the array using the median as pivot
4) if i<n/2 search for the i-th element in the array A[0,n/-1]
   else search for the (n/2 - i)th element in the array [n/2+1,n]
```

```
def SELECT(A,i)
      m <- READ_MEDIAN(A)
      if (i = n/2) return m
      PARTITION(A,m)
      A1 <-A[0,n/2-1]
      A2 <- A[n/2+1,n]
      if(i>n/2) SELECT(A2,n/2-i)
      else SELECT(A1,i)
end def
```

Exercise 5

Here I provide a solution for the theoretical exercise that were not been solved during the lesson.

Es c

$$T(n) = 3T(n/2) + O(n)$$

TREE RECURRENCE ANALYSIS

LEVEL	DIMENSION	CALL COST	NUMBER OF CALLS	LEVEL COST
0	n	n	1	n
1	$n/2$	$n/2$	3	$3(n/2)$
2	$n/4$	$n/4$	3^2	$3^2(n/4)$
......				
i	$n/2^i$	$n/2^i$	3^i	$3^i(n/2^i)$
...				
$\log_2 n$	1	$T(1)$	$3^{\log_2 n}$	$3^{\log_2 n}$

because the "base" cost is $O(n)$

$$T(n) \leq \sum_{m=0}^{\log_2 n - 1} n\left(\frac{3}{2}\right)^i + 3^{\log_2 n}$$

$$\leq n \frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\frac{1}{2}} + 3^{\log_2 n}$$

$$\leq 2n\left(\frac{3^{\log_2 n}}{2^{\log_2 n}}\right) - 2n + 3^{\log_2 n}$$

$$\leq 2n^{\log_2 3} - 2n + n^{\log_2 3}$$

$$\leq 3n^{\log_2 3} - 2n$$

but $\log_2 3 > 1 \Rightarrow T(n) = O(n^{\log_2 3})$

SUBSTITUTION METHOD.

$T(n) = 3T(n/2) + O(n)$

GUESS THAT $T(n) \in O(n^{\log_2 3})$

REPRESENTATIVE FOR $O(n^{\log_2 3}) = cn^{\log_2 3} + 2n$

" " $O(n) = n$

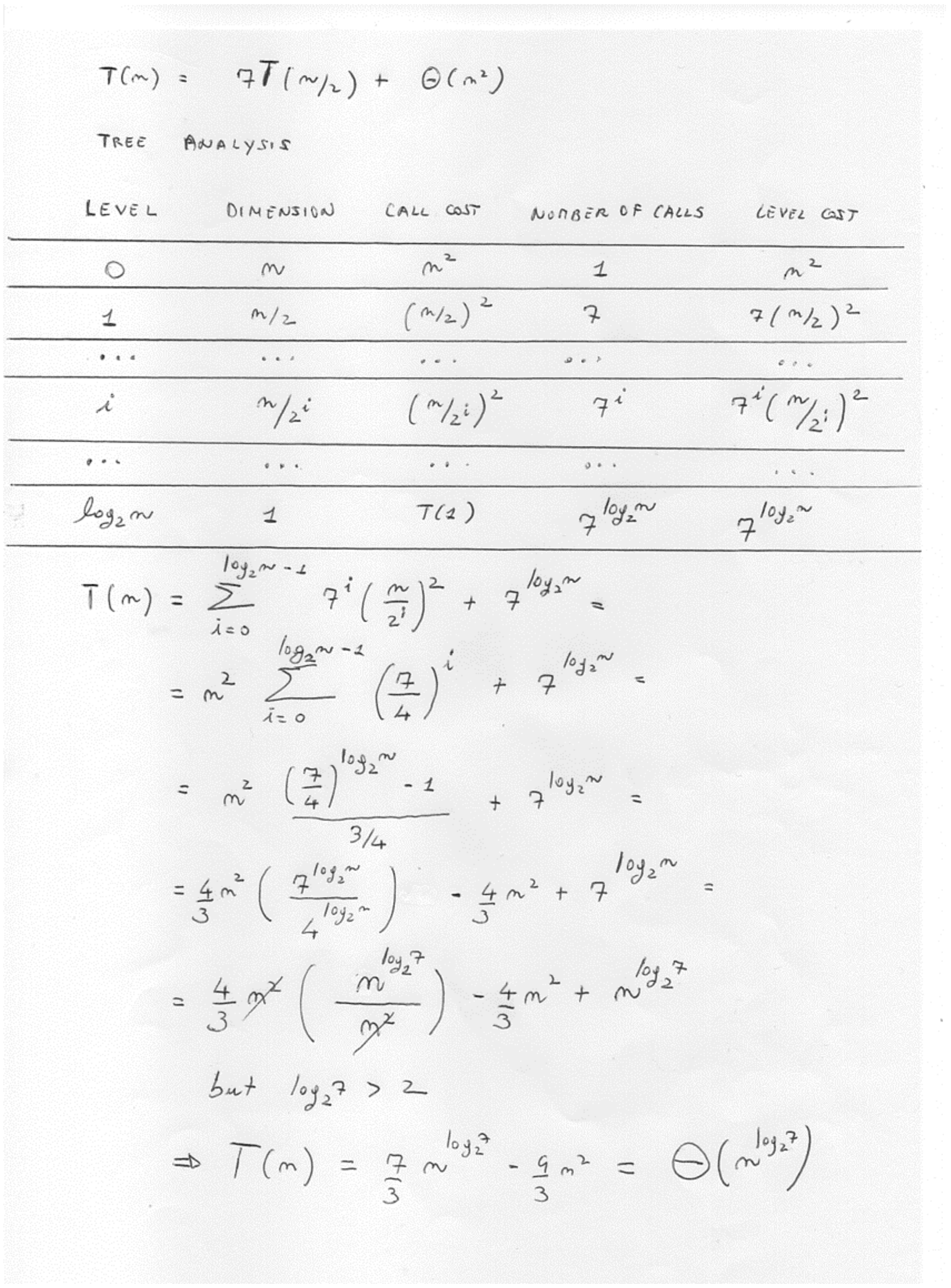
SHOW THAT $\exists c > 0, \exists n \geq 0 : T(n) \leq cn^{\log_2 3} + 2n$

$$T(n) = 3T(n/2) + O(n)$$

$$\leq \cancel{3} c \frac{n^{\log_2 3}}{\cancel{2^{\log_2 3}}} + 2\frac{n}{2} + n$$

$$\leq cn^{\log_2 3} + 2n$$

$$\Rightarrow T(n) \in O(n)$$



$T(n) = 7T(n/2) + \Theta(n^2)$

TREE ANALYSIS

LEVEL	DIMENSION	CALL COST	NUMBER OF CALLS	LEVEL COST
0	n	n^2	1	n^2
1	$n/2$	$(n/2)^2$	7	$7(n/2)^2$
...	...	...	...	...
i	$n/2^i$	$(n/2^i)^2$	7^i	$7^i(n/2^i)^2$
...	...	...	...	...
$\log_2 n$	1	$T(1)$	$7^{\log_2 n}$	$7^{\log_2 n}$

$$T(n) = \sum_{i=0}^{\log_2 n - 1} 7^i \left(\frac{n}{2^i}\right)^2 + 7^{\log_2 n} =$$

$$= n^2 \sum_{i=0}^{\log_2 n - 1} \left(\frac{7}{4}\right)^i + 7^{\log_2 n} =$$

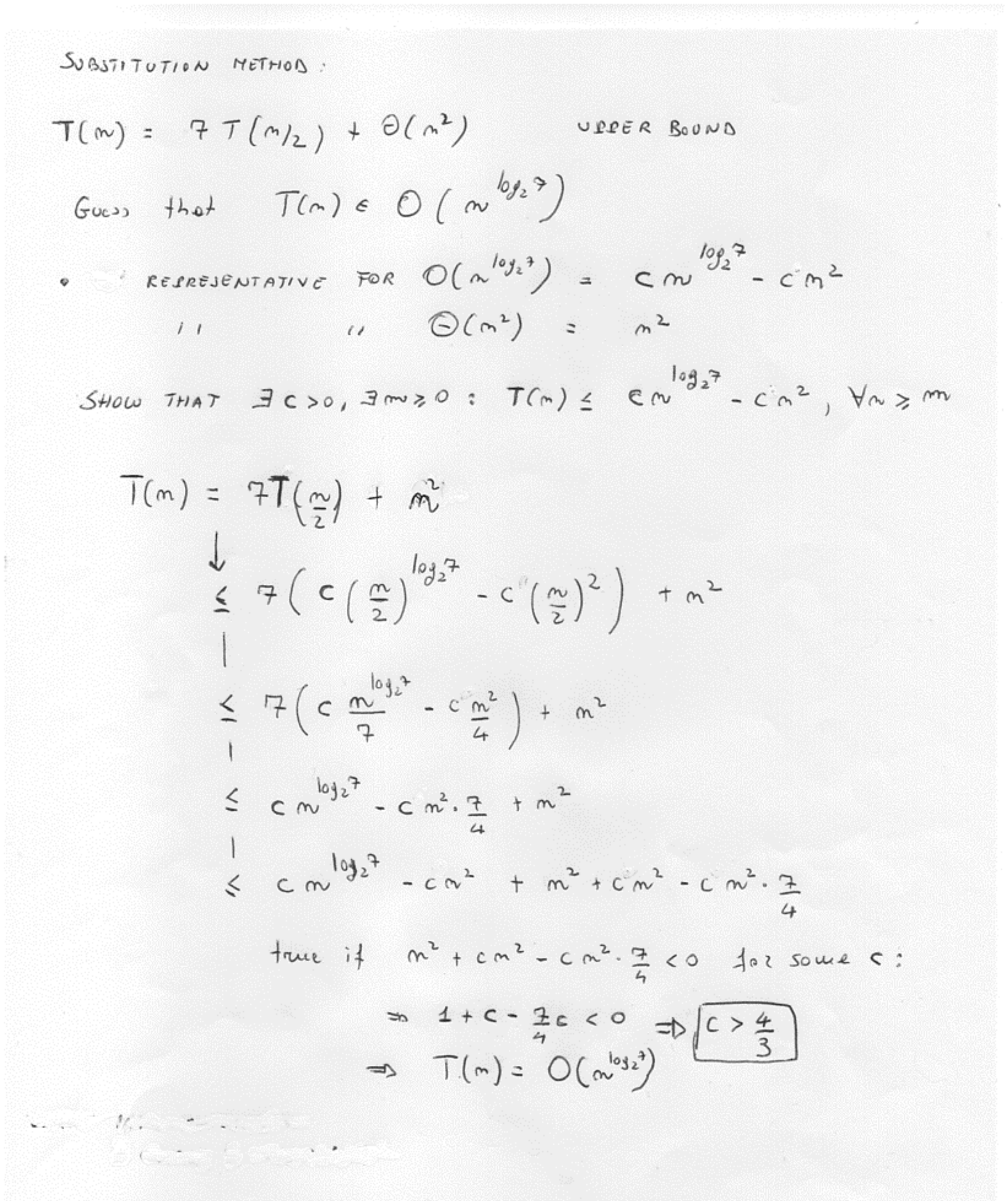
$$= n^2 \frac{\left(\frac{7}{4}\right)^{\log_2 n} - 1}{3/4} + 7^{\log_2 n} =$$

$$= \frac{4}{3} n^2 \left(\frac{7^{\log_2 n}}{4^{\log_2 n}}\right) - \frac{4}{3} n^2 + 7^{\log_2 n} =$$

$$= \frac{4}{3} \cancel{n^2} \left(\frac{n^{\log_2 7}}{\cancel{n^2}}\right) - \frac{4}{3} n^2 + n^{\log_2 7}$$

but $\log_2 7 > 2$

$$\Rightarrow T(n) = \frac{7}{3} n^{\log_2 7} - \frac{4}{3} n^2 = \Theta\left(n^{\log_2 7}\right)$$



SUBSTITUTION METHOD:

$T(n) = 7\,T(n/2) + \Theta(n^2)$ UPPER BOUND

Guess that $T(n) \in O(n^{\log_2 7})$

- REPRESENTATIVE FOR $O(n^{\log_2 7}) = cn^{\log_2 7} - cn^2$

  " " $\Theta(n^2) = n^2$

SHOW THAT $\exists c > 0, \exists m \geq 0 : T(n) \leq cn^{\log_2 7} - cn^2, \forall n \geq m$

$$
\begin{aligned}
T(n) &= 7T\left(\frac{n}{2}\right) + n^2 \\
&\leq 7\left(c\left(\frac{n}{2}\right)^{\log_2 7} - c\left(\frac{n}{2}\right)^2\right) + n^2 \\
&\leq 7\left(c\,\frac{n^{\log_2 7}}{7} - c\,\frac{n^2}{4}\right) + n^2 \\
&\leq cn^{\log_2 7} - cn^2 \cdot \frac{7}{4} + n^2 \\
&\leq cn^{\log_2 7} - cn^2 + n^2 + cn^2 - cn^2 \cdot \frac{7}{4}
\end{aligned}
$$

true if $n^2 + cn^2 - cn^2 \cdot \frac{7}{4} < 0$ for some c:

$\Rightarrow 1 + c - \frac{7}{4}c < 0 \Rightarrow \boxed{c > \frac{4}{3}}$

$\Rightarrow T(n) = O(n^{\log_2 7})$

LOWER BOUND

$T(n) = 7T(n/2) + \Theta(n^2)$

Guess that $T(n) = \Omega(n^{\log_2 7})$

REPRESENTATIVE FOR $\Omega(n^{\log_2 7}) = cn^{\log_2 7} + cn^2$

" " $\Theta(n^2) = n^2$

SHOW THAT $\exists c > 0, \exists n_0 \geq 0 : T(n) \geq cn^{\log_2 7} + cn^2, \forall n \geq n_0$

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$\geq 7\left(c\left(\frac{n}{2}\right)^{\log_2 7} + c\left(\frac{n}{2}\right)^2\right) + n^2$$

$$\geq cn^{\log_2 7} + c\frac{n^2}{4} \cdot 7 + n^2$$

$$\geq cn^{\log_2 7} + cn^2 + cn^2\frac{7}{4} + n^2 - cn^2$$

$$\Rightarrow cn^2\frac{7}{4} + n^2 - cn^2 > 0 \Rightarrow \frac{7}{4}c - c > -1$$

$$c > 0 \Rightarrow T(n) = \Omega(n^{\log_2 7})$$

$$T(n) = \Omega(n^{\log_2 7}) \ \& \ T(n) = O(n^{\log_2 7})$$

$$\Rightarrow T(n) = \Theta(n^{\log_2 7})$$