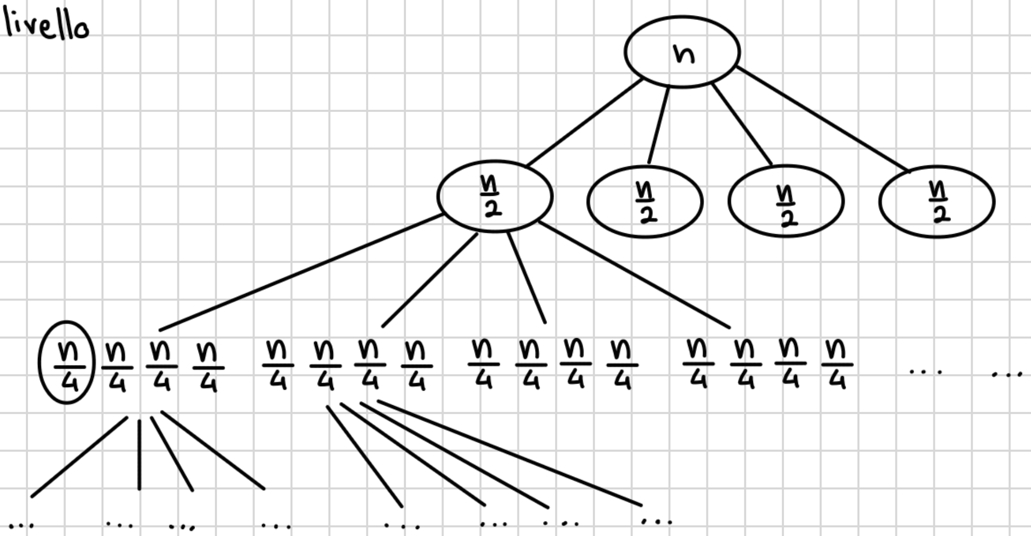
$$T(n) = \begin{cases} 4T\left(\frac{n}{2}\right) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Dove L = livello



1° L 16T

2° L 64T

3° L 256T

0 / n

1 / $\frac{n}{2}$

2 / $\frac{n}{4}$

...

i / $\frac{n}{2^i}$

$\log n$ / $\frac{n}{2^{\log n = 1}}$

$$T(n) = n + 4T\left(\frac{n}{2}\right) =$$
$$= n + 4\frac{n}{2} + 16T\left(\frac{n}{4}\right) =$$
$$= n + 2n + 16\frac{n}{2} + 64T\left(\frac{n}{8}\right) =$$
$$= \dots$$
$$= n + 2n + 4n + 8n + \dots + 2^{\log n - 1} n + 4^{\log n} T(1)$$
$$= n \sum_{J=0}^{\log n - 1} 2^J + 4^{\log n}$$

Rq $q \geq 2$

$$T_q(n) \leq \begin{cases} q \cdot T\left(\frac{n}{2}\right) + c \cdot n & \forall\, n \geq 3 \\ c & n \in \{0, 1, 2\} \end{cases}$$

The runtime for solving an instance of n elements, is no more than the runtime for solving q instances of $\frac{n}{2}$ elements + some linear time.

For MergeSort q is equal 2.

Theorem: In the case of MergeSort $q=2$, so It's possible to upperbound $T_2(n)$ with $O(n \log n)$.

$$T_2(n) \leq O(n \log n).$$