Exercises second lecture

Tommaso Tarchi

Ex.1

1. To sort a k-long list, insertion sort takes a $\Theta\left(k^2\right)$ worst-case time. Therefore, for $\frac{n}{k}$ lists it will take a total time of $\frac{n}{k}\Theta\left(k^2\right) = \Theta\left(nk\right)$.
2. Representing the algorithm as a tree (in which leaves are the k-long sublists), we can see that there are $\log n - \log k = \log \frac{n}{k}$ levels. On each level we have to merge lists for a total of n elements (just like in standard mergesort), so the worst-case time is $\Theta\left(n \log \frac{n}{k}\right)$.
3. We can see how putting k equal to any (positive) power of n would give a worst-case time of $\Omega\left(n \log n\right)$ (but not $O\left(n \log n\right)$). Therefore, also any function that is $\Omega\left(n\right)$ would give the same result. If, instead, we take $k = \log n$ (but any $\Theta\left(\log n\right)$ would be fine), we get:

   $$nk + n \log \frac{n}{k} = n \log n + n \log \frac{n}{\log n},$$

   in which $n \log n$ is clearly the dominant term, so that the worst-case time of the modified merge sort is $\Theta\left(n \log n\right)$. So $k \sim \log n$ is (asymptotycally speaking) the "largest" function of k for which the performances are ~equivalent.

EX.2

Yes, because we are sure that each level has all elements larger or equal to all the elements of the previous level; which implies in particular that each node has a larger value than its father node.

Ex.3

No, because it doesn't even have the largest element in the root.