Exercise 6.3

Saturday, 30 September 2023 13:44

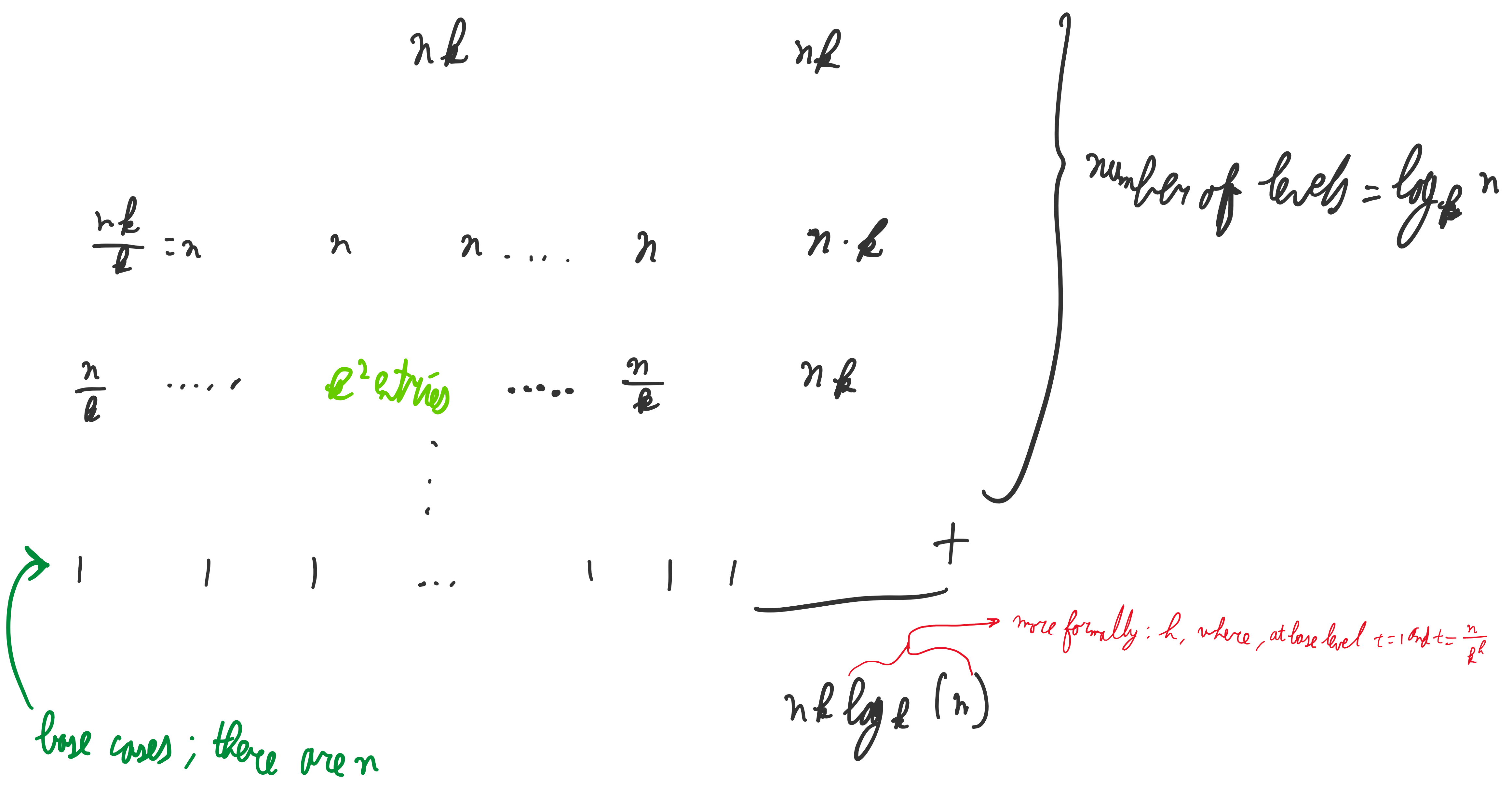
The running time can be described by the following recurrence:

better: use t, as n is already used

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ k\,T\left(\frac{n}{k}\right) + O(nk) & \text{if } n>1 \end{cases}$$

~~Implements merge as~~

```
~~for i = 1 to n~~
    ~~m = A_1[i]~~
    ~~for j = 2 to k~~
        ~~m = min(m, A_j[i])~~
```



$n + nk\log_k(n)$

$\downarrow$

$O(nk\log_k(n))$ ✓ verified by quiz

To achieve a running time of $O(n \log n)$ time, one can implement the merge procedure using a min priority queue. In this implementation, the priority queue contains one element from each of the k subarrays. More precisely, at any point in time, the queue will contain the minimal element of each subarray which has not been placed into the merged array yet. Obtaining the minimal element from the queue then gives the minimal element across all subarrays, and this element can then be placed as the next element of the merged array. After the element is placed into the merged array, the next-lowest element from the subarray from which the element was taken will be placed in the priority queue.
We note that looking up the minimal element from a priority queue can be done in constant time. Furthermore, inserting an element into the priority queue takes $\log k$ time; since there are n elements which (at some point) need to be inserted into the queue, this takes $n \log k$ time in total.
In other words, we need to replace all the nk-terms with $n \log k$ in the recursion tree above. This gives a total running time of

$$O\left(n \log_2(k) \log_k(n)\right) = O\left(n \log_2(k) \frac{\log_2(n)}{\log_2(k)}\right) = O(n \log_2(n)),$$ as we intended