Solution. We begin by initialising a set of events based on the given intervals and sorting them. For some interval $I_i \in \mathcal{I}$, where $i = 1, \ldots, n$ with $I_i = [a_i, b_i]$ where $a_i < b_i$, we set a_i as a start event and b_i as an end event. Repeat this for all intervals, making sure that each start and end event are corresponds to their original interval. This should be done in $O(n)$ time. If several intervals have end events at the exact same position as start events for other intervals, we need to append the start events first. This will take $O(n \log n)$.

Now, initialise an empty priority queue of available colours and a tracker for the maximum colours in the list, set to 0. For each event E in the sorted list, we use the following procedure.

- If E is a start event, check if there are any available colours in the list. If there are, assign it to the interval and remove it from the queue. This operation will be $O(\log n)$. Otherwise, increment the maximum tracker by one and add a new, distinct colour by some choice.
- If E is an end event, add the corresponding colour back to the queue of available colours.

Since there are $2n$ events, and each check is $O(\log n)$, our process will be $O(n \log n)$. This means our algorithm will be $O(n \log n)$.