F: Departures

Let's start with a slightly modified problem. Suppose that there are no trains running between weeks. We identify each train as an interval, whose endpoints are the departure and arrival times. Two trains may belong to the same group if and only if the corresponding intervals do not nest, i.e. one of them does not start before and ends after the other. Let's sort all the intervals (first by the beginnings and then by the ends). Let's create a list of empty groups to which we will assign intervals, so that each group does not contain a nested pair of intervals. For each group, we maintain the farthest end that is reached by any of the intervals in that group. We go through the sorted list of intervals one by one and append an interval into the first group (we use binary search to find such a group), which farthest end is at most as far as the end of the considered interval. Note that with such assignment, no two intervals in a group can nest. Also note that we can use binary search, because we always add intervals to the first matching group, which means that the ends of the groups form a descending sequence. It can be proved that this group assignment uses the minimal number of groups. We will omit this proof for now and return to it at the end.

Now let's return to the original problem. If the train is running between weeks, then let's consider it twice, as if it was running at the beginning and at the end of the week. Now we could use the same approach as before. However, there might be a problem. We need to prove that every interval which was copied will be assigned to the same group.

Let's observe that in the algorithm we'll start by assigning all copied intervals into groups (they start before the actual week). Then we'll move on to the next intervals until we finally start hitting second copies of the intervals. Note that they are long (sticking out beyond a week). Let's consider what influences the assignment of such an interval. It only depends on the intervals which ends reach farther. However, only other, previously occurring, copied intervals can affect the assignment, which means that the same intervals as at the beginning have an impact on the location of a given interval, i.e. we will assign them to the same group as at the beginning.

Let's return to the previous statement and prove it. We want to show that presented algorithm indeed uses minimal number of groups. Let's call the process of adding intervals with the same starting endpoint a *phase*. Suppose we've just added a first interval to a k-th group. We've done it because the $k-1$-th group had an interval that was added in one of the previous phases and reaches farther. And that interval was added because of some other interval in the $k-2$-th group and so on. This means that there are k intervals which are pairwise nested. Because of that there cannot exist an assignment into less than k groups. This proves that the presented algorithm uses minimal number of groups.