Time Complexity Explanation-

There are two methods in the main app, parseFile and printToConsole.

For parseFile-

This method parses the contents in file and keeps the content to create report in the memory map (a treemap which has keys as day and value as corresponding (urls, counts) in ascending order).

Since the number of days (k), is very less as compared to the unique urls (N), there would be minimal effect of sorting the number of days, time complexity: O(N), as k<< N and O(N.log(k)) is nearly equal to O(N).

For printConsole-

This method prints the report to the console. If there are 'k' days and a maximum of 'm' distinct hit rates in one of the day, the complexity would be O(klog(k) (mlog(m) + (n1 + n2+ ..nk)). Given that the number of distinct hit count values (m) are much smaller than the number of unique urls m << N, and number of days, k<<N, the time complexity would be almost O(N).

Thus the overall time complexity becomes O(N).

Another Approaches-

1. We can also directly hashmap, instead of treemap (to sort the hit rates, a treemap uses red-black tree internall), and then sort using Insertion Sorting (since the time complexity of almost sorted list, is O(n), best cases.
2. We can also implement the functionality using pyspark (map reduce), and do things even faster.