PolePosition:

In room D of the semifinals, the focus was on efficient array visitation. In both the level 1 and level 3 problems, greedy algorithms were necessary to avoid timing out. The level 2 problem was easier, requiring only a translation between data representations and an iteration through possible solutions.

In the level 1 problem (hyperlink) the solution is to find an arrangement of drivers that maximizes the position of a given driver in which that driver wins the race. Clearly, a brute-force check of all possibilities would have a computation time of O(N!) - which might be fast enough if the code were well written but is unnecessarily inefficient. To solve this problem properly, two facts need to be noticed:

1. Driver X, when placed in lane Y, cannot win the race if a faster driver is in a lower lane.
2. Driver X, when placed in lane Y, is more likely to win if, given two drivers slower than driver X, the slowest of those two is in the lowest lane of the two lanes those drivers occupy.

This information yields that the best possible configuration for a driver is given by sorting the drivers by speed in descending order and placing them on the racetrack in that order. If this configuration does not end in a win or tie by the driver, then the best configuration in which the driver is one lane lower is simply given by swapping the driver with the driver one lane lower.

Thus, if we have N drivers, the problem can be solved easily in a computation time of about N squared, where N is the number of drivers.